Vers une approche d'ingénierie multi-agents à base de lignes de produits logiciels

Dounia Boufedji

THÈSE DE DOCTORAT DE SORBONNE UNIVERSITÉ

Spécialité
Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris 6)

Présentée par

Dounia BOUFEDJI

Pour obtenir le grade de

DOCTEUR DE SORBONNE UNIVERSITÉ

Sujet de la thèse :

Vers une approche d'ingénierie multi-agents à base de lignes de produits logiciels

soutenue le 14 décembre 2020 devant le jury composé de :

Mme. Zahia GUESSOUM	Directrice de thèse
Mme. Aicha AISSANI MOKHTARI	Co-directrice de thèse
M. René MANDIAU	Président du jury
M. Stéphane GALLAND	Rapporteur et examinateur
M. Abdelhak-Djamel SERIAI	Rapporteur et examinateur
Mme. Anarosa ALVES FRANCO BRANDÃO	Examinatrice
M. Mohamed AHMED NACER	Examinateur

Remerciements

Je tiens à remercier :

Ma directrice de thèse Madame Zahia GUESSOUM qui m'a proposé le sujet de thèse et dirigée vers ce monde attrayant et combien motivant de la recherche. Le sujet proposé est tellement formateur et intéressant que la recherche en elle-même est dans le sens du développement actuel de la technologie.
Merci Madame d'avoir été compréhensive et merci d'avoir organisé la relecture de mon document de thèse et de m'avoir permis de faire ma soutenance malgré les difficultés d'éloignement nonobstant la fermeture des frontières en raison de la pandémie de Coronavirus.

Je n'omettrai surtout pas, de remercier ma co-directrice de thèse Madame AISSANI MOKHTARI Aicha, qui a été particulièrement à mon écoute, qui m'a reçue à plusieurs reprises dans son bureau au détriment de ses charges d'enseignement et qui m'a soutenue et encouragée dans mon cursus Doctoral.

Je tiens à remercier également, Monsieur René MANDIAU, Président du Jury ainsi que les membres du jury, Madame Ana-rosa ALVESFRANCO BRANDÄO, Messieurs Stéphane GALLAND, Abdelhak-Djamel SERIAI , et Mohamed AHMED-NACER , pour leurs précieux conseils et remarques qui m'ont permis de prendre un recul positif et didactique par rapport à mon travail de recherche.

Je n'omettrai pas non plus, de remercier Messieurs Stéphane GALLAND, Rapporteur et examinateur et Abdelhak-Djamel SERIAI, Rapporteur, examinateur grâce auxquels j'ai pu améliorer la structuration de mon manuscrit

et mieux mettre en avant et en valeur mon travail de recherche.

Je remercie également, les membres du PDI IRD dont Cheikho, Dorra, Lydia et particulièrement Messieurs Christophe CAMBIER et Nicolas MARILLEAU, qui m'ont accompagnée dans ma Formation Doctorale et m'ont prodigué à tout moment aide et soutien et de très bons conseils relatifs au métier de chercheur, tout en m'enseignant le devoir de se relever face aux difficultés quoi qu'il arrive.

Je n'oublierai pas non plus, de remercier Monsieur Bertrand GRANADO, l'ancien Directeur de l'Ecole Doctorale ainsi que Monsieur Habib MAHREZ, Directeur actuel de ledite, pour leur précieuse sollicitude et générosité d'écoute.

Encore merci également à mes directrices de thèse pour la relecture du manuscrit. Merci à Monsieur Abdelhak-Djamel SERIAI, chargé de la validation de la copie finale du manuscrit.

De même que mes remerciements sont destinés au Docteur Seifeddine KRAMDI, qui a été pour moi d'un très grand soutien dans les moments les plus cruciaux de ma thèse, me faisant part de son expérience en tant que chercheur, tout en me prodiguant des conseils inestimables qui me serviront pour le reste de ma carrière et que j'ai pu mettre en avant le jour même de ma soutenance.

Dédicaces

Je dédie ma présente thèse de doctorat en sciences informatiques, aux trois personnes qui sont pour moi les êtres les plus chers que j'ai sur terre.

Mes chers parents et ma chère sœur, que je ne remercierai jamais assez d'avoir été là pour moi durant ce parcours et de m'avoir toujours encouragée, et soutenue.
Vous avez été les piliers inébranlables qui m'ont permis de tenir bon et surtout de ne jamais abandonner quoi qu'il arrive. Merci encore de ne m'avoir jamais lâché la main même dans les moments les plus difficiles, et si j'y suis arrivée c'est grâce à votre soutien et votre amour que ce diplôme de doctorat est devenu le nôtre, et une réalité palpable.

Vous avez vu naître cette thèse et l'avez suivie jusqu'au bout. Durant mes travaux de recherche, j'ai travaillé sur les similitudes et la variabilité, et j'ai bien appris et retenu que si la variabilité fait partie de la vie tout comme l'a été mon parcours de thèse, le point commun lui, a construit le cœur autour duquel tout ce qui gravite peut tenir, comme ne peut le faire que le socle familial.

Vous avez toujours été à mes côtés et vous êtes ce cœur dont la générosité de présence nous a permis à tous, de traverser solidairement et en toutes circonstances, les aléas et difficultés qui ont émaillé mon parcours Doctoral.

Je continuerai et persisterai à vous rendre fiers de ce parcours, en continuant ma carrière dans la recherche, grâce à cette thèse qui m'a réellement forgée et appris à ne jamais abandonner, à aller toujours de l'avant et surtout à me relever avec

courage et optimisme face aux aléas de la vie. Durant mon parcours Doctoral, on m'a, de tout temps, confié que la préparation d'une thèse est certes didactique et formatrice, mais demeure émaillée de dures réalités avérées et très éprouvantes en particulier sur le plan moral.
Pour ma part, j'ai su me montrer forte grâce à votre amour pour voir le bout du tunnel et assurer enfin une soutenance réussie grâce à votre soutien indéfectible durant tout un septennat.
Merci, encore une fois de m'avoir encouragée, accompagnée et surtout de n'avoir jamais renoncé à croire en moi.
Je saisis cette opportunité pour remercier mes tantes et mes oncles, ainsi que mes amis qui m'ont de tout temps soutenue et encouragée.
Je remercie particulièrement mes tantes Sadjia, Saïda, Zoubida, Houria et mon oncle Moussa

Je remercie également mes amis Lydia BENKAÏDALI, Meriem et Réda BENAÏSSA, et Mohamed BENAÏ.
Je n'omettrai point, par affection particulière, de dédier également cette thèse de doctorat à mes feus grands parents maternels et paternels ; qu'ils reposent en paix.

Dounia

Résumé

Les Systèmes Multi-Agents (SMA) représentent une technologie et une solution idéale qui a déjà fait ses preuves pour la modélisation de systèmes complexes. L'ingénierie des SMA ou AOSE (Agent Oriented Software Engineering) offre différentes méthodologies, méta-modèles, templates et patrons de réutilisation qui facilitent leur développement et accélère leur acceptation au sein de l'industrie du logiciel.

Cependant, les approches existantes de l'ingénierie des SMA ne permettent pas la gestion et le développement d'applications similaires dites familles de SMA. Ces applications présentent des points communs, ainsi que des différences appelées variabilité. La gestion de la variabilité peut se faire à différents niveaux (conception, développement et autre), sauf qu'elle n'est pas prise en considération dans les approches existantes.

Afin de pallier à cette absence de gestion de variabilité au sein des familles multi-agents au niveau des approches orientées agents, l'ingénierie de LdPL (Lignes de Produits logiciels) s'avère être la solution idoine pour laquelle la gestion de la variabilité reste un élément clé.

Dans ce contexte, l'exploitation des techniques d'ingénierie de LdPL dans le cadre d'approches multi-agents est connue sous l'appellation d'ingénierie MAS-PL (Multi-agent systems Product Lines).

Le présent sujet de thèse s'inscrit dans cette thématique d'approches MAS-PL afin d'améliorer la gestion de la variabilité au sein de familles de SMA, et par conséquent rendre meilleurs les aspects de réutilisation s'articulant autour de la variabilité.

C'est ainsi que notre approche qui repose sur le processus général des LdPL, repousse les limites des approches MAS-PL actuelles. L'originalité de notre approche est qu'elle permet une réutilisation indépendante du domaine d'application. Pour ce faire, nous avons considéré par exemple non pas un seul mais plusieurs méta-modèles issus de l'ingénierie orientée agents, et ce contrairement aux approches MAS-PL telles que PASSI-PL ou encore GAIA-PL qui ne prennent en considération qu'un seul méta-modèle sans pour autant que ces concepts y soient spécifiés en termes de la variabilité.

Cette variabilité pouvant être spécifiée par exemple par les modèles de caractéristiques ou FM (feature model) qui sont les plus utilisés dans ce domaine, et que nous avons par conséquent utilisé. Par ailleurs, la particularité de notre approche est que nous avons procédé à un découpage de l'analyse (resp. implémentation) du domaine en analyse (resp. implémentation) générique (réutilisable indépendamment du domaine) et spécifique.

Ce découpage apparait au sein du FM ainsi qu'au niveau des artefacts d'implémentation. De ce fait, le présent sujet de thèse offre un double intérêt pédagogique et scientifique, tant pour le milieu universitaire et celui de la recherche scientifique que dans le domaine de l'industrie du

logiciel visant à exploiter au mieux les techniques de SPLE pour la communauté des concepteurs et développeurs de SMA.

Sur la base de cette approche, nos travaux de recherche ont été illustrés par trois cas d'études (la gestion de l'emploi du temps, la vente de livres, et le multi-agent Contest).

Aussi, il est utile de souligner que l'évaluation de notre approche s'est articulée autour de la dérivation d'une ligne multi-agents pour le Contest. C'est ainsi que cette ligne multi-agents a été expérimentée en dérivant un sous-ensemble des variantes sur la base de l'ensemble des configurations valides obtenues.

De même que les variantes ont été déployées après leur dérivation montrant ainsi que chacune d'entre elles est simulée au sein de l'environnement du Contest.

En effet, les résultats ont montré que notre approche améliore les taux de réutilisation des caractéristiques (resp. d'implémentations).

Abstract

Multi-Agent Systems (MAS) represent a technology and an ideal solution that has already proved positive for the modeling of complex systems.
The engineering of MAS or AOSE (Agent Oriented Software Engineering) offers different methodologies, meta-models, templates and reuse patterns that facilitate their development and accelerate their acceptance within the software industry.
However, the existing approaches to MAS engineering do not allow the management and development of similar applications known as MAS families. These applications have some commonalities, as well as differences called variability.
The management of variability can be done at different levels such as design and development, except that it is not taken into account in existing approaches.
In order to compensate for the lack of variability management within multi-agent families at the level of agent-oriented approaches, SPL (Software Product Lines) engineering turns out to be the appropriate solution for which the management of variability remains. a key element.
In this context, the exploitation of SPL engineering techniques within the framework of multi-agent approaches is known as MAS-PL (Multi-agent systems Product Lines) engineering.
This thesis subject is part of this thematic of MAS-PL approaches meant to enhance the management of variability within families of MAS; what, consequently, improves the aspects of reuse revolving around variability.
This is how our approach, which is based on the general SPL process,in favor of an improvement, pushes the limits of current MAS-PL approaches.
Thence,the originality of our approach is the fact that it allows reuse independent of the application domain. To do this, we have considered ,for instance, not a single but several meta-models resulting from agent-oriented engineering, and this unlike MAS-PL approaches such as PASSI-PL or GAIA-PL which only take into account a single meta-model without these concepts are being specified in terms of variability.
This variability can be specified, for example, by the feature models or FM (feature models) which are the most used in this field, and which we consequently used. The particularity of our approach is that we have divided the analysis (resp. Implementation) of the domain into analysis (resp. Implementation), generic (reusable independently of the domain) and specific.
This breakdown appears within the FM as well as at the level of implementation artifacts.
This is how the present thesis subject offers a dual educational and scientific interest, both for academia and that of scientific research and in the field of the software industry aiming to make the best use of SPLE techniques for the community of designers and developers of MAS.
Based on this approach, our research was illustrated by three case studies (management of the timetable, the booktrading, and the multi-agent Contest).
It is useful to underline that the evaluation of our approach was articulated around the derivation of a multi-agent line for the Contest. This is how this multi-agent line was tested by deriving a subset variants based on the set of valid configurations obtained.
Likewise, the variants were deployed after their derivation, thus showing that each of them is simulated within the Contest environment.
The results of our work have shown that our approach improves the reuse rates of characteristics.

Table des matières

Table des figures

Liste des tableaux

Première partie

Introduction générale

Introduction générale

Contexte et problématique

Les Systèmes Multi-Agents (SMA) représentent de nos jours une technologie et une solution idéale qui a déjà fait ses preuves pour la modélisation et la simulation de systèmes complexes.

L'ingénierie orientée agents ou AOSE (Agent Oriented Software Engineering) est une branche d'ingénierie logicielle qui permet d'analyser, de concevoir et de développer des SMA.

Cette branche d'ingénierie a prouvé son efficacité pour la modélisation des systèmes complexes, en proposant entre autres, différentes méthodologies, méta-modèles, templates et patrons de réutilisation afin de faciliter le développement des SMA et d'accélérer leur acceptation au sein de l'industrie du logiciel.

Bien que les approches proposées jusqu'ici tentent de répondre aux besoins des concepteurs et développeurs des SMA, un des problèmes qui se pose est qu'elles présentent un faible taux d'acceptation industrielle en comparaison avec les approches utilisées pour le développement d'autres types d'applications que les SMA.

Une des raisons est que, dès lors que la conception des SMA est une tâche complexe qui n'est pas entièrement contrôlée par un processus encore validé et accepté par l'industrie du logiciel, les SMA ne font toujours pas partie du courant dominant du développement d'applications d'entreprise [26].

Beaucoup de recherches scientifiques se sont focalisées sur les différents aspects permettant de faciliter l'adoption des SMA, et de pallier à certaines faiblesses allant à l'encontre de cette adoption, notamment promouvoir la réutilisation.

Comme soulevé par Akbari et al. [27], une des raisons qui empêchent leur adoption est que la disponibilité et la commercialisation de plusieurs méthodologies orientées agents soient concurrentielles. Ce qui constitue un obstacle à leur adoption industrielle généralisée, car elle conduit à une confusion des utilisateurs de la méthodologie sur la méthode à adopter.

De plus, qu'en est-il des domaines d'application qui nécessitent la mise en place de plusieurs versions ou variantes de la même application ? Dans le cas du e-commerce par exemple, il faudrait être en mesure de développer des applications multi-agents similaires appelées aussi

familles de SMA.

L'objectif de ces variantes multi-agents, serait de répondre à divers besoins, que ce soit des besoins spécifiques d'utilisateurs, des besoins économiques ; ou encore d'autres types de besoins.

Un exemple de besoins économiques, serait de proposer une application de vente gratuite, et une autre qui serait payante, mais qui offrirait plus de fonctionnalités aux utilisateurs.

Un exemple relatif aux besoins spécifiques d'utilisateurs, serait de développer une application multi-agents de vente à prix fixes, à prix négociables ou de ventes aux enchères.

Le problème qui se pose avec le développement de ce type d'applications, est que le passage d'une variante à l'autre n'est pas toujours facile. Dès lors que pour répondre à un nouveau besoin, il faudrait effectuer des changements à plusieurs niveaux, sans pour autant que l'écart entre ces changements et leur implémentation soit réduit.

Par exemple, pour le passage d'une application multi-agents de vente à prix fixes à une application de vente aux enchères, il faudrait entre autres effectuer des changements au niveau du modèle d'interactions d'agents, mais aussi au niveau de l'implémentation du protocole d'interaction répondant à ce besoin.

Plus encore, d'autres types de besoins peuvent aussi engendrer des changements liés au niveau de la méthodologie, ou encore du modèle multi-agents utilisé dans le cas où ces derniers ne supporteraient pas des concepts nécessaires à la solution du problème.

Un exemple serait de passer d'une première variante d'applications multi-agents, qui aurait été développée selon un modèle organisationnel, basé sur le concept de rôles d'agents pour structurer le système ; à une seconde application multi-agents instaurant aussi des normes liées aux rôles.

Le concept de normes, n'étant pas supporté par le modèle ayant servi au développement de la première application ; celui-ci, ne pourrait donc pas permettre à la seconde variante de répondre au nouveau besoin, et par conséquent de supporter cette différence ou variabilité.

Tous ces exemples, montrent bien que le problème lié au développement de ce type d'applications est un problème de recherche récurrent. Dans ce contexte, de nombreuses pratiques de réutilisation du domaine multi-agents, ont tenté de répondre à ce type de problèmes.

Ces travaux de réutilisation s'inspirent de solutions utilisées en génie logiciel, et ont permis de faciliter le développement de ce type d'applications. Par exemple, les patrons de conception (design pattern) proposent des solutions à des problèmes récurrents dans le domaine des SMA. Le principe de Separation Of Concern (SOC) aussi a été utilisé dans de nombreuses méthodologies de modélisation de SMA.

Des librairies de code source ont également été mises à disposition des développeurs, pour des réutilisations au niveau du code source.

Cependant, le problème qui se pose avec ce type de pratiques de réutilisation, est que la réutilisation a lieu de manière opportuniste. En effet, même si le corps architectural lui, est

réutilisable, la variabilité liée aux différents niveaux comme nous avons pu le voir à travers les exemples cités (conception, développement et autre), nécessitera des modifications manuelles dans la majorité des cas, et plus particulièrement si l'opportunité de réutiliser un code source par exemple ne se présenterait pas.

Le présent sujet de thèse s'inscrit en ligne droite avec les problèmes de recherche cités, et par conséquent avec des aspects de réutilisation et de méthodologies AOSE liées aux applications multi-agents similaires ou familles de SMA. Le problème principal auquel nous tentons de répondre étant celui de faciliter leur développement.

Pour soulever de tels verrous de recherche, nous nous sommes intéressés à l'ingénierie des lignes de produits multi-agents ou MAS-PL (Multi-agent System Product Lines) ; qui est une sous branche de l'ingénierie AOSE dédiée au développement du type d'applications que nous avons décrites. Ces applications qui présentent ainsi des points communs ou similitudes, et des différences appelées variabilité. En effet, comme son nom l'indique, cette branche d'ingénierie est dédiée aux lignes de produits multi-agents. Une ligne de produits multi-agents représente une famille de SMA, et par conséquent un ensemble d'applications multi-agents similaires.

Cette branche d'ingénierie est issue de l'ingénierie de lignes de produits logiciels ou SPLE (Software Product Line Engineering), qui a la particularité d'utiliser une approche de réutilisation non opportuniste, mais plutôt prédictive.

Cette pratique, s'avère être une solution intéressante qui permettrait d'une part de réutiliser les points communs, et d'autres part d'exploiter la variabilité d'une ligne de produits multi-agents de façon à ce que celle-ci soit prise en compte dès les premières étapes d'ingénierie des SMA, supportant ainsi la dérivation du code source relatif aux variantes du système.

Dans ce contexte, le présent sujet de thèse propose d'adopter une nouvelle approche MAS-PL, visant à exploiter au mieux les techniques d'ingénierie de lignes de produits logiciels SPLE pour la communauté de concepteurs et développeurs des familles de SMA.

En effet, tout comme Akbari et al. [27], nous pensons que ce qui peut faire la force des méthodologies AOSE, serait d'adopter les dernières avancées et paradigmes existants de l'ingénierie, et de la réutilisation logicielle les plus connus et les plus exploités en industrie. Ce qui permettrait, de résoudre en partie le problème lié au faible taux d'acceptation des SMA en industrie.

Par cette démarche, il est également possible de mettre à la disposition d'une communauté d'étudiants, et de chercheurs une approche MAS-PL, afin de les aider à utiliser des pratiques récentes pour le développement de lignes de produits multi-agents.

En résumé, l'objectif principal du présent travail de recherche, est de faciliter le développement des applications multi-agents similaires, en proposant une approche de lignes de produits multi-agents MAS-PL. L'approche devrait, par conséquent, permettre la spécification des points communs et de la variabilité des lignes de produits multi-agents, tout en étant capable de réduire

l'écart existant entre la variabilité et son implémentation. Ce qui pourrait faciliter le développement de ce type d'applications, en allant vers une capitalisation d'expertise d'implémentation des SMA.

Plusieurs approches de lignes de produits multi-agents ou MAS-PL ont été proposées. Ces approches ont permis d'introduire la notion de variabilité dès les premières phases de l'ingénierie des familles de SMA. Ces approches, sont principalement construites entant qu'extensions de méthodes multi-agents existantes. Ce qui les rend plus facilement applicables par les habitués du domaine. Ces extensions sont réalisées de différentes façons. Par exemple, certaines méthodes proposent des extensions sous forme de stéréotypes pour l'annotation de la variabilité, qui est spécifiée sous forme d'options ou encore de choix alternatifs à faire entre les fonctionnalités du SMA.

Cependant, le principal inconvénient de ces approches, demeure dans le fait que les similitudes et la variabilité spécifiés, soient spécifiques au domaine d'application. En effet, comme nous avons pu le voir à travers les exemples que nous avons donnés préalablement, la variabilité peut éventuellement, être considérée à divers niveaux, comme celui des concepts issus des différents modèles du domaine multi-agents. Cependant, aucune des approches MAS-PL existantes ne tient compte d'un tel niveau de variabilité. Les modèles utilisés au sein des approches existantes sont principalement des modèles de caractéristiques FM (Feature Model). Ces derniers sont constitués de caractéristiques issues d'un seul niveau de variabilité et qui est celui du domaine d'application.

De plus, il en est de même pour les implémentations, qui ne sont également pas réutilisables dans d'autres domaines d'applications, dès lors qu'elles ne capitalisent pas l'expertise d'implémentation pour les lignes de produits multi-agents.

C'est pour ces raisons que nous avons pensé à proposer une approche MAS-PL, qui en plus de résoudre les problèmes cités relatifs au développement d'applications multi-agents similaires, serait capable de repousser certaines limites des approches MAS-PL existantes.

Contributions

Nos principales contributions s'articulent autour des points suivants :

1. *Extension de l'approche VOYELLES :*

   L'approche MAS-PL que nous proposons s'inscrit dans la catégorie d'extensions d'approches. La méthode que nous avons choisie d'étendre est VOYELLES [28, 29]. Cette approche a la particularité de permettre la recherche des éléments descriptifs de chacune des dimensions d'un système multi-agents de façon indépendante, tout en permettant de former un modèle cohérent. Les dimensions du SMA y sont décrites par chacune des voyelles A (Agent), E (Environnement), I (Interaction), et O(Organisation).

2. *Découpage des activités d'analyse et d'implémentation du domaine :*

   Notre approche MAS-PL s'appuie sur le processus général des lignes de produits logiciels, et couvre les deux phases d'ingénierie du domaine et d'application.

   L'originalité de notre approche est qu'elle permet une réutilisation indépendante du domaine d'application. C'est pourquoi, nous avons procédé à un découpage des activités d'analyse et d'implémentation du domaine.

   Cette distinction d'activités, a été réalisée dans le but d'obtenir des modèles de similitudes et de variabilité appelés FM (Feature Model), et artefacts d'implémentations réutilisables indépendamment du domaine d'application.

   Ceci a été possible en s'inspirant des lignes de produits multiples qui s'appuient sur des modèles de caractéristiques à plusieurs niveaux de variabilité (multi-niveaux).

3. *Introduction d'un nouveau niveau de variabilité indépendante du domaine d'application :*

   Afin de résoudre le problème relatif aux approches MAS-PL qui ne tiennent pas compte des similitudes et de la variabilité des concepts du domaine des SMA ; nous avons pensé à construire un modèle issu d'une analyse des points communs et variables du domaine multi-agents. Par conséquent, nous avons choisi d'utiliser un modèle de caractéristiques FM à deux niveaux. Chaque niveau traite ainsi une différente variabilité. Le premier niveau, est associé à une analyse des similitudes et de la variabilité du domaine multi-agents. Le second niveau, quant à lui, est associé à une variabilité d'un domaine d'application spécifique.

   Le niveau générique de notre modèle, est organisé par le biais de l'approche VOYELLES [28, 29], que nous avons choisie d'étendre grâce à une analyse des similitudes et de la variabilité des dimensions d'agents, d'environnement, d'interaction et d'organisation du système.

4. *Analyse de plusieurs méta-modèles du domaine multi-agents :*

   Une des particularités de notre travail, est que contrairement aux autres approches MAS-PL, il ne se limite pas à un seul méta-modèle. Car pour réaliser une analyse des similitudes et de la variabilité des dimensions d'agents, d'environnement, d'interaction et d'organisation du système ; nous avons dû considérer plusieurs méta-modèles issus des différentes connaissances du domaine des SMA (méthodologies, modèles, ontologies etc.).

   Nous avons procédé à une délimitation du domaine d'analyse constitué des connaissances relatives à chacune des caractéristiques d'agents, d'environnement, d'interaction et d'organisation.

   Cette particularité, permet à notre approche de se distinguer des autres. En effet, à l'inverse des approches MAS-PL où le modèle de caractéristiques FM est construit de zéro. Notre approche permet aux concepteurs et développeurs de la ligne de produits

multi-agents, de sélectionner l'ensemble des concepts qui sont nécessaires à la réalisation d'un modèle spécifique de caractéristiques. Notre premier niveau de modèle de caractéristiques pouvant donc être réutilisé de façon spécifique.

5. *Implémentations réutilisables indépendamment du domaine d'application :*
   Dès lors que chacune des caractéristiques du premier niveau de variabilité (générique) a été implémentée, et donc associée à un ensemble d'artefacts d'implémentations réutilisables, le développement d'une ligne de produits multi-agents pour un domaine d'application spécifique, ne sera pas réalisée de zéro. Nous avons en effet proposé une solution permettant de réduire l'écart entre la variabilité et son implémentation. Contrairement aux approches MAS-PL existantes qui ne proposent aucune implémentation de départ, la notre donne la possibilité aux développeurs de la ligne de produits multi-agents de réutiliser les artefacts dont ils ont besoin, et de les raffiner de façon spécifique au domaine d'application. L'ensemble des implémentations que nous avons réutilisées proviennent entre autres de plateformes multi-agents. L'ensemble de ces implémentations réunies au sein d'un seul et unique modèle, permet de faire un pas vers une capitalisation de l'expertise d'implémentation des familles de SMA.

Organisation du document

La suite du présent manuscrit de thèse est organisée comme suit :

Nous commençons tout d'abord par une partie relative aux bases et pré requis (background) nécessaires pour la compréhension de notre travail de recherche. Le premier chapitre de cette partie, concerne les principes élémentaires des lignes de produits logiciels, ainsi que les approches et les éléments importants, qui ont servi pendant notre recherche.

Le second chapitre quant à lui, présente les bases nécessaires relatives aux systèmes multi-agents. Ces derniers, nous ont servi pendant notre analyse du domaine des SMA.

Nous passerons ensuite à la partie relative à l'état de l'art. Cette dernière comporte le chapitre 3, qui présente les différentes approches d'ingénierie de lignes de produits multi-agents MAS-PL. Nous donnerons entre autres une classification de ces approches, ainsi qu'une analyse détaillée sur chacune de ces classes. Nous discuterons également des différents avantages et inconvénients de ces approches, avant de clôturer notre chapitre par une analyse et étude comparative sur la base des critères que nous avions jugés être importants.

Après cela, nous entamerons la partie du manuscrit relative à nos contributions. Celle-ci est constituée des chapitres 4 et 5.

Le chapitre 4 présente l'approche MAS-PL que nous proposons afin de résoudre les problèmes cités, et de pallier aux principaux inconvénients que présentent les approches présentées dans notre état de l'art. Ce chapitre donne une vue globale de notre approche qui comporte

entre autres notre positionnement par rapport aux approches du domaine. Nous présentons également en détail chacune des activités faisant partie de notre approche. Les exemples choisis permettront d'illustrer chacune de ces activités.

Le chapitre 5, présente les étapes qui nous ont permis de construire le modèle de caractéristiques réutilisable de façon indépendante du domaine d'application. Ce modèle étant le résultat obtenu par la première activité de notre approche. Nous donnerons l'ensemble des résultats obtenus, qui sont issus de notre délimitation du domaine, d'analyse des concepts, et de leur sémantique. A l'issu de l'analyse de chacune des dimensions du SMA relatives à l'approche VOYELLES, nous allons donner les modèles de caractéristiques d'agents, d'environnement, d'interaction et d'organisation qui ont été construits, et qui ont été regroupés au sein du modèle FM générique des caractéristiques des SMA. Nous finirons par des exemples sur les contraintes intégrées au niveau du modèle afin d'être en mesure d'obtenir des configurations valides pour les variantes multi-agents.

La dernière partie du manuscrit concerne la validation expérimentale, qui comporte le chapitre 6. Ce chapitre, présente les différents types d'évaluation que nous avons réalisées en expliquant la démarche adoptée. Nous présentons entre autres une évaluation de la faisabilité de notre approche par le biais de trois différents cas d'études. Nous présentons et analysons chacun des résultats obtenus selon un ensemble de métriques choisies de façon à pouvoir évaluer entre autres le gain d'effort de développement des lignes de produits multi-agents spécifiques au multi-agent Contest. Nous discutons aussi des avantages que présente notre approche, ainsi que ses inconvénients.

Nous concluons notre manuscrit avec un résumé sur nos principales contributions, ainsi qu'un ensemble de perspectives de recherche.

Deuxième partie

Bases et pré requis

Chapitre 1

L'ingénierie des LdP (Lignes de Produits logiciels)

1.1 Introduction

La réutilisation logicielle représente une solution méthodologique utilisée dans le développement de systèmes logiciels complexes qui devient de plus en plus indispensable. Les chercheurs dans ce domaine tentent de trouver les mécanismes les plus adéquats permettant entre autres aux industries une meilleure réutilisation et configuration des systèmes logiciels. Dans ce contexte, l'ingénierie des Lignes de Produits Logiciels (LdP) ou Software Product Line Engineering (SPLE) représente un paradigme dont l'objectif principal tend, non pas à la modélisation et le développement d'un seul système logiciel, mais plutôt d'un ensemble de systèmes logiciels appartenant au même domaine.

Dans ce chapitre, nous présentons les concepts de base de ce type d'ingénierie, tels que les concepts clés de *Caractéristique*, de *points communs* et de *Variabilité* lesquels sont des concepts primordiaux dans ce domaine. Nous détaillerons par la suite, les principales activités impliquées lors des différents niveaux d'exécution du processus relatives à l'ingénierie de lignes de produits logiciels. Nous expliquerons également de quelle façon et par quels moyens la modélisation et l'implémentation sont réalisées dans ce type de méthodologies. Nous fournirons notamment des exemples d'approches de modélisation telles que les modèles OVM (Orthogonal Variability Model), ou encore les modèles de caractéristiques FM (Feature Model) représentant les points communs et variables au sein d'une famille ou ligne de produits logiciels. Nous aborderons également un cas particulier de modèles de caractéristiques, et qui est celui du modèle multi-niveaux qui constitue un des éléments clés sur lequel s'appuie notre travail de recherche.

1.2 Les concepts de l'ingénierie des LdP

Dans cette section, nous commencerons par définir une Ligne de Produits Logiciels, et donnerons juste après les principaux concepts de l'ingénierie des LdP. Nous développerons par la suite les principes et apports liés à ce type d'ingénierie.

1.2.1 Définition des Lignes de Produits Logiciels

L'ingénierie de LdP occupe une place de plus en plus importante dans le domaine des méthodologies de réutilisation logicielle. Ceci est peut être dû à sa particularité dans la vision de la conception et de la modélisation des produits logiciels. Cependant, il serait utile, avant de présenter cela, de donner la définition de LdP proposée par Clements et al. [30], qui résume les aspects de cette vision particulière non pas limitée à un seul produit mais à un ensemble de produits logiciels : *"Une ligne de produits logiciels est un ensemble de produits logiciels qui partagent et gèrent un ensemble de caractéristiques satisfaisant les besoins spécifiques d'un segment de marché particulier ou une mission et qui sont développées à partir d'un même ensemble d'atouts essentiels décrits d'une manière prescrite"*

1.2.2 Principes et apports de l'ingénierie de LdP

L'ingénierie de Lignes de Produits Logiciels (LdP) permet de développer une ligne de produits logiciels avec un gain considérable en termes de temps, de coût, de qualité et d'effort [30] [1] [3].

Contrairement aux méthodologies proposant de modéliser et de développer un seul logiciel, l'ingénierie de lignes de produits logiciels s'intéresse à la modélisation et au développement de toute une ligne ou famille de produits logiciels. Une ligne de produits logiciels consiste donc en un ensemble de logiciels représentant des similarités (points communs) ainsi que des différences (points variables). Ces points communs et variables peuvent être modélisés de différentes façons. Cette modélisation s'articule autour des caractéristiques (Features) des produits logiciels de la ligne.

Bien que la mise en place de ce type de méthodologie nécessite plus de temps au démarrage avant la mise sur le marché des produits par rapport à l'ingénierie classique d'un seul produit, le temps imparti est très vite compensé par la réduction du coût de développement et l'amélioration de la qualité des produits [1]. Tel que présenté dans la Figure 1.1, le temps de mise sur le marché reste constant lorsqu'il s'agit du développement d'un unique produit, tandis que pour l'ingénierie de LdP, le temps de construction des éléments communs à tous les produits est important au départ mais il est très vite compensé grâce à la réutilisation des éléments

communs et à la dérivation des produits sur la base de la variabilité représentant l'élément clé de ce type d'ingénierie.

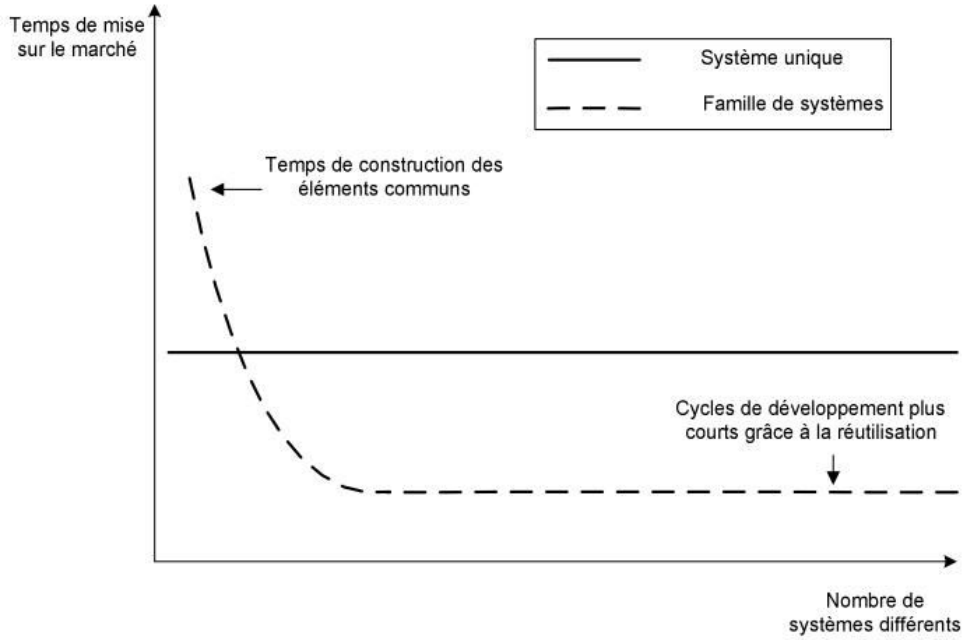


FIGURE 1.1 – Temps de mise sur le marché de produits avec et sans ingénierie de Lignes de Produits [1]

Le processus général d'ingénierie des lignes de produits logiciels comporte deux niveaux (voir e.g [31] et [1]) : *ingénierie du domaine* et *ingénierie d'application*. Ces deux niveaux d'ingénierie comportent différentes activités qui seront détaillées dans la prochaine section, allant de l'analyse du domaine à la dérivation de produit logiciel.

Tel qu'il a été précédemment souligné, au lieu de développer des systèmes logiciels en démarrant de rien, il nous serait donc possible de construire ces derniers à partir de parties logicielles réutilisables. De plus, au lieu de composer un logiciel de la même façon à chaque fois, celui-ci devrait plutôt s'adapter aux besoins des utilisateurs. Les développeurs logiciels seraient ainsi capables de sélectionner les différentes options du système désiré à partir de l'espace de configurations possibles afin de satisfaire les besoins des utilisateurs. L'approche de lignes de produits logiciels provient d'une personnalisation par construction de solutions individuelles basées sur un carnet ou portefeuille (portfolio) de composants logiciels réutilisables.

1.2.3 Les "Caractéristiques" dans l'ingénierie de LdP

Les caractéristiques ou features représentent l'une des préoccupations principales autour de laquelle s'articule l'ingénierie de lignes de produits logiciels. Il existe cependant plusieurs définitions du concept de caractéristiques dans ce domaine. Reprenons l'une de ces définitions [32] : *" Une caractéristique est une fonctionnalité ou un comportement end-user visible d'un système logiciel. Les features ou caractéristiques sont utilisées dans l'ingénierie de lignes de produits logiciels pour spécifier et communiquer les similitudes et les différences entre les produits intervenants, et de guider la structure, la réutilisation ainsi que la variation à travers toutes les phases du cycle de vie du logiciel".*

Une autre définition proposée par Kang [33] retient notre attention à ce sujet : *"Une Feature représente tout aspect important et distinctif ou caractéristique visible par les diverses parties prenantes".*

1.2.4 La "variabilité" dans l'ingénierie de LdP

La variabilité représente un des principes primordiaux de l'ingénierie des lignes de produits logiciels. Ce concept clé permet de développer des produits en réutilisant des artefacts, après configuration du produit. La configuration étant réalisée sur la base de sélection de caractéristiques désirées. La variabilité permet ainsi de personnaliser les produits en fonction des besoins spécifiques des utilisateurs, tandis que les besoins communs seront spécifiés par l'ensemble des caractéristiques communes à tous les membres de la ligne de produits. La variabilité représente ainsi selon Weiss et al. : *"une hypothèse sur la façon avec laquelle les membres d'une même famille peuvent être différenciés"* [34].

Svahnber et al. [35] la définissent comme suit : *"La variabilité du logiciel est la capacité d'un système logiciel ou un artefact à être efficacement étendu, modifié, personnalisé ou configuré pour être utilisé dans un contexte particulier".*

Une autre définition un peu plus récente proposée par Apel et al. [3] stipule que : *" La variabilité représente l'habilité à dériver différents produits à partir d'un ensemble commun d'artefacts."*

Selon Pohl et al. [1], la variabilité peut être de deux types : Temporelle ou Spatiale. Les auteurs définissent la première comme étant *"l'existence de différentes versions d'un artefact qui sont valides à des moments différents"* et la seconde comme étant *"l'existence d'un artefact sous différentes formes en même temps"*. La variabilité peut être donc détectée en observant l'évolution d'un artefact dans le temps. Cette évolution implique l'existence de différentes versions valides (variantes) de l'artefact en question.

La variabilité est gérée tout au long du processus général de développement de la ligne de produits, allant de son analyse à sa modélisation. Pour ce faire, il existe différentes approches

dont il sera question dans ce chapitre après la présentation des deux niveaux d'ingénierie connus dans le processus de développement. Nous verrons entre autres comment la variabilité est modélisée par le biais des features ou caractéristiques, en utilisant des approches de modélisation tel que le modèle de caractéristique ou feature model qui reste l'approche la plus utilisée dans le domaine de lignes de produits logiciels. Mais tout d'abord, il est nécessaire de donner la démarche à suivre pour la mise en place d'une approche de lignes de produits logiciels pourles industriels désirant ne plus passer par une implémentation à partir de zéro pour ces produits.

1.3 Mise en place d'une approche de LdP

Afin de mettre en place une approche de LdP à la place d'une approche où les produits sont développés de zéro, différents paramètres doivent être considérés. Ainsi, la mise en place d'une nouvelle approche peut dépendre par exemple du fait que l'entreprise possède ou pas quelques produits de la ligne préalablement développés. Selon Krueger et al. [2], il est possible de distinguer trois approches de transition d'une approche de développement classique à une approche de ligne de produits logiciels : *l'approche proactive*, *extractive* et *réactive*.

1.3.1 L'approche proactive

Ce type d'approche suppose qu'aucun produit de la ligne n'a déjà été développé au préalable. De ce fait, la ligne de produits logiciels est développée en suivant le processus général de développement (voir la prochaine section) en suivant étape par étape les activités d'analyse et de conception existantes. Avec ce type d'approche, les développeurs peuvent planifier la variabilité désirée pour la ligne de produits. Cependant, ce type d'approche est considéré comme étant idéaliste et trop académique car en pratique elle nécessite d'être combinée partiellement à des idées provenant des deux autres stratégies [3].

La Figure 1.2 illustre ce type d'approche. Cette approche comporte les activités suivantes :

— Analyse du domaine après délimitation de ce dernier (domain scoping). Cette activité sera expliquée au sein de la prochaine section,
— Sélection de l'approche d'implémentation de la ligne de produits qui sera utilisée,
— Implémentation de l'ensemble des produits de la ligne de produits. L'implémentation devra se faire pour toutes les caractéristiques importantes avant que le premier produit de la ligne ne soit dérivé.

1.3.2 L'approche extractive

Le point de départ de ce type d'approche se fait à partir d'une collection d'un ensemble de produits existants ayant été développés de zéro de façon classique. Cette approche est idéale

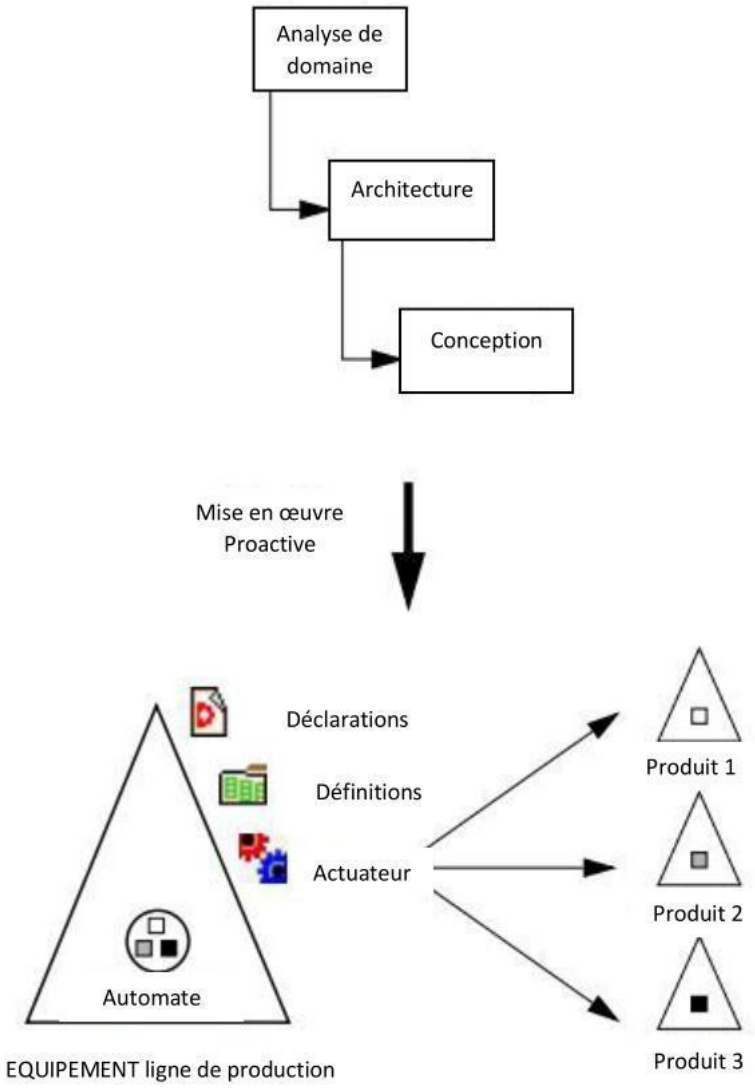


FIGURE 1.2 – Approche proactive pour la mise en place des Lignes de produits logiciels [2]

lorsque l'entreprise qui désire mettre en place une approche LdP possède déjà un ensemble de produits du même domaine. Le but principal de cette approche est de créer une transition d'un à plusieurs produits pour l'obtention d'une LdP plus structurée.

La Figure 1.3 illustre ce type d'approche. Cette approche comporte les activités suivantes :

— Identification des points de variations des produits existants, en se basant sur la connaissance du domaine ainsi que sur la définition des besoins,
— Extraction ou implémentation du cœur des fonctionnalités sous la forme d'artefacts réutilisables du domaine,
— Extraction et implémentation des variations en utilisant des techniques d'implémenta-

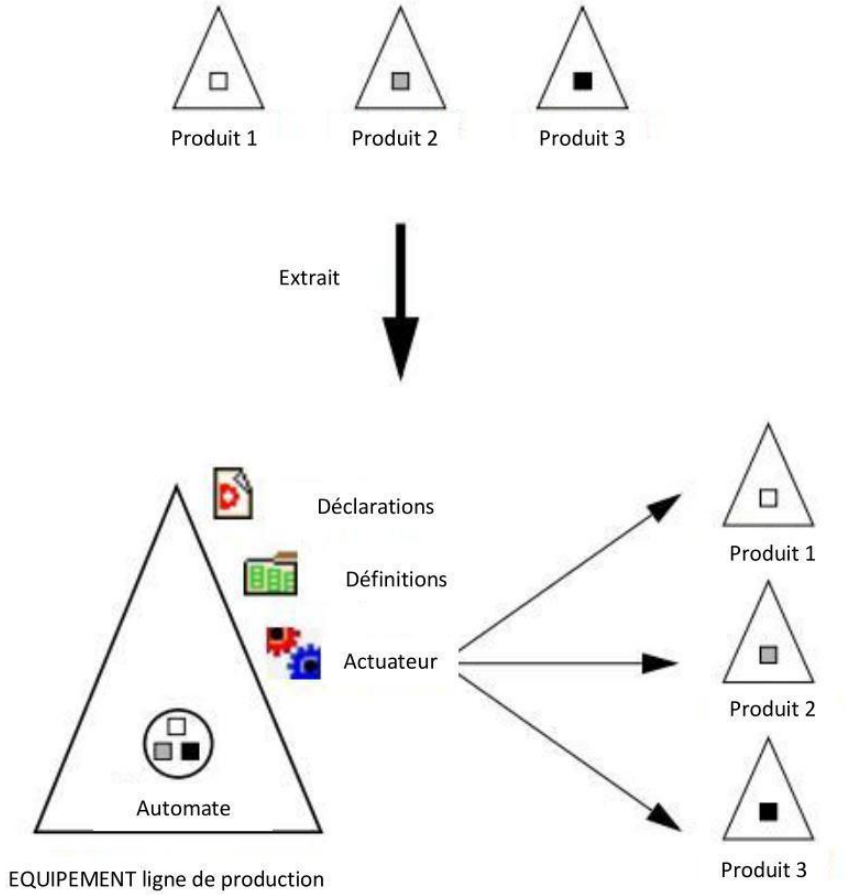


FIGURE 1.3 – Approche extractive pour la mise en place des Lignes de produits logiciels [2]

tions appropriées.

1.3.3 L'approche réactive

Cette dernière approche démarre d'un petit ensemble de logiciels facilement gérables. Cet ensemble qui représente la ligne initiale pourrait même ne comporter qu'un seul produit logiciel pour commencer. La ligne est ensuite étendue de façon incrémentale grâce à de nouvelles caractéristiques et artefacts d'implémentation. Cette approche s'aligne au rang des méthodes agiles de développement de logiciels.

La Figure 1.4 illustre ce type d'approche. Cette approche comporte les activités suivantes :

— Exploration et caractérisation des besoins relatifs à de nouveaux produits logiciels actuellement non inclus dans la ligne de produits existante. La nouvelle version de la ligne couvrira donc l'ensemble des produits logiciels .

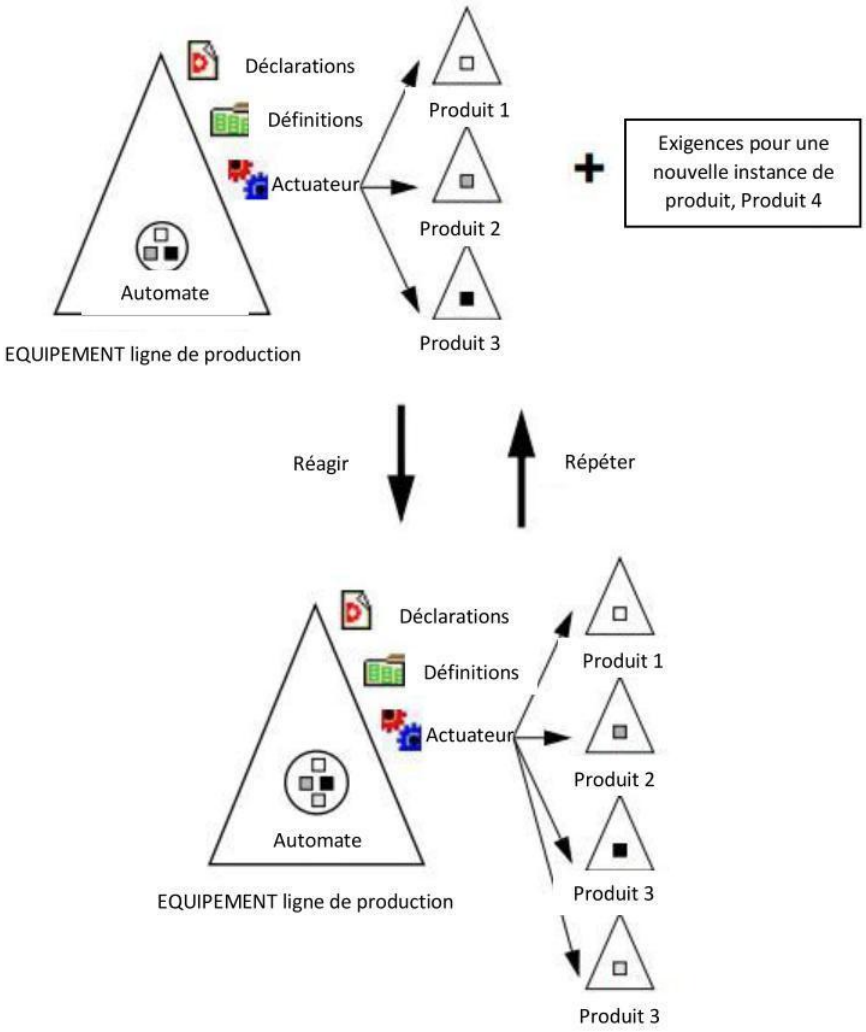


FIGURE 1.4 – Approche réactive pour la mise en place des Lignes de produits logiciels [2]

— Description des aspects delta menant à l'amélioration des produits
— Implémentation du delta de façon convenable et adaptée

1.4 Présentation du processus de développement des LdP

Le processus général des LdP est représenté dans la figure 1.5 avec ses deux niveaux connus : Ingénierie du domaine et Ingénierie d'application. Cette figure nous donne une vision globale du processus, tel que l'ingénierie du domaine a pour principe de se baser sur un développement pour la réutilisation, tandis que l'ingénierie d'application s'appuie sur un développement avec réutilisation. Ce principe de fonctionnement tend donc vers la compensation du temps de déve-

loppement des artefacts communs durant la première phase, et la réutilisation de ces derniers durant la seconde phase.

Les principales activités incluses dans le processus comportent : l'Analyse du domaine, l'implémentation du domaine, l'analyse des besoins et la dérivation de produit. Chacune de ses activités est détaillée par l'ensemble des étapes présentées dans la Figure 1.6.

1.4.1 Ingénierie du domaine

La première phase du processus permet d'effectuer tout d'abord une analyse du domaine suivie de sa modélisation et d'implémenter ensuite les assets réutilisables et de les tester. Il s'agit d'un développement pour la réutilisation. Le détail des activités de cette première phase comporte :

— **Analyse et modélisation du domaine**
Cette activité permet d'une part d'analyser la portée du domaine, ce qui est connu sous le nom de *"Domain Scoping"* ou *"Délimitation du domaine"*, et d'autre part de documenter le modèle correspondant. Cette dernière activité est connue sous le nom de *"Domain Modeling"* ou *"Modélisation du domaine"*.
L'analyse de la portée du domaine se fait en déterminant l'ensemble des produits couverts par la ligne de produits ; et en spécifiant les caractéristiques pertinentes qui devraient être implémentées entant qu'artefacts réutilisables.
Quant à la modélisation du domaine, elle permet de définir l'architecture de la ligne de produits logiciels, et pour ce faire il existe différentes façons de la réaliser.
L'architecture obtenue donne une vue structurelle sur les différents membres de la ligne de produits logiciels. Cette activité doit alors inclure les mécanismes de variabilité utilisés et identifier les parties réutilisables de l'architecture.
La modélisation consiste donc à documenter ou représenter le résultat suite à l'analyse du domaine. L'une des représentations la plus utilisée est de documenter la communalité et la variabilité par le biais d'un Feature Model (FM) [3] ou Modèle de Caractéristiques (MC) que nous allons présenter par la suite.

— **Implémentation du domaine et réalisation de tests**
Cette activité permet de développer dans un premier temps les artefacts réutilisables, qui correspondent aux caractéristiques ou features détectées, et de procéder à un ensemble de tests dans un second temps. Il existe en ingénierie de lignes de produits logiciels différents types d'artefacts tels que les artefacts d'implémentation, de test et de documentation [3]. L'implémentation de ces artefacts permet d'obtenir les différentes parties réutilisables de l'architecture formant ainsi des composants logiciels réutilisables. Cette activité peut être réalisée par le biais de différentes approches, parmi lesquelles nous retrouvons les

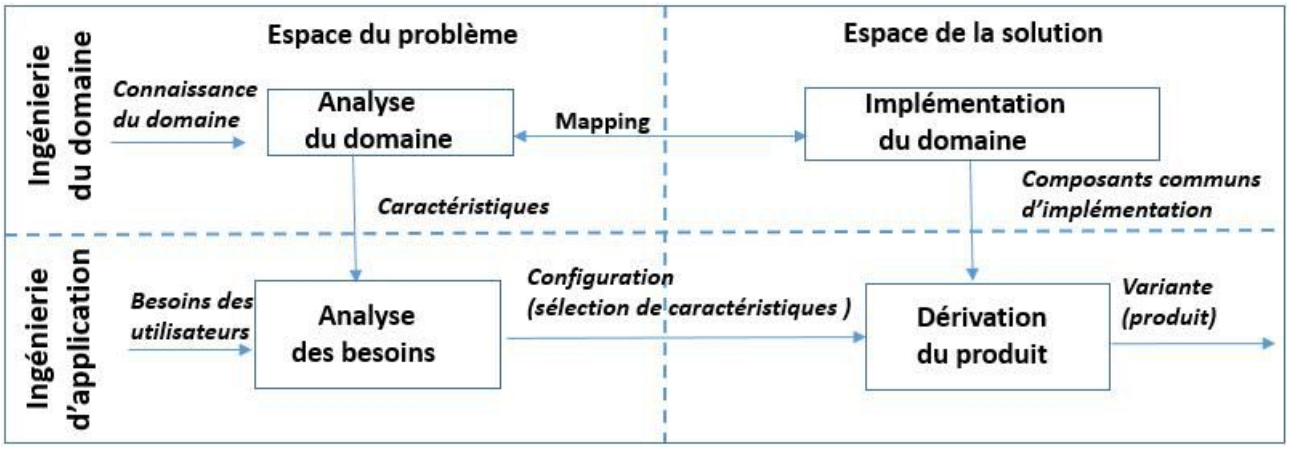


FIGURE 1.5 – Processus d'ingénierie de lignes de produits logiciels [3]

approches *annotatives* et *Compositionnelles*. Dans cette thèse, nous nous intéressons aux approches compositionnelles. Ces approches implémentent les caractéristiques ou features en unités composables, leur associent des fragments de code, et composent ces fragments pour implémenter une configuration de caractéristiques [3]. Les approches annotatives quant à elles, représentent la variabilité du code en utilisant des notations pouvant être lourdes à mettre en place.

Les tests vont servir à la validation et à la vérification des composants réutilisables obtenus à l'issue de l'activité précédente. Les composants sont testés par rapport à leur conformité aux spécifications établies au préalable, à l'architecture et aux artefacts de modèle.

1.4.2 Ingénierie d'application

La seconde phase du processus permet de sélectionner les caractéristiques du produit souhaité en créant ainsi une configuration, et ensuite de dériver le produit correspondant à la configuration. Lors de cette phase de développement, on parle de réutilisation car lors de la dérivation du produit, les artefacts implémentés lors de la première phase sont réutilisés pour développer le produit cible.

— **Analyse des besoins liés à l'application**

Ce type d'analyse permet de capturer et de considérer les besoins utilisateurs afin de produire une configuration personnalisée, et ce en sélectionnant les caractéristiques souhaitées. Cette activité d'analyse va passer en premier lieu par *l'Ingénierie des Exigences (EI)* de l'application, et par conséquent prendre en considération le concept de *but* (voir standard IEEE 830) pour la documentation des exigences, et dont l'importance a été mise en avant par Zave et.al [36] *" L'Ingénierie des Exigences (E.I) s'intéresse aux buts*

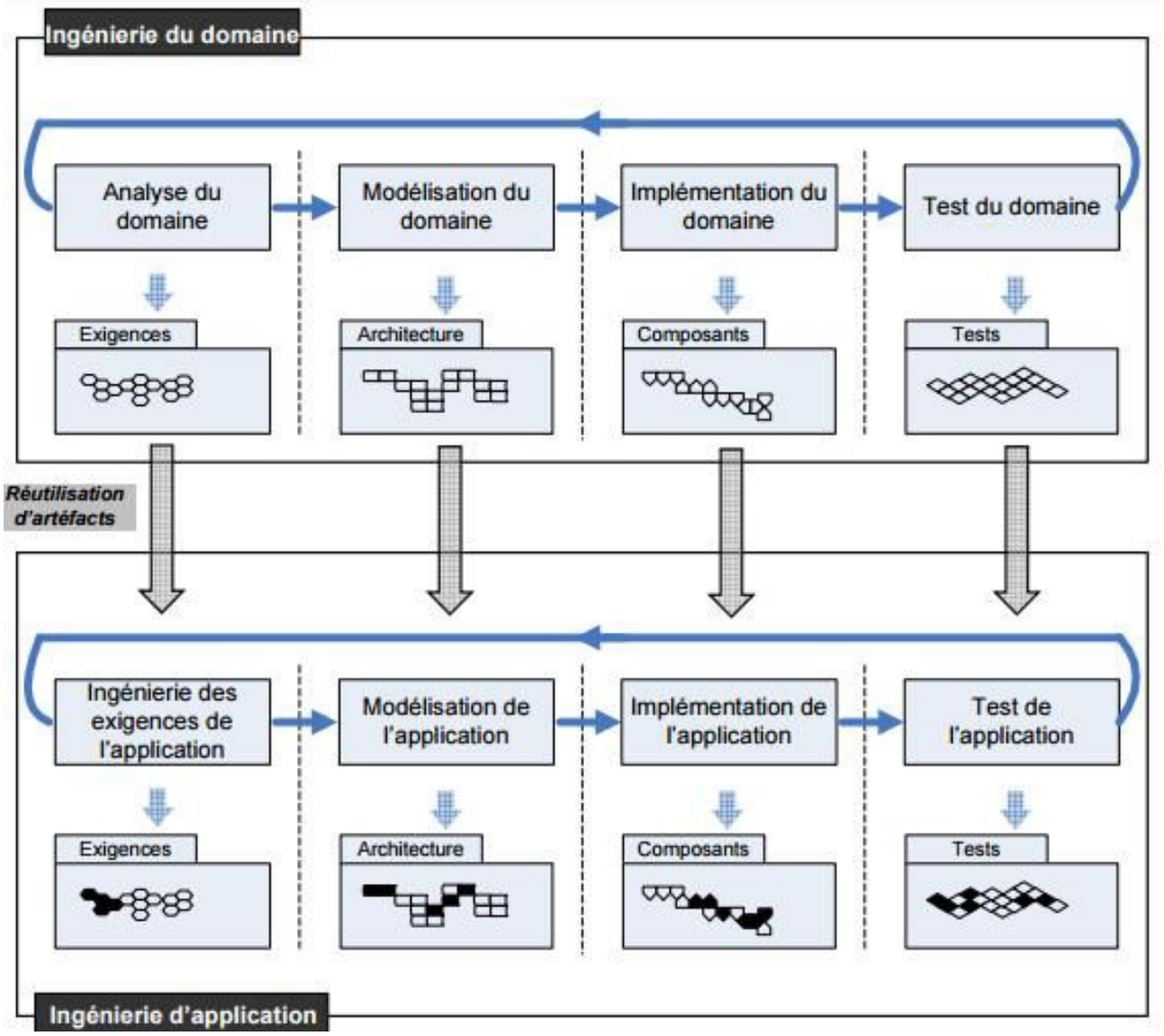


FIGURE 1.6 – Détail des activités du processus d'ingénierie de lignes de produits logiciels [1]

du monde réel qui contraignent les systèmes logiciels et à leurs liens avec les spécifications de ces systèmes".

Le concept de but apparaît dans standard IEEE 830 de documentation des exigences [IEEE1998].

En second lieu, l'activité va passer par la modélisation de l'application (voir Figure 1.6). L'ingénierie des exigences de l'application va désigner les besoins ou exigences spécifiques des produits souhaités, ce qui va influencer directement sur les artefacts à réutiliser lors de la construction de ce produit spécifique.

Quant à la modélisation de l'application, elle va permettre d'obtenir l'architecture du

produit à partir de l'architecture globale définie par la modélisation du domaine. Pour ce faire, les parties requises du modèle sont sélectionnées et incorporées afin de créer une configuration.

— **Dérivation du produit logiciel**

Le but de cette activité est de dériver le produit en fonction de la configuration établie au préalable et qui est fournie en sortie de l'analyse des besoins des utilisateurs. Il est important de souligner que l'utilisation d'approches d'implémentation annotatives, nécessite de dériver le code source avec un code composeur. Pour réaliser cette activité il faut passer d'une part par l'implémentation de l'application et d'autre part par le test de l'application.

L'implémentation de l'application : cette activité consiste à la réalisation du produit souhaité, en assemblant les composants correspondants aux parties sélectionnées dans l'étape précédente lors de la configuration.

Le test de l'application quant à lui, va permettre de valider et de vérifier que le produit obtenu est bien conforme aux exigences, à l'architecture et aux parties sélectionnées du modèle.

1.5 La modélisation des Lignes de Produits Logiciels

Dans cette section, nous nous intéressons à la modélisation des lignes de produits logiciels qui fait partie de la phase d'ingénierie du domaine présentée ci-dessus. Pour ce faire, il est primordial de passer par la modélisation des points communs et de la variabilité. Cette dernière pouvant être réalisée par le biais de différents modèles tels que *les modèles de décision* [37], *les modèles de variabilité orthogonale* [1], ou encore les *modèles de caractéristiques*. Ces derniers pouvant s'inscrire dans le cadre d'approches *annotatives* de la variabilité. Ces approches représentent la variabilité par le biais d'annotations au niveau des modèles. Par exemple, il est possible d'utiliser différents stéréotypes en UML afin d'annoter la variabilité [38]. Ces stéréotypes d'annotations de variabilité peuvent également être regroupés au sein de profil UML. Ziadi et al. [39] ont proposé un profil UML pour les lignes de produits. Leur travail introduit deux types de variabilité pour les diagrammes de classes qui sont modélisés au sein des stéréotypes : *l'Optionalité* :utilisant le stéréotype *optional*, et la *Variation* : utilisant les deux stéréotypes *variant* et *variation*. Quant au niveau comportemental relatif aux diagrammes de séquences, les auteurs ont introduit en plus de *l'optionalité* et de la *Variation*, la *Virtualité* : le stéréotype utilisé correspond à *virtual* et est utilisé pour étendre la méta-classe *interaction*. L'interaction étant elle-même considérée comme un point de variation.

Ces deux types de variabilité proposés (optionalité et virtualité) peuvent donc servir à modéliser les différentes versions d'un artefact au sein d'une ligne de produits logiciels, que la

variabilité soit temporelle ou spatiale [1]. En d'autres termes, que les différentes versions d'un produit soient valides à des moments différents (variabilité temporelle), ou qu'elles existent en même temps (variabilité spatiale), elles vont toutes les deux nécessiter une modélisation faisant intervenir l'optionalité et/ou la virtualité au sein des modèles.

Nous verrons plus tard dans ce chapitre les annotations utilisées au niveau de l'implémentation.

La Figure 1.7 représente un exemple d'annotations de variabilité sur l'optionalité qui est utilisée au niveau du diagramme de séquence (SD).

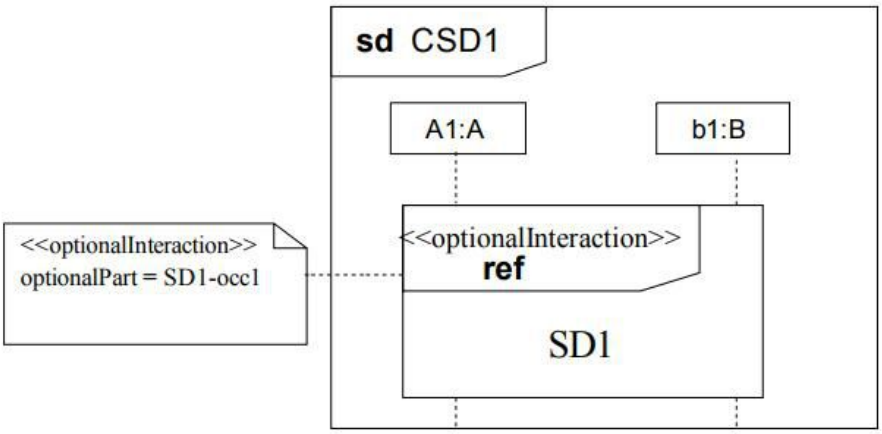


FIGURE 1.7 – Un exemple d'annotation de variabilité optionnelle pour un diagramme de séquence en UML2.0 [4]

Les points communs entre les différents membres d'une ligne de produits logiciels vont représenter en quelques sortes le coeur de la ligne de produits. Tandis que la variabilité va permettre de distinguer entre ces différents membres d'une même famille de logiciels. Il ressort de la littérature que la variabilité peut être définie comme faisant partie intégrante des artefacts de développement, ou encore être répartie à travers des modèles de variabilité distincts [1]. De nombreux travaux de recherche ont suggéré l'intégration de la variabilité dans les modèles de développement de logiciels traditionnels tels que les modèles de cas d'utilisation [40] [41] . D'autres travaux ont proposé de nouveaux modèles spécifiques aux lignes de produits afin de représenter les points mais également la variabilité [42] [43].

Dans cette section, nous nous intéressons à cette dernière catégorie de travaux, et allons donc donner quelques exemples de ces différentes approches de modélisation qui sont spécifiques aux lignes de produits logiciels tels que : le modèle de caractéristiques FM (Feature Model) [43], le modèle de variabilité orthogonale OVM (Orthogonal Variability Model) [1], ou encore le modèle

CVL (Commonality and Variability Language) [44].

Dans le cadre de cette thèse, nous nous intéressons particulièrement à l'approche des *modèles de caractéristiques* pour laquelle nous allons donner plus de détails, en comparaison avec les autres approches, car il s'agit du type de modèle que nous avons choisi durant notre travail de recherche. Quand aux raisons relatives à un tel choix, celles-ci seront présentées ultérieurement dans la suite du manuscrit (voir le chapitre 4).

1.5.1 Les modèles de variabilité orthogonale OVM

Avant de décrire le modèle OVM (Orthogonal Variability Model), il est primordial de comprendre pourquoi faut-il considérer une *variabilité orthogonale*. Celle-ci a pour principe de définir les différentes informations sur la variabilité qui est répartie au niveau de modèles séparés. Deux des raisons principales qui ont conduit les chercheurs à une telle vision sur la variabilité sont que d'une part, les concepts utilisés pour la définition de la variabilité, diffèrent entre la multitude des modèles du cycle de développement logiciel ; par conséquent, la variabilité définie au niveau des différents modèles n'est pas regroupée ni intégrée au sein d'une seule et unique image globale de la variabilité logicielle, alors que celle-ci s'avère essentielle pour l'ingénierie des lignes de produits logiciels. D'autres part, il est difficile d'identifier quelles sont les informations sur la variabilité des besoins qui influencent les informations sur la variabilité de la conception, du développement ou encore des tests des artefacts. C'est pour ces raisons que le modèle de variabilité orthogonale a été proposé entre autres [5] [1] [45] [46].

Les caractéristiques essentielles des modèles de variabilité orthogonale ressortent de la définition suivante : *" An orthogonal variability model is a model that defines the variability of a software product line. It relates the variability defined to other software development models such as feature models, use case models, design models, component models, and test models [1]."*

1.5.1.1 Propriétés du modèle OVM

Comme représenté au sein de la figure 1.8, le modèle OVM de variabilité orthogonale permet de relier la variabilité des modèles de base qui incluent les modèles des besoins, de conception, de composants, d'implémentation et de tests. Quant à la traçabilité entre le modèle OVM et les différents types de modèles de base, elle est établie par le biais des dépendances d'artefacts qui sont représentées par les lignes pointillées.

Le modèle OVM comporte deux éléments essentiels qui sont d'une part les *points de variation (VP)*, et d'autre part les *variants (V)*. Un point de variation permet de documenter les aspects variables d'une ligne de produits. Ces derniers sont choisis par le client ou l'ingénieur de la ligne de produits logiciels. Chaque variant est relié directement à un point de variation et doit donc documenter la façon avec laquelle le point de variation varie. La figure 1.9 regroupe les

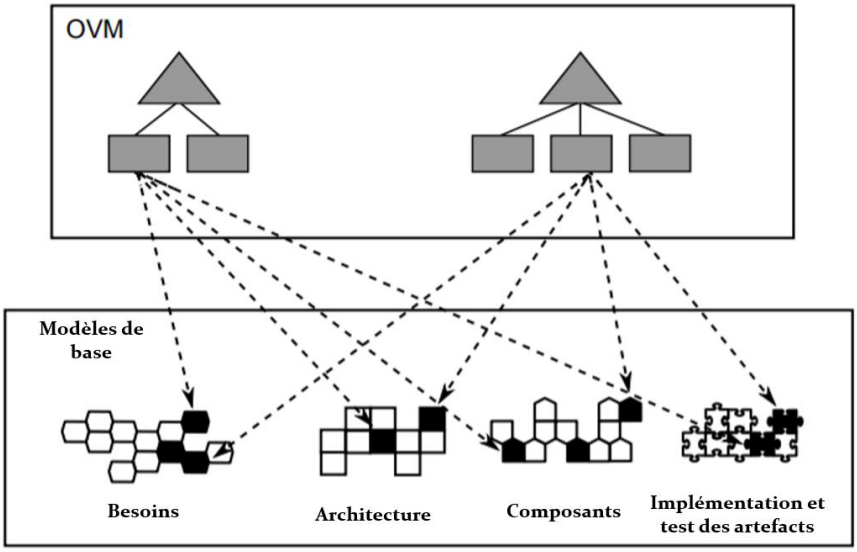


FIGURE 1.8 – Orthogonalité du modèle OVM [5]

différentes notations graphiques utilisées au sein d'un modèle de type OVM [1]. Il est à noter qu'il existe également d'autres notations graphiques utilisées pour ce même type de modèle [41], mais qui ne vont cependant pas avoir une sémantique différente des notations que nous avons présentées.

Il est possible de conclure à partir de la sémantique de ces notations, que les modèles OVM et le modèle de caractéristiques FM(Fetaure Model) sont similaires car les deux vont proposer une variabilité spécifiée par le biais de relations de type obligatoire (mandatory), optionnelle ou encore alternative. Par contre comme Roos-Frantz et al. [47] l'ont soulevé, il est néanmoins possible de distinguer les principales différences au niveau de leurs structures par exemple. Un modèle de caractéristiques FM se compose de caractéristiques organisées au sein d'un unique modèle sous forme d'arbre ayant une seule et unique caractéristique racine, tandis que le modèle OVM est composé de points de variations et de variants organisés au sein d'un seul ou plusieurs arbres à deux niveaux.

Nous présentons ci-après, le modèle de caractéristiques avec plus de détails. En outre, les modèles de caractéristiques permettent d'annoter les informations qualitatives au sein du modèle lui-même, alors que les modèles OVM réalisent cela de façon externe au modèle. Une autre différence soulevée est que la caractéristique racine étant toujours obligatoire, elle fera partie de tous les produits logiciels de la ligne de produits, alors que dans les modèles OVM les points de variations peuvent être optionnels. Ceci permet ainsi de dériver un produit logiciel à partir d'une configuration ne contenant aucun variant ou point de variation.

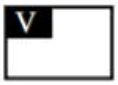
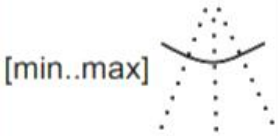
	**Point de variation Obligatoire (mandatory)**
	**Point de variation Optionnel (optional)**
	**Variant**
	Dépendance de variabilité « **Mandatory** »: Le variant doit être lié à chaque fois que son parent l'est
	Dépendance de variabilité « **Optional** »: Le variant peut être ou ne pas être lié à chaque fois que son parent l'est
	**Dépendance de variabilité** « **Alternative** »: La cardinalité détermine comment plusieurs variants du groupe peuvent être liés
	« **Requires** »: contrainte de dépendance
	« **Excludes** »: contrainte de dépendance

FIGURE 1.9 – Les notations du modèle OVM [1]

1.5.1.2 Exemple de modèle OVM

La figure 1.10 reprend un exemple de modèle de variabilité orthogonale du domaine de la sécurité d'une habitation. Ce modèle correspond à trois différents points de variations ("security package", "intrusion detection", et "door locks") identifiés et représentés séparément, tout en étant reliés entre eux par les différentes dépendances. Chacun des trois sous-modèles (points de variations) inclut deux à trois variants. Par exemple, la partie droite de la figure qui correspond au cas d'utilisation d'ouverture/déverrouillage d'une porte "door locks" illustre deux variants qui sont spécifiés par un choix alternatif à faire entre la possibilité d'ouverture/déverrouillage de la porte à l'aide d'un clavier "keypad", ou de l'empreinte digitale "fingerprint". Chaque variant est associé au cas d'utilisation correspondant par le biais d'une dépendance d'artefact.

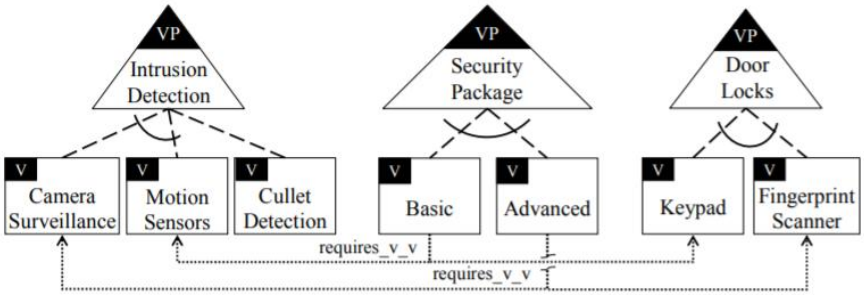


FIGURE 1.10 – Un exemple de modèle OVM associé au cas d'utilisation correspondant [1]

1.5.2 Les modèles de communalité et de variabilité obtenus avec CVL

L'approche de modélisation des lignes de produits en CVL (Commonality and Variability Language) [48] [49] [44] est un DSL pour la modèlisation de la variabilité, et représente ainsi un langage de modélisation qui permet de spécifier et de résoudre de la variabilité sur n'importe quel modèle dont le méta-modèle est conforme au MOF(Meta-Object Facility). Cette spécification OMG (Object Management Group) permet de définir un méta-modèle universel pour décrire les langages de modélisation.

En effet, l'ensemble des approches de modélisation issues de l'ingénierie des lignes de produits logiciels ont certes un même but, qui est celui de spécifier les points communs et la variabilité des lignes de produits logiciels, cependant, elles n'utilisent pas la même approche. C'est pour cette raison que l'OMG a pris l'initiative de développer une approche qui puisse être adoptée de façon générique en introduisant la modélisation CVL comme standard pour la variabilité.

1.5.2.1 Propriétés des modèles réalisés avec CVL

Afin de de déterminer les propriétés du CVL, nous nous sommes intéressés aux principaux concepts représentés dans la figure 1.11. Comme le montre cet extrait de méta-modèle, le modèle CVL peut contenir la spécification de la variabilité sous forme de *modèle VSpec* (Variability Specification), qui n'est autre qu'un arbre permettant de modéliser les caractéristiques de la ligne de produits logiciels.

Haugen et al. [48] précisent que l'approche CVL se concentre sur la spécification de la variabilité dans un modèle séparé du modèle de base de la ligne de produits. Il peut ainsi y avoir plusieurs modèles de variabilité rattachés au même modèle de la ligne de produits.

Ces spécifications de la variabilité contiennent d'une part les primitives exécutables qui forment le flux de l'exécution du CVL, et d'autre part les déclarations qui servent d'entrée à

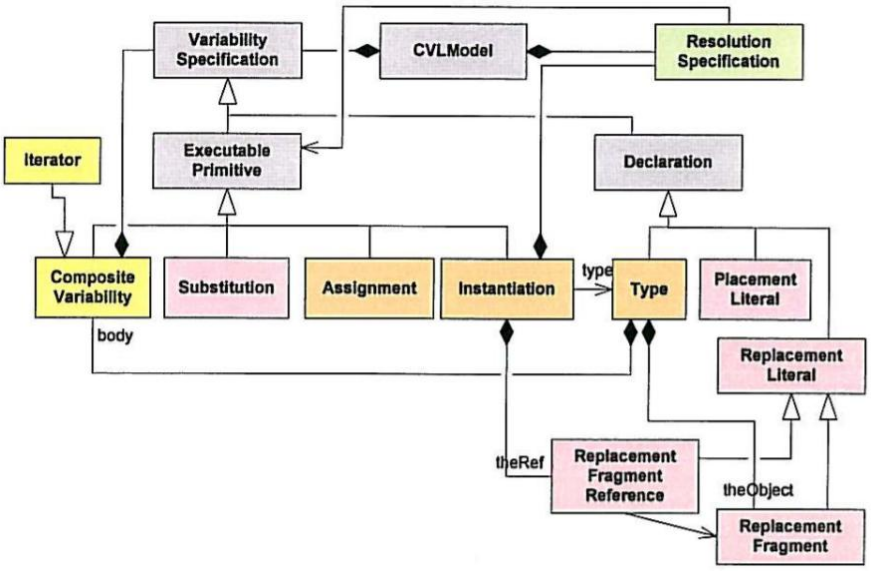


FIGURE 1.11 – Extrait du méta-modèle du CVL [6]

ces primitives. Afin de grouper les éléments, deux mécanismes peuvent être utilisés : *iterator* et *composite variability* que nous retrouvons dans la figure 1.12 afin de spécifier les différentes relations existantes au sein du modèle. Il est facile de constater que ce modèle est similaire au modèle de caractéristiques FM (Feature Model), et utilise donc la même sémantique comme présenté dans la figure 1.12, qui montre les associations entre les éléments des modèles en CVL et les symboles utilisés dans les modèles de caractéristiques. En effet, tel que Czarnecki et al. [50] l'ont rappelé pendant leur comparaison d'approches de modélisation de la variabilité, CVL utilise la syntaxe FODA (Feature Oriented Domain Analysis) [33] et par conséquent adopte la même hiérarchie que le modèle de caractéristiques FM. De plus, son utilisation permet de décomposer et d'organiser le modèle tout en appliquant les contraintes parent-enfant.

De plus, CVL peut contenir une ou plusieurs spécifications de résolution associées aux *modèles de résolution RM (Resolution Model)*. Ces *modèles de résolution RM*, et ceux de la réalisation VRM (Variability Realization Model) et de *l'abstraction de la variabilité VAM (Variability Abstraction Model)*, contiennent les informations nécessaires à la dérivation.Ils contiennent également les informations du modèle de base. Le modèle résolu (Resolved Model) est, par conséquent, sans variabilité. En d'autres termes, le modèle résolu est une instance du méta-modèle auquel le modèle de base est conforme.

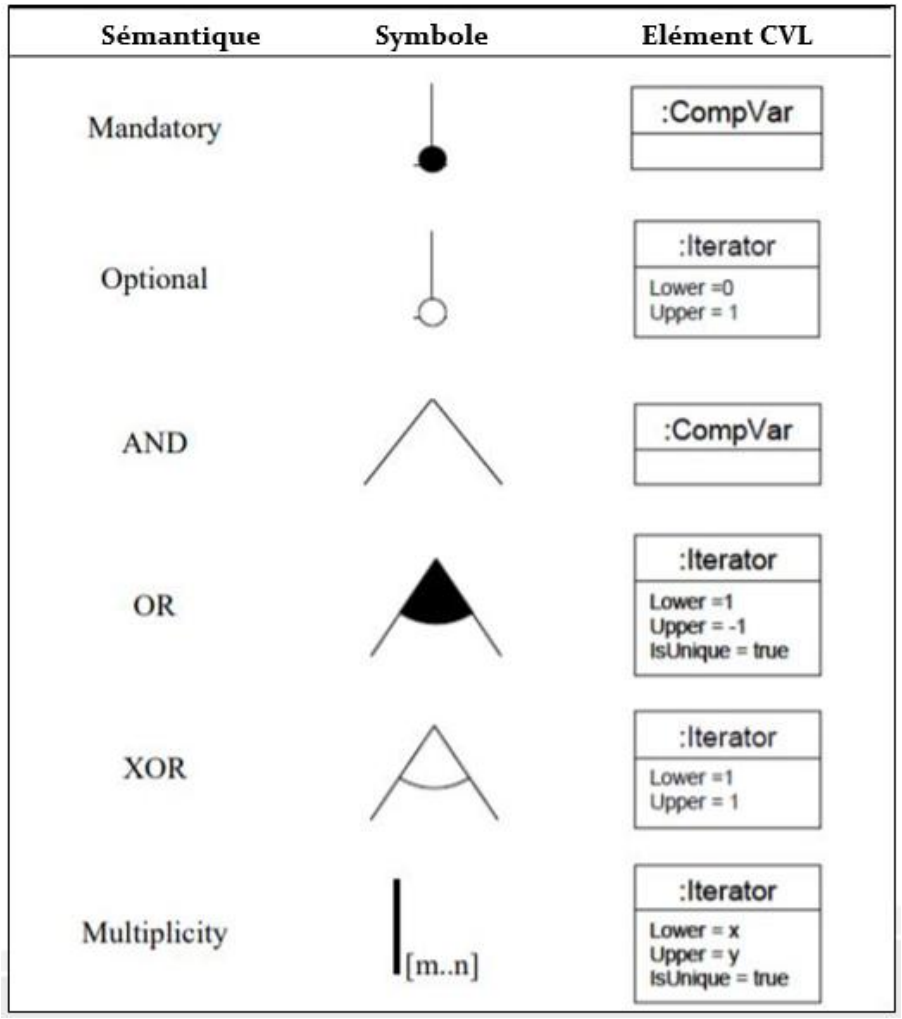


FIGURE 1.12 – Association entre le modèle de caractéristiques FM et les éléments d'un modèle réalisé avec CVL [6]

1.5.2.2 Exemple d'un modèle obtenu avec CVL

La figure 1.13 illustre un exemple de modèle d'abstraction de variabilité VAM (Variability Abstraction Model) obtenu en utilisant l'approche CVL. Ce modèle sert à spécifier la variabilité d'un site d'achat en ligne, tel que les choix CVL qui permettent de spécifier l'ensemble des caractéristiques du site comme celles du paiement, et du catalogue. Ces caractéristiques entre autres sont spécifiées au sein du modèle comme étant obligatoires, tandis que d'autres caractéristiques comme celle de la recherche sont optionnelles. Il est également possible de distinguer une multiplicité de groupes avec des valeurs de bornes supérieures différentes. Une borne de valeur 1 indique des choix enfants qui sont mutuellement exclusifs (élevé ou standard pour la sécurité), alors qu'une borne avec une valeur supérieure à 1 indique que plusieurs choix sont

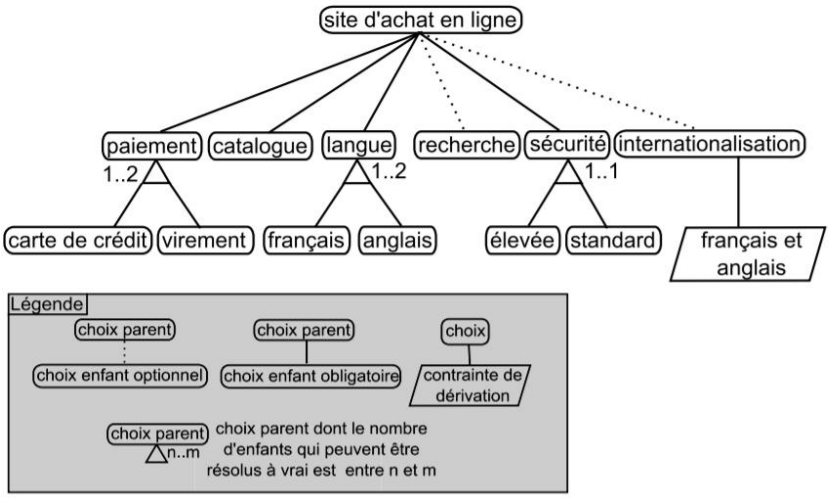


FIGURE 1.13 – Exemple d'un modèle VAM basé sur CVL pour la spécification de la variabilité d'un site d'achat en ligne [7]

possibles (carte de crédit et virement pour les paiements).

1.5.3 Les modèles de caractéristiques FM

L'approche des modèles des caractéristiques étant la plus répandue et la plus utilisée dans le domaine de modélisation de lignes de produits logiciels a commencé à être utilisée à travers l'approche FODA (Feature Oriented Domain Analysis) [33]. Cette approche a pour objectif de capturer les points communs et les points variables au niveau des besoins de la ligne. Pour ce faire, il faut utiliser des *Feature models* ou *Modèles de caractéristiques* avec leur représentation graphique *Feature Diagram* ou *Diagramme de caractéristiques*, car ces derniers représentent les modèles de variabilité les plus populaires [3]. Cependant, c'est par abus de langage que nous utilisons dans le présent manuscrit l'appellation Feature Model pour désigner la représentation graphique également. Avant de présenter cette approche de modélisation, il est important de préciser que certaines informations sont au préalable nécessaires à communiquer.

1.5.3.1 Propriétés des caractéristiques

Les caractéristiques (features) servent à représenter les abstractions du domaine. Pour ce faire, il faut préalablement récolter toutes les propriétés des caractéristiques représentatives du domaine. Les experts du domaine doivent entre autre recueillir cette liste d'informations extraite du livre de Sven et Apel [3], qui n'est pas complète mais qui permet d'avoir une vision des informations les plus importantes à récolter sur les caractéristiques avant de passer à la

représentation graphique :

- Description de la caractéristique (fonctionnalité) ainsi que de l'ensemble de ses besoins ;
- Description de sa relation avec les autres caractéristiques telle que la hiérarchie et l'ordre ;
- Dépendances externes liées par exemple aux ressources nécessaires à la caractéristique définissant sa fonctionnalité ;
- Estimation du coût de la mise en place d'une caractéristique ;
- Liste des clients potentiels qui seraient intéressés par la caractéristique en question ainsi que l'estimation du revenu correspondant ;
- La connaissance liée à la configuration, tel que les caractéristiques représentant des fonctionnalités existantes par défaut ('activated by default') ;
- L'insertion des contraintes entre les caractéristiques (Or, And, Xor, Not.etc) un exemple serait la contrainte suivante : "requires feature X and excludes feature Y" ;
- L'inclusion des types des différentes spécifications tels que les invariants, les pré/post conditions ;
- L'inclusion des attributs tels que les paramètres (textuels ou nombres) à utiliser ultérieurement lors de la personnalisation durant la génération de produits ;
- Potentielles interactions entre les caractéristiques.

La liste des informations à collectionner peut se limiter à un sous ensemble des caractéristiques relatives au domaine à analyser. Dans le cadre de cette thèse, seules les contraintes logiques permettant d'obtenir des configurations valides, les connaissances sur les features ainsi que leur implémentation nous intéressent.

1.5.3.2 Représentation graphique du modèle de caractéristiques (Feature Model)

Le Feature Model décrit les relations (parent-enfant) dans un arbre hiérarchique. Les features expriment les points communs (similitudes) et les différences (variabilités) entre les produits de la ligne.

Une des définitions formelles du diagramme des caractéristiques (feature diagram) stipule que :

" Un diagramme de caractéristiques est une représentation graphique d'un modèle de caractéristiques sous forme d'un arbre articulé autour d'un ensemble F de caractéristiques (Features)". Chaque branche de l'arbre est définie par exactement une seule contrainte, et ce par le biais de déclaration d'une des contraintes relativement aux caractéristiques pouvant être de type obligatoire (mandatory), optionelle (optional), alternative, ou bien or [3].

$root(f) \equiv f$

$mandatory(p,f) \equiv f \leftrightarrow p$

$optional(p,f) \equiv f \Rightarrow p$

$alternative(p,f_1,..f_n) \equiv ((f_1 \vee ... \vee f_n) \Leftrightarrow p) \wedge \bigwedge_{i<j} \neg (f_i \wedge f_j)$

$or(p,f_1,..f_n) \equiv (f_1 \vee ... \vee f_n) \leftrightarrow p$

De plus, un ensemble d'autres contraintes cross-tree peuvent être définies. Les formules propositionnelles correspondantes aux contraintes ainsi que les contraintes cross-tree sont regroupées au sein d'une seule formule propositionnelle représentant la sémantique de l'ensemble du diagramme de caractéristiques."

Des symboles graphiques spécifiques permettent de distinguer les caractéristiques obligatoires de celles qui sont optionnelles ou alternatives. Les relations entre les caractéristiques sont elles aussi représentées par des symboles graphiques. L'ensemble des relations inclut le "OR" et "XOR" par exemple. D'autres contraintes dites transversales de types inclusion ou exclusion peuvent être considérée également. Les contraintes d'une ligne de produits quant à elles définissent les produits logiciels valides qui peuvent être obtenus. La notion de configuration est utilisée pour représenter un produit d'une LdP. Elle consiste en une sélection de caractéristiques qui sont compatibles avec les contraintes établies.

La Figure 1.14 regroupe les différentes notations graphiques utilisées pour représenter les relations qu'il est possible d'utiliser. Les caractéristiques *Mandatory* et *Optional* se distinguent par un cercle de différentes couleurs un au niveau du nœud fils "B", tandis que le nœud parent est représenté par "A".

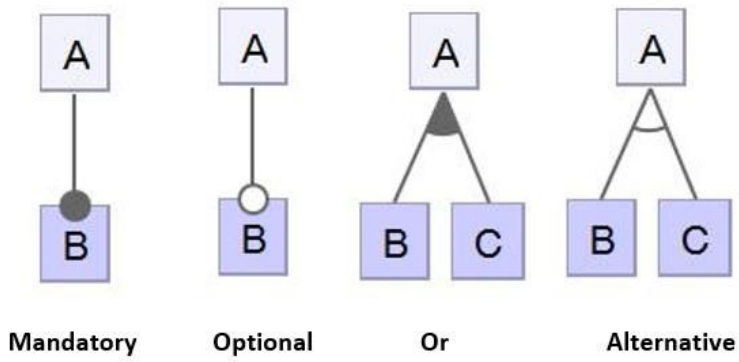


FIGURE 1.14 – Les notations graphiques utilisées dans les modèles de caractéristiques

Dans cette même figure, nous retrouvons des liens entre la caractéristique "A" avec ses deux fils "B" et "C". Ces deux types de relations représentées par le "Or" et "Alternative" sont utilisés avec au moins deux fils. La notation graphique représentative de la relation "Or" signifie le choix d'une ou plusieurs caractéristiques de l'ensemble de la liste des caractéristiques enfants. Ceci représente une disjonction inclusive [3]. La notation graphique de la relation "alternative" quant à elle, signifie un choix possible d'une seule caractéristique fille parmi l'ensemble des enfants de la caractéristique racine. Cette relation est traduite par une disjonction exclusive. Il est par exemple possible de considérer les différentes implémentations réalisables

de la même caractéristique, ou encore les différentes plateformes techniques tel que le système d'exploitation [3].

La figure 1.15 quant à elle représente un ensemble de contraintes logiques permettant comme souligné au préalable, d'éliminer des configurations invalides. Dans cet exemple, la première contrainte spécifie que si la caractéristique *"Credit card"* fait partie d'une configuration, cela implique que la caractéristique "High" relative au degré de la sécurité, sera elle aussi sélectionnée et fera donc partie de cette même configuration.

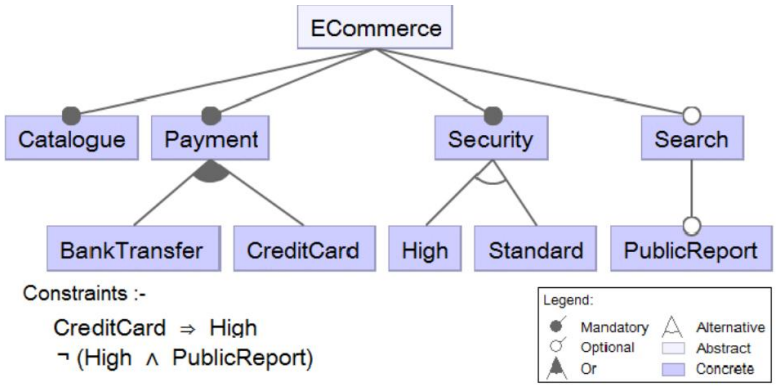


FIGURE 1.15 – Un exemple de modèle de caractéristiques avec contraintes [8]

1.5.3.3 Exemple de modèle de caractéristiques

Dans cet exemple, nous allons utiliser l'ensemble des notations graphiques présentées ci-dessus. Le modèle de caractéristiques concernant une LdP basique d'une famille de produits de type voiture (voir Figure 1.16). La caractéristique "Car" racine représente la ligne de produits. Cette caractéristique est décomposée en sous-caractéristiques (sub-features) qui sont elles aussi décomposées jusqu'à atteindre les caractéristiques feuilles. Dans cet exemple, la caractéristique "Engine" est obligatoire tandis que la caractéristique "AirConditioning" est optionnelle. Comme stipulé dans l'exemple, la relation *Or* représentée entre les deux caractéristiques concernées spécifie que la caractéristique "Engine" peut être réalisée via un moteur "Gasoline" ou "Electric" ou bien encore les deux ensemble. La "Transmission" par contre peut être "Manual", ou bien "Automatic" mais pas les deux ensemble.

1.5.3.4 Les modèles de caractéristiques multi-niveaux

Les approches de modélisation multi-niveaux permettent entre autres d'améliorer la réutilisation logicielle. Les différents niveaux considérés dépendent de l'objectif à atteindre. Par exemple, dans [51], les auteurs proposent un modèle multi-niveaux pour la conception des

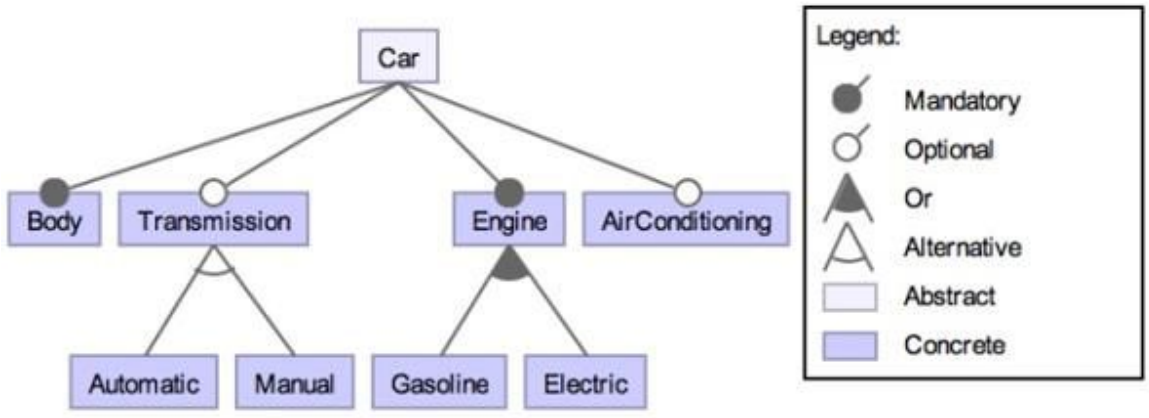


FIGURE 1.16 – Un exemple de modèle de caractéristiques d'une ligne de produits de type voiture

DSMLs (Domain-Specific Modeling Languages) visant à améliorer les gains de productivité en minimisant la redondance conceptuelle à tous les niveaux d'abstraction.

Dans [52], le modèle multi-niveaux proposé est appliqué au niveau des besoins incluant ainsi l'ensemble des besoins des experts du domaine, des ingénieurs de la ligne de produits, des produits, et des gestionnaires de projet. Ce modèle multi-niveau permet d'assembler tous les besoins allant des besoins fondamentaux aux besoins alternatifs ; et aussi de recombiner les besoins requis pour construire un nouveau modèle de besoins plus facilement, et désassembler les besoins alternatifs ou encore ajouter de nouveaux besoins lors de la reconfiguration d'un produit.

Ce même principe peut être appliqué pour la modélisation des lignes de produits logiciels. Ainsi, une ligne de produits logiciels est dite multiple, s'il est possible de considérer plusieurs niveaux de variabilité. Par conséquent, les caractéristiques sont déterminées en fonction des différents niveaux d'abstractions auxquels elles appartiennent. De ce fait, l'instanciation de la variabilité peut se faire de façon progressive, et les configurations associées seront également reliées aux différents niveaux. Par exemple Czarnecki et al. [9] proposent des choix de configurations disponibles à chaque niveau à travers des modèles de caractéristiques séparés.

La figure 1.17 illustre le fait que la configuration à plusieurs niveaux n'est autre qu'une forme de configuration par étapes où les choix disponibles pour chaque étape sont représentés par des modèles de caractéristiques distincts. Dans cet exemple, nous retrouvons trois niveaux tel que chaque modèle de caractéristiques de niveau n représente les choix possibles de configurations disponibles à l'étape n. De cette façon, les choix de configurations disponibles à l'étape 0 sont représentés par le modèle de caractéristiques (fonctionnalités) de niveau 0. De même, les choix de configurations disponibles à l'étape 1 sont représentés par le modèle de caractéristiques

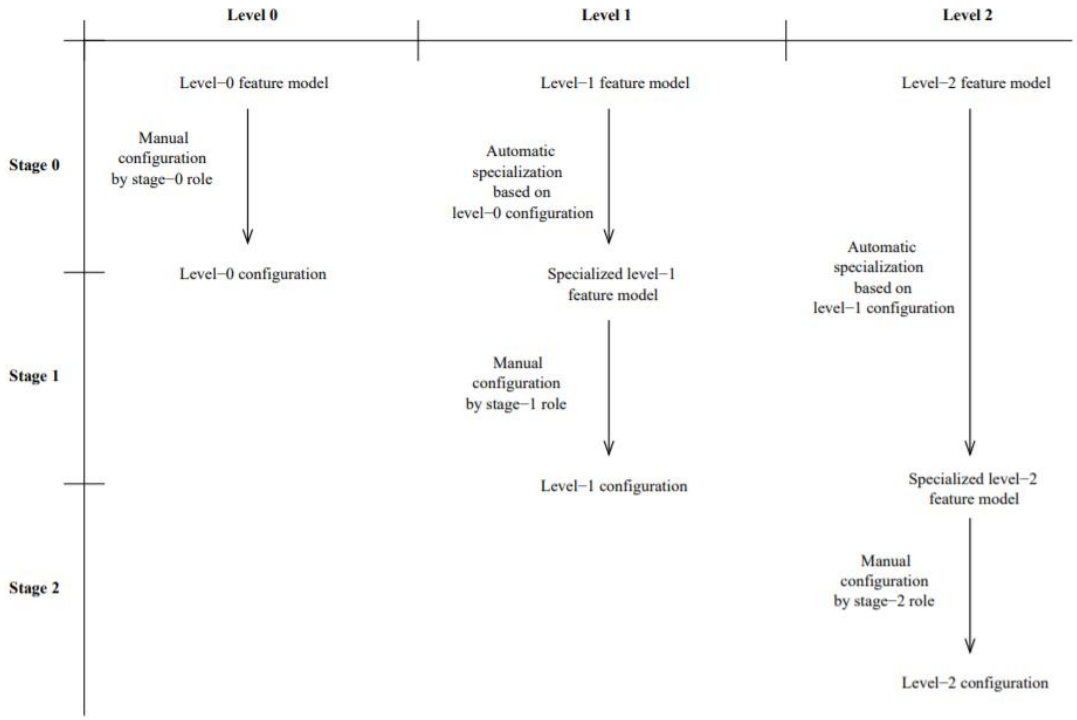


FIGURE 1.17 – Modèles de caractéristiques et de configurations à trois niveaux [9]

(fonctionnalités spécialisées) de niveau 1 disponible au début de l'étape 1. De façon générale, un modèle de caractéristiques de niveau n est obtenu par spécialisation/raffinement du modèle de caractéristiques de niveau *n-1*.

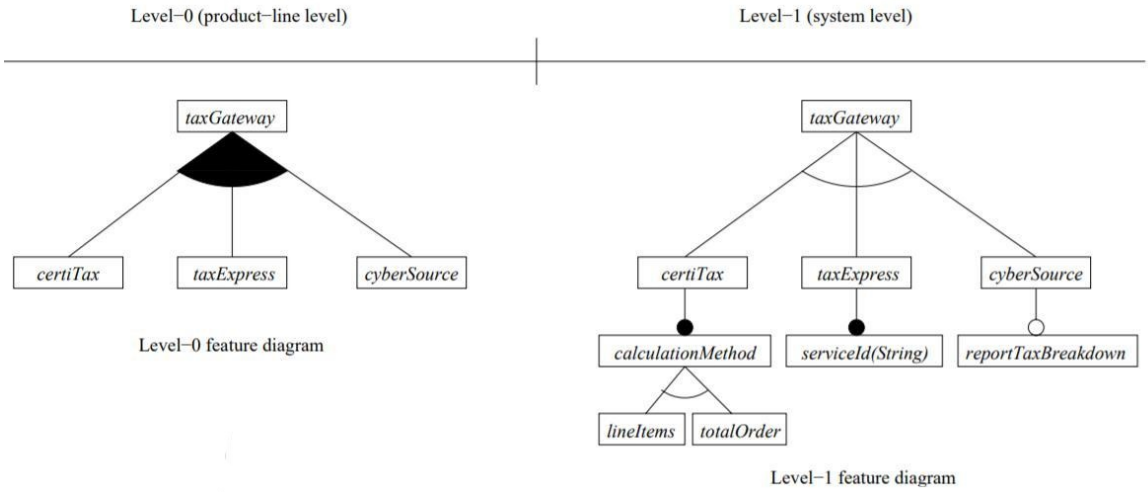


FIGURE 1.18 – Modèles de caractéristiques de passerelle fiscale à deux niveaux [9]

Dans la figure 1.18 par exemple, deux modèles de caractéristiques de passerelles fiscales sont associés à deux niveaux différents (niveaux 0 et 1). Ces passerelles fiscales sont représentées sous forme de services électroniques tels que les services web. Ces services sont offerts par des tiers et peuvent donc être utilisés par des boutiques électroniques afin de calculer les taxes pour les produits vendus. Le premier niveau du modèle de caractéristiques de la ligne de produits permet de supporter plusieurs passerelles qui pourraient être disponibles (product line level). Le second niveau de modèle de caractéristiques est associé directement au niveau du système (system level).

1.6 L'implémentation des lignes de produits logiciels

Une fois la modélisation de la ligne de produits réalisée, il est nécessaire de passer à la phase d'implémentation (voir Figure 1.5).

Dans cette section, nous allons présenter les aspects les plus importants liés à l'implémentation des LdP. Nous allons, ainsi présenter les différentes approches existantes pour représenter au niveau du code les aspects liés à la variabilité logicielle.

Nous considérons dans cette section les trois approches existantes. En premier les approches *annotatives*, en second les approches de *compositions* [3]. Pour finir nous considérons également un autre type d'approches combinant les deux approches précédentes dites approches *combinatoires* [53].

1.6.1 Les approches d'annotations

Afin de permettre l'implémentation de la variabilité, les approches d'annotations associent les artefacts d'implémentation des caractéristiques aux fonctionnalités en les annotant.

Les variantes individuelles peuvent être générées en supprimant des parties représentatives des caractéristiques non souhaitées. Ce type d'approches est très utilisé en pratique. Elles sont nativement supportées par beaucoup d'environnements de programmation. Cependant ces approches sont critiquées à cause de leur manque de modularité entre autres [54] [3].

Une définition extraite du livre de Apel et al. [3] est la suivante :

"Les approches d'annotations annotent un code commun de base, tel que le code qui appartient à une certaine caractéristique sera marqué. Lors de la dérivation de produit, tout le code appartenant aux caractéristiques non sélectionnées lors de la configuration, ou à des combinaisons invalides de certaines caractéristiques, tous les codes correspondants seront retirés (au moment de la compilation) ou encore ils seront ignorés (au runtime) et ce afin de former le produit final". Il est important de souligner que le code de base représente le code commun présent au sein de l'ensemble des variantes de la ligne de produits.

La figure 1.19 représente un exemple d'implémentation basée sur l'approche annotative. Les bouts de code relatifs à chacune des caractéristiques sont représentés au niveau du même code source et sont distingués grâce aux annotations représentées par les différentes couleurs correspondantes.

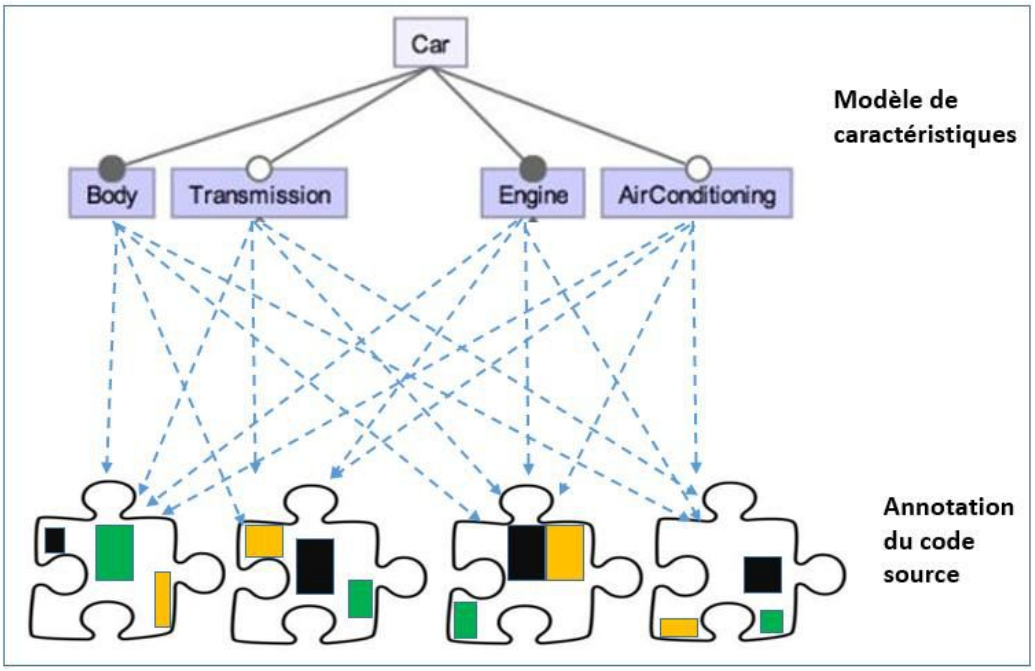


FIGURE 1.19 – Illustration d'approche annotative pour l'implémentation d'une ligne de produits logiciels

1.6.2 Les approches compositionnelles

Les approches compositionnelles se basent sur le fait que les caractéristiques variables d'un système soient mappées à des modules dédiés, et représentées sous formes d'unités séparées. L'avantage principal est que cette modularisation améliore la traçabilité des caractéristiques et la séparation des différentes préoccupations système.

La définition extraite du livre de Apel et al. [3] est la suivante :

"Les approches compositionnelles implémentent les caractéristiques sous la forme d'unités composables, idéalement au nombre d'une unité par caractéristique. Lors de la dérivation, toutes les unités de toutes les caractéristiques sélectionnées ainsi que celles des combinaisons valides de caractéristiques sont composées afin de former le produit final ".

La figure 1.20 illustre les approches compositionnelles par la représentation des différentes unités de code source relatives à chacune des caractéristiques permettant de développer une

ligne de produits logiciels.

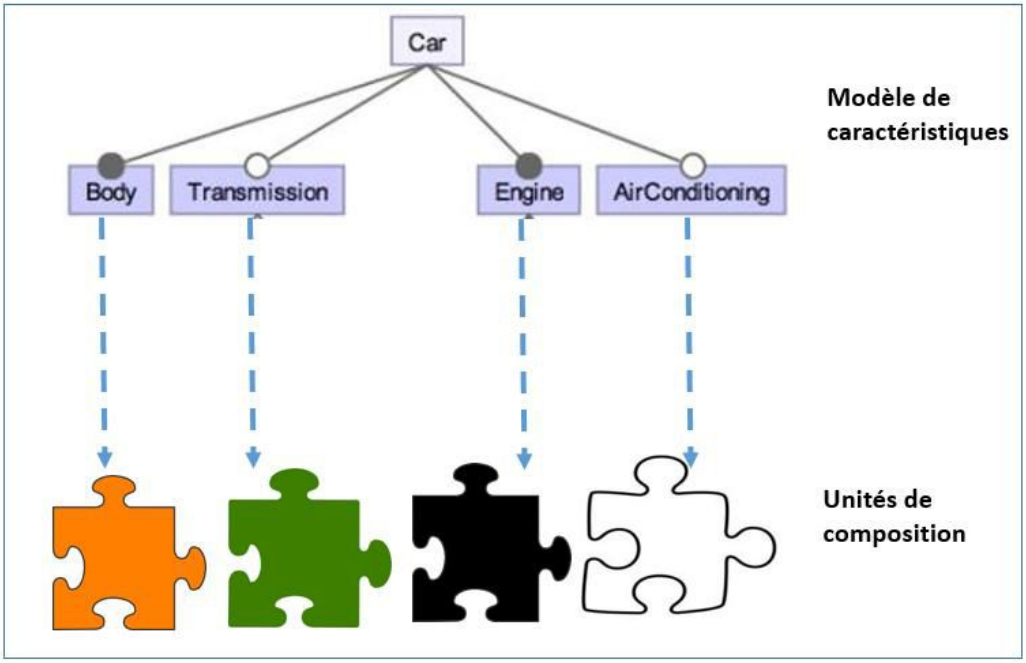


FIGURE 1.20 – Illustration d'approche compositionnelle pour l'implémentation d'une ligne de produits logiciels

Ce type d'approche associe donc le code appartenant à une caractéristique ou encore à une combinaison de caractéristiques au niveau d'un fichier source dédié, d'un conteneur ou même d'un module relatif. Les caractéristiques de la ligne de produits sont ainsi implémentées sous formes de fragments séparés. Un exemple classique est un framework pouvant être étendu par des plugins (de façon idéale un plugin par caractéristique). Différents produits peuvent ainsi être générés en intégrant les caractéristiques adéquates lors de la configuration. Lorsque nous utilisons ce type d'approches, il est nécessaire d'utiliser un outil permettant de composer les différents bouts de code désigné en anglais par "code composer", tel que Feature House [32]. Ce dernier supportant plusieurs langages de programmation et ayant été intégré au sein de l'outil FeatureIDE [1], permet également de spécifier les modèles de caractéristiques et de définir une configuration.

La figure 1.21 représente un exemple utilisant une approche compositionnelle. Dans cet exemple, nous utilisons FeatureIDE pour la spécification du modèle de caractéristiques ainsi que FeatureHouse comme compositeur de code. Comme illustré, la caractéristique Agent_Basic est implémentée par la classe C1. Cette dernière doit faire référence aux méthodes obligatoires

1. https ://marketplace.eclipse.org/content/featureide

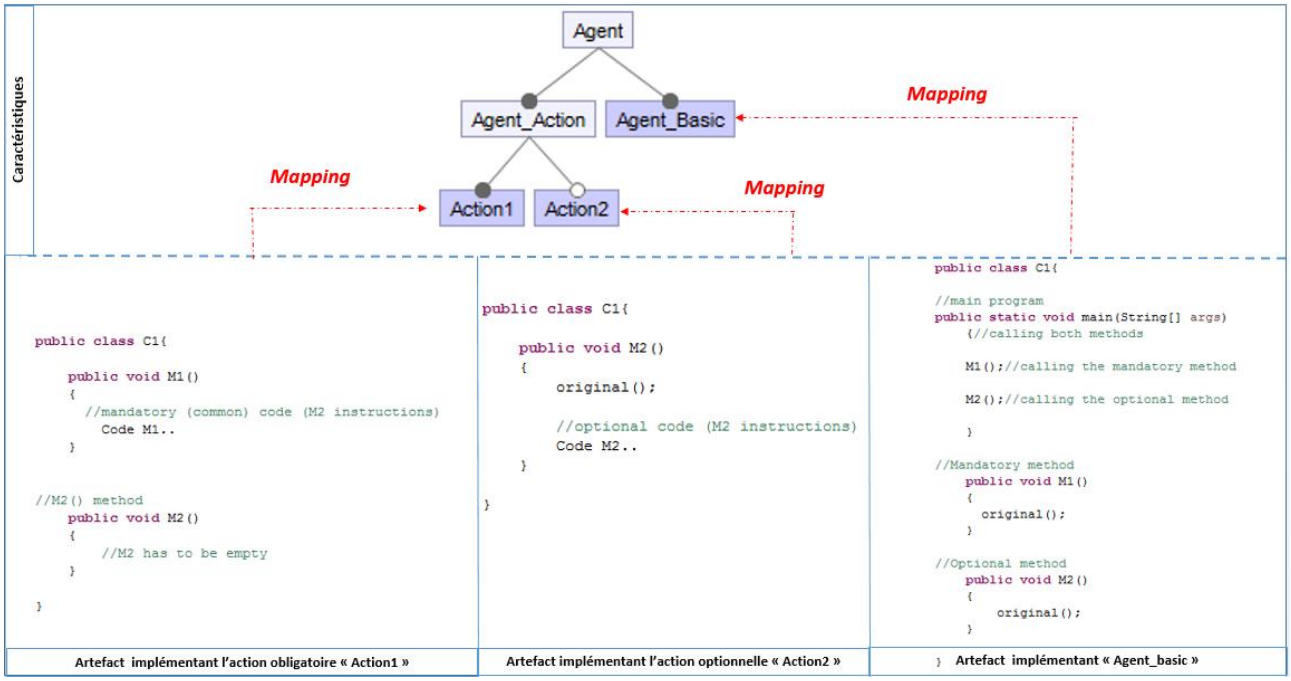


FIGURE 1.21 – Exemple d'implémentation par approche compositionnelle

et optionnelles. En effet, les méthodes M1() et M2() implémentant respectivement les actions Action1 et Action2, raffinent la classe C1.

Le code généré dépendra de la configuration choisie. Les deux figures 1.22 et 1.23 représentent respectivement les deux configurations possibles en (a). Le diagramme de collaboration est représenté en (b), il permet de visualiser le croisement entre les caractéristiques et le code source à dériver. Le code obtenu est représenté en (c).

La configuration 1 (voir 1.22) permet l'exclusion de la caractéristique optionnelle Action2. Le code de M2() ne sera pas compris dans la classe C1 générée. Quant à la configuration 2 (voir figure 1.23), elle permet au contraire d'inclure la caractéristique optionnelle dans la variante générée. FeatureHouse assemble ainsi les bouts de code représentés en (b) afin d'obtenir (c) sur la base des caractéristiques sélectionnées en (a).

1.6.3 Les approches combinatoires

Bien que les approches *compositionnelles* soient plus utilisées que les approches *annotatives*, Apel et al. ont soulevé le fait qu'il était possible de combiner ces deux types d'approches dans le but d'empêcher les développeurs de suivre une seule approche avec des coûts incertains et des risques. L'approche dite *combinatoire* se base sur un processus de migration au moment adéquat pour le passage des annotations aux compositions. Avec cette intégration, le développeur peut

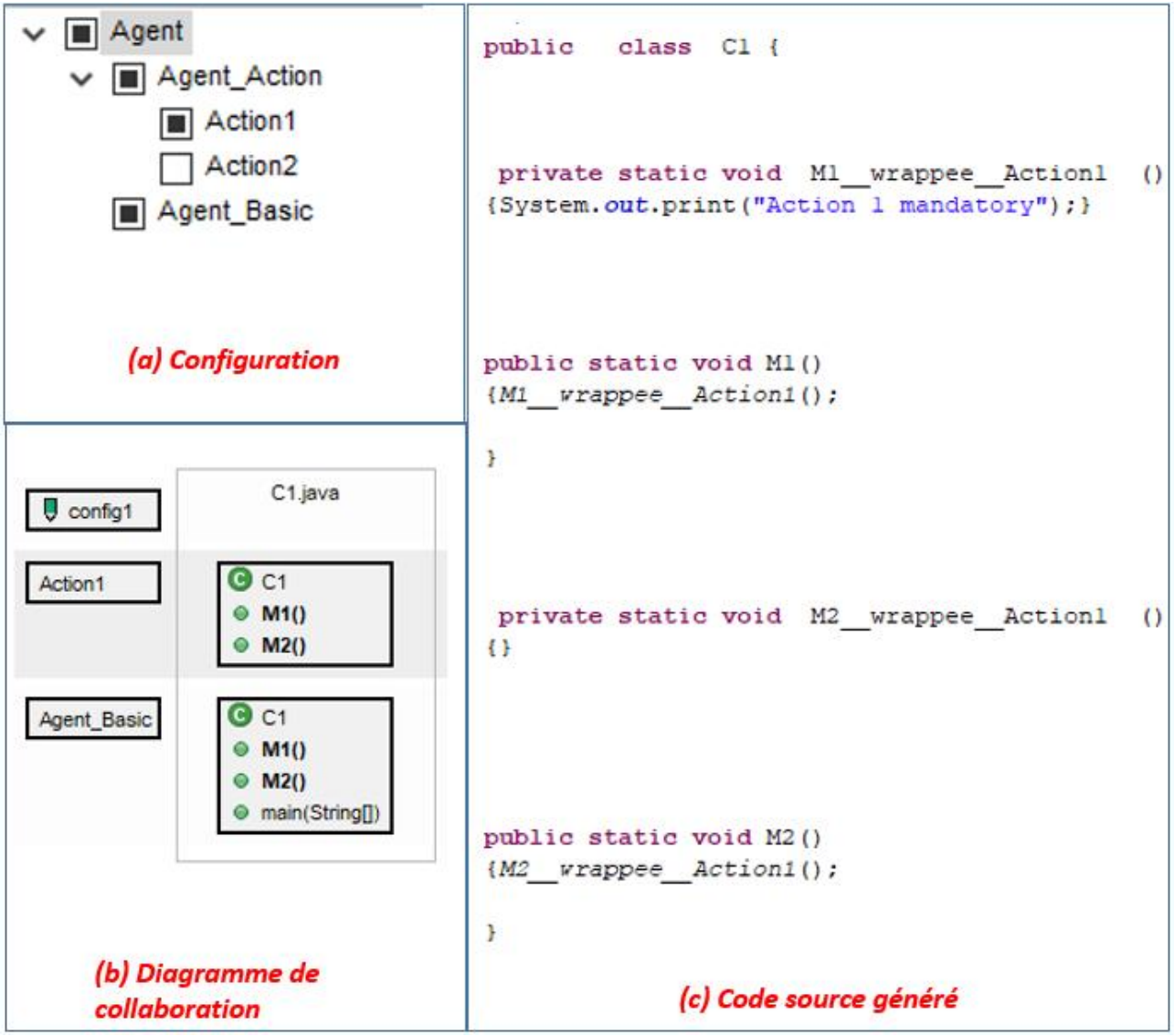


FIGURE 1.22 – Exemple de configuration minimale (Action 1) avec génération de code par FeatureHouse

choisir séparément quel mécanisme de mise en œuvre utiliser pour chaque type de problème ; ce qui facilite l'adoption des LdP. Lors de l'adoption des technologies LdP, il est possible d'utiliser d'abord une annotation légère et de se réfracter graduellement vers la composition autant que possible.

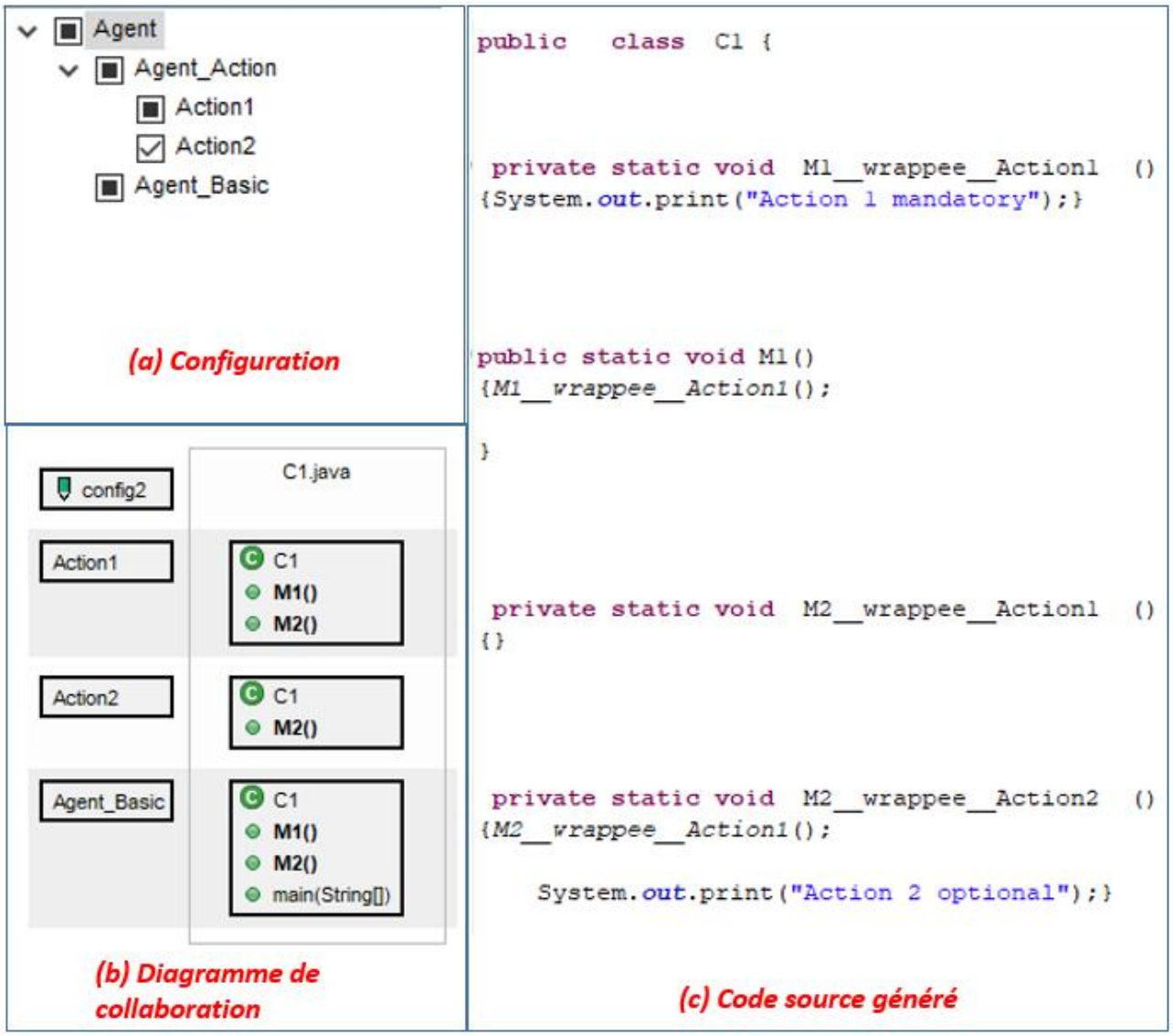


FIGURE 1.23 – Exemple de configuration (Action 1 et Action 2) avec génération de code par FeatureHouse

1.7 Conclusion

Dans ce chapitre, nous avons abordé les grandes lignes et principes d'ingénierie de lignes de produits logiciels, ces concepts sont nécessaires à la compréhension de l'approche MAS-PL que nous proposons dans la cadre de cette thèse. Nous avons notamment présenté les concepts de similitudes, de variabilité et de caractéristiques, en ce qu'ils représentent les concepts les plus importants autour desquels s'articule l'ingénierie de LdP. Avant de présenter le processus général des LdP servant de référentiel dans ce domaine, nous avons présenté les différentes approches

permettant le passage d'une approche de développement classique de produits (un à un) vers une approche de LdP. Les approches basées sur le processus général sont dites proactives, et suivent un processus composé de deux phases d'ingénierie : *Ingénierie du domaine* et *Ingénierie d'application* : la première phase permet un développement pour la réutilisation, et la seconde un développement par la réutilisation. Plus précisément concernant la première phase, nous avons présenté quelques approches de modélisation dont celle des modèles de caractéristiques FM avec un cas particulier qui est celui des modèles de caractéristiques multi-niveaux.

Cette approche de modèles de caractéristiques est considérée comme le moyen le plus utilisé pour la représentation graphique des similitudes et des variabilités entre les produits logiciels de la ligne. Concernant la seconde phase d'ingénierie, nous avons décrit les différents types d'implémentation des similitudes et de la variabilité, telles que les approches compositionnelles qui nécessitent l'utilisation d'un code composeur contrairement aux approches annotatives représentant la variabilité au niveau du code source par l'intermédiaire d'annotations, mais qui peuvent parfois s'avérer lourdes si elles sont réparties sur plusieurs niveaux. L'avantage principal de l'ingénierie de LdP est qu'elle permet une réutilisation visant à la conception et l'implémentation de toute une gamme de produits logiciels au lieu des techniques usuelles de réutilisation qui ciblent un seul produit logiciel à la fois. Le second avantage de ce type d'approches est de formaliser la spécification et l'implémentation des similitudes et de la variabilité des membres d'une même ligne de produits logiciels. Contrairement à l'ingénierie de LdP, l'ingénierie multi-agents ne propose pas des approches de réutilisation visant des familles de systèmes. Certes les patterns multi-agents de réutilisation tels que les "design patterns", permettent d'améliorer la réutilisation multi-agents à différents niveaux. Cependant, bien qu'ils puissent servir à concevoir plusieurs systèmes, ils ne permettent pas de spécifier les points communs et variables par rapport aux versions ultérieures. De plus, la variabilité peut être prise en compte dans les patterns de réutilisation, cependant elle décrit la variabilité au sein d'un même produit. Pour répondre donc au besoin de réutilisation au sein d'une famille ou de ligne de produits multi-agents, l'introduction des techniques d'ingénierie de LdP au sein de l'ingénierie multi-agents est une solution intéressante de laquelle a émergé une nouvelle branche d'ingénierie nommée MAS-PL (Multi-agent systems product lines) que nous allons présenter et analyser dans la partie de l'état de l'art du présent manuscrit. Mais tout d'abord, le prochain chapitre est consacré aux principaux concepts et caractéristiques des SMA qui nous ont servi de base à la réalisation d'une partie très importante de notre travail de recherche et qui est celle de l'analyse du domaine multi-agents afin d'en déduire les éléments réutilisables (voir chapitre 5).

Chapitre 2

Les SMA (Systèmes multi-agents)

2.1 Introduction

Les systèmes multi-agents ou SMA s'avèrent être une solution idéale pour la modélisation de systèmes complexes. C'est pourquoi l'ingénierie des SMA propose un ensemble de méthodologies, modèles et outils dont l'objectif principal est de faciliter la conception, et l'implémentation des systèmes complexes, tout en assistant les concepteurs et développeurs durant les phases du cycle de vie du système multi-agents.

Par conséquent, l'ingénierie des SMA doit tenir compte des différents aspects et caractéristiques des systèmes complexes, offrant entre autres une autonomie de l'entité principale du système représentée par l'agent. Ou encore la possibilité d'interaction entre ces agents. D'autres éléments tels que l'organisation du système, ou encore l'environnement d'agents sont à prendre en compte également.

Dans ce chapitre, nous allons présenter l'ensemble des définitions, et concepts les plus pertinents dans le cadre de notre travail de recherche, et nécessaires à la compréhension de notre contribution, car ces éléments ont servi de base et de référentiel à notre analyse du domaine multi-agents (voir le chapitre 5).

Nous allons présenter les différentes caractéristiques d'un SMA, en nous basant sur une décomposition du système proposée par l'approche VOYELLES [28, 29] que nous avons choisie d'utiliser pour les raisons qui seront mentionnées ultérieurement (voir chapitres 4 et 5).

Nous décrivons donc dans ce qui suit un SMA par le biais des différentes voyelles : A (Agent), E (Environnement), I (Interaction), et O (Organisation), en donnant les différentes composants ainsi que les visions de ces concepts et leurs sémantiques à travers les modèles et méthodologies qui les supportent.

Pour le concept d'agent nous présenterons par exemple l'un des modèles d'architectures délibératives les plus connues et qui est celle du BDI (Belief-Desire-Intention), ainsi que les

différents modes de perception (active et passive). Pour l'environnement nous présenterons de façon synthétique les éléments essentiels issus des différentes visions des modèles d'environnements d'agents existants tel que le modèle *A&A* Agent and artefact proposé par Ricci et al. [55]. Tandis que pour les interactions nous présenterons les concepts relatifs aux interactions directes et indirectes. Pour finir, nous aborderons des visions différentes des modèles d'organisation proposés pour les SMA allant de la vision la plus minimaliste proposée par le modèle organisationnel AGR [20], à d'autres visions tenant compte d'avantages d'éléments tels que Moise+ [21] qui intègre une vue dite déontique/normative de l'organisation des SMA.

2.2 Description générale de l'approche VOYELLES (A.E.I.O)

Le paradigme VOYELLES a été proposé par Demazeau et al. [28] [29] afin de décrire les composants principaux d'un système multi-agents.

Cette approche considère que l'analyse, la conception, l'implémentation et le déploiement d'un SMA, s'articulent autour de quatre aspects fondamentaux, et par conséquent un SMA peut être décrit par les quatre voyelles suivantes :

- *Voyelle "A" pour "Agent" :* Représente les architectures internes de l'agent.
- *Voyelle "E" pour "Environnement" :* Représente le milieu dans lequel évoluent les agents.
- *Voyelle "I" pour "Interaction" :* Représente les moyens par lesquels les agents interagissent.
- *Voyelle "O" pour "Organisation" :* Représente les moyens utilisés pour structurer l'ensemble des entités du système, tout en tenant compte des relations sociales pouvant exister entre les éléments du SMA.

Après avoir donné une description générale de l'approche voyelles, nous allons voir de plus près les aspects les plus importants de chacune de ces voyelles (A.E.I.O).

2.3 Le concept d'Agent

Un agent est une entité autonome physique ou virtuelle capable de percevoir son environnement grâce à des capteurs (sensors), et capable d'agir sur cet environnement par le biais d'effecteurs (effectors) comme représenté dans la Figure 2.1. Il existe différents modèles d'agents en fonction des caractéristiques qu'ils possèdent.

Un exemple de modèle d'agents situés et dotés d'un module de délibération est représenté dans la Figure 2.2. Cette architecture décompose le comportement d'agents en un ensemble de modules fonctionnels. De ce modèle ressort donc une partie des aspects et modules importants d'agents. Le module de *Perception* prend en charge la perception de l'agent en associant l'état local de l'environnement à une perception (pi). Le module de *Mémorisation* permet à l'agent

de mémoriser les informations qu'il peut avoir, en d'autres termes lors d'une mise à jour l'agent utilisera l'information de la perception la plus récente afin d'ajuster son état interne (state). Le module de *Décision* représente le cœur de l'agent lui permettant de décider de l'action à exécuter. L'agent prend donc des décisions en fonction de la perception la plus récente (pi), la connaissance actuelle (si) en sélectionnant l'opérateur qui convient (oi) afin d'exécuter l'action. Cette action aura un effet sur l'environnement qui changera son état.

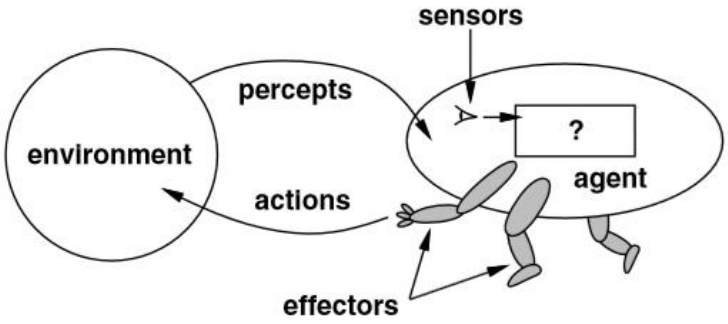


FIGURE 2.1 – La représentation d'agent en interaction avec son environnement [10]

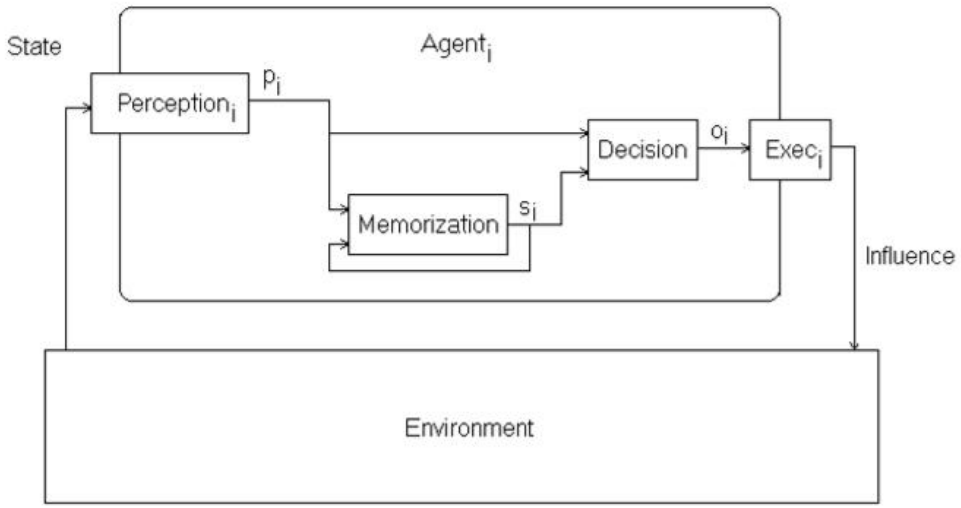


FIGURE 2.2 – Exemple de modèle générique d'Agents [11]

2.3.1 Les architectures d'agents :

Les architectures d'agents représentent les caractéristiques internes de la structure d'agents. Elles sont indépendantes des interactions que l'agent pourrait avoir avec d'autres agents, et indépendantes de l'organisation du système. L'architecture d'un agent sert à définir ses fonctionnalités afin de lui permettre d'agir et d'interagir dans un environnement dynamique. La plupart des architectures existantes suivent la boucle perception-délibération/décision-action.

Il existe différentes classifications d'architectures d'agents, la plus courante consiste à considérer les trois catégories [56] : (i) Réactive, (ii) Délibérative et (iii) Hybride.

Les agents réactifs suivent un cycle de type Perception-Stimulus- "Re"action ; alors que les agents délibératifs suivent un cycle de type Perception-Delibération-Action,

Certaines méthodologies prennent justement en ligne de compte ces aspects de capacités mentales de l'agent comme la méthodologie Ingenias [57] qui reprend entre autre des concepts que nous retrouvons au sein du modèle d'architecture délibérative la plus connue et utilisée dans le domaine multi-agents, soit l'architecture délibérative BDI [58] qui est devenue le modèle standard parmi les modèles d'agents représentés à la base du standard FIPA [59].

2.3.1.1 L'architecture délibérative BDI :

Le modèle Belief-desire-intention ou BDI [58] comprend trois attitudes mentales : Croyances, Désirs et Intentions.

La figure 2.3 représente l'architecture BDI composée des principaux modules qui la constituent. La première fonction BRF permet de mettre à jour les croyances de l'agent en fonction des perceptions que ce dernier reçoit en entrée. Grâce à la fonction OGF les désirs de l'agent sont générés avant d'être filtrés afin que l'agent puisse déterminer ses nouvelles intentions. Et enfin, l'agent décide de l'action à exécuter en sortie.

2.3.2 Perception d'agents :

Comme nous l'avons vu plus haut la perception représente l'un des modules importants de l'agent. Elle est définie comme étant "l'habilité de l'agent à capter son environnement, et donnant comme résultat une perception de cet environnement. Les percepts décrivent donc l'environnement sous forme d'expressions comprises par l'agent" [11]. La perception dans tous les cas joue un rôle clé pour l'agent, c'est pourquoi nous retrouvons ce concept au niveau de plusieurs méta-modèles tels que celui de Prometheus [60] où le concept "percept" est directement associé au concept "capability", c'est à dire que la perception de l'agent est associée aux aptitudes de ce dernier.

Il existe deux types de perception [61] [62] : *Active* et *Passive*.

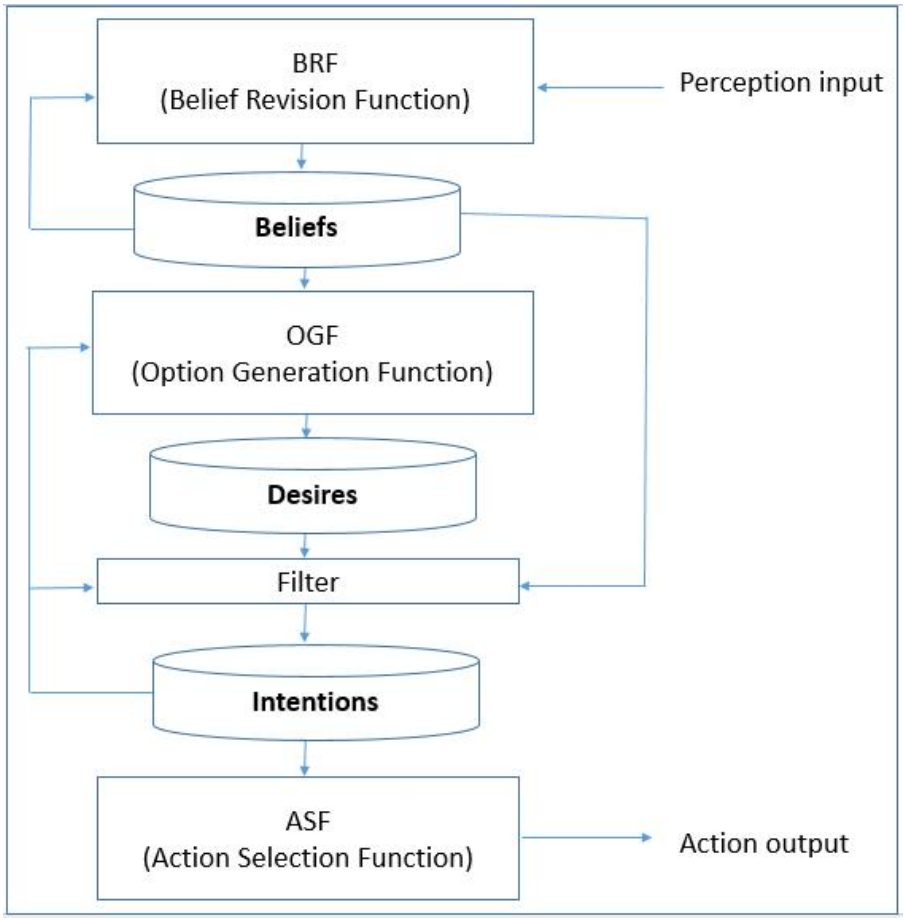


FIGURE 2.3 – Le modèle d'architecture d'agents BDI

La perception dite *Active* est définie par un processus descendant (top-down), elle est idéalement contrôlée par un processus de raisonnement orienté but (goal-oriented), et permet à l'agent de diriger sa perception sur les aspects les plus pertinents de son environnement relativement aux tâches qu'il devra accomplir.

La perception *Passive* quand à elle est définie par un processus inverse et par conséquent ascendant (bottom-up). Ce type de perception est à l'origine des capteurs permettant d'atteindre des buts indépendant (goal-independent), ainsi qu'une perception opportuniste.

Dans certains cas la perception passive peut s'avérer insuffisante comme dans le cas d'environnement physique. En effet, dans ce type d'environnement, les ressources de l'agent associées au processus de perception sont délimités ; les capteurs ont des capacités limitées également telles que la distance, de plus les objets peuvent être hors de portée.

2.3.2.1 Modèle de perception active :

La perception active permet à l'agent de diriger sa perception vers les aspects les plus pertinents de l'environnement relativement à la tâche courante qu'il devra réaliser. Il existe différents modèles pour ce type de perception [11] [63] [61].

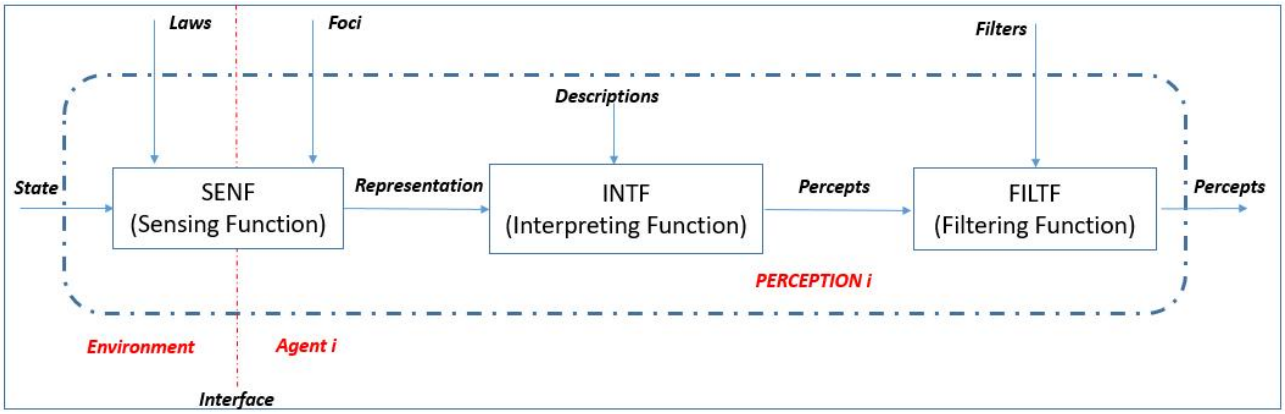


FIGURE 2.4 – Un modèle générique de perception active [11]

Les principales fonctionnalités de ce type de perception sont 1) Une fonction *sensing*, 2) une *interpreting* et 3)une fonction *filtering* [11].

La fonction *sensing* permet de capter et sélectionner un ensemble de *foci*. La sélection de focus permet à l'agent de capter des types spécifiques de données de l'environnement. Suite à cela, la représentation de l'état est composée en fonction d'un ensemble de lois relatives à la perception. Ces lois imposent des contraintes spécifiques du domaine sur la perception. Alors que la détection physique (capteurs) incorpore naturellement de telles contraintes, dans l'ingénierie multi-agents les contraintes doivent être modélisées explicitement.

La fonction *interpreting* permet aux agents d'interpréter les représentations par le biais de *descriptions*. Ces dernières permettent d'associer les représentations aux perceptions. Ces perceptions sont représentées sous forme d'expressions compréhensibles par l'agent de façon interne.

Enfin la fonction de filtrage ou *filtering* a lieu par la sélection d'un ensemble de filtres par l'agent et ce dans le but d'améliorer sa perception en restreignant les données perçues.

2.3.2.2 Modèle de perception passive :

La majorité des architectures d'agents proposent une perception passive qui permet à l'agent d'avoir des informations sur son environnement, sans avoir la possibilité de rediriger sa perception de façon à ce qu'elle soit pertinente. De plus, ce type de perception comme nous l'avons

vu restreint certains éléments de la perception comme son étendue par exemple. Généralement, elle n'est donc pas optimale et ne convient pas toujours lorsqu'il s'agit d'environnements physiques [64].

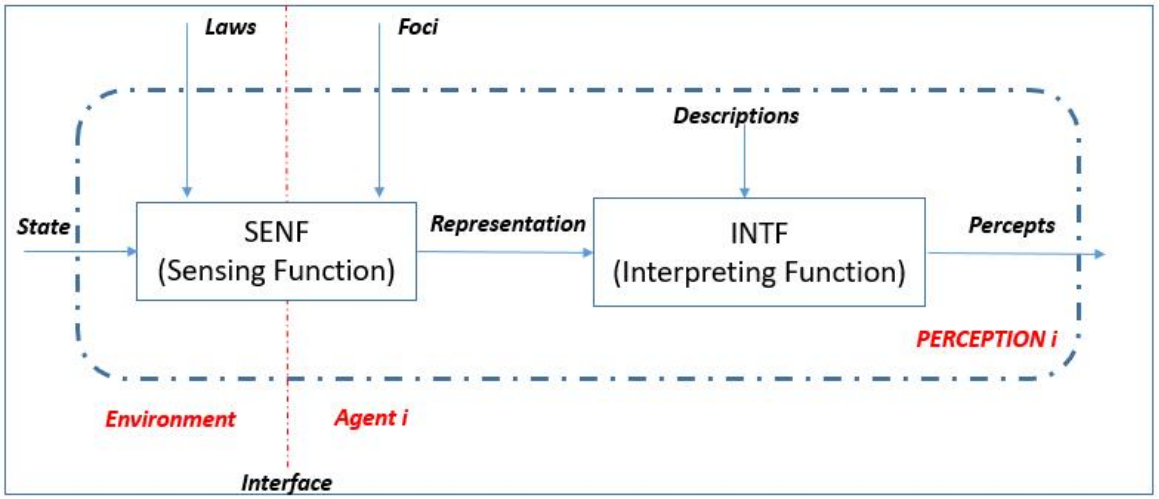


FIGURE 2.5 – Un modèle générique de perception passive

2.4 L'environnement d'agents

Un SMA représente un ensemble d'entités autonomes (agents) situés au sein d'une entité structurée et partagée (environnement). L'une des premières définitions d'agents autonomes proposée par Graeser et Franklin [65] met l'accent sur l'importance de l'environnement : *" an autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future"*.

2.4.1 Le rôle de l'environnement

L'environnement peut être vu comme le lieu où les agents interagissent avec les objets du domaine ainsi que ses ressources, et également avec d'autres agents. Il est possible de résumer les principaux aspects de l'environnement mettant en avant le rôle que l'environnement peut avoir à différents niveaux :

1. ***Structuration du système*** **:**

   L'environnement est considéré comme un espace commun et partagé par les agents qui peut structurer le système [16]. La structuration du système multi-agents peut se faire

d'une part par le biais d'organisations, mais encore de façon spatiale pour l'environnement [66].

Dans le cas de structures spatiales de l'environnement, des propriétés telles que la localité ou encore le positionnement d'agents devront être pris en compte. Ceci est le cas lors d'interactions indirectes médiatisées par l'environnement avec prise en compte des caractéristiques spatiales (voir figure 2.14).

2. ***Gestion des services et ressources :***

   L'environnement peut inclure des services et des ressources. L'environnement sert de ce fait de conteneur. Les services sont considérés comme des entités réactives qui encapsulent certaines fonctionnalités, tandis que les ressources représentent des objets pouvant changer d'états. L'environnement gère ces services et ressources tout en permettant aux agents d'y accéder, mais également en contrôlant cet accès [16].

   Nous retrouvons d'ailleurs ces deux concepts dans plusieurs méta-modèles. Par exemple, le méta-modèle supporté par la méthodologie Gaia [67] considère que l'environnement est représenté par un ensemble de *ressources* dont les agents auraient besoin et avec lesquelles ils interagissent , ou encore sur lesquels ils agissent afin de mener à bien leurs rôles. Quant au concept de *services*, Gaia considère qu'ils sont fournis par l'agent et qu'ils doivent être conçus sous forme de blocs d'activités dans lesquels les agents s'engageront car ils s'avèrent nécessaires pour la réalisation des rôles des agents [67].

3. ***Mise en place de règles d'accès aux ressources :***

   L'environnement peut définir différents types de règles au sein du système. Les règles peuvent par exemple restreindre l'accès à des ressources spécifiques ou à des services pour des types particuliers d'agents. Les règles d'environnement peuvent devenir un outil très puissant pour exprimer les capacités d'un environnement qui doit assurer la cohérence du système multi-agents [16]. Par exemple, dans les institutions électroniques [68], les agents doivent suivre des protocoles de communication bien définis. Ces règles peuvent ainsi restreindre ou étendre le champ d'action de l'agent. L'environnement peut ainsi définir et appliquer les règles imposées aux interactions d'agents au sein de l'institution électronique.

2.4.2 Description de l'environnement

Lors d'interactions indirectes entre agents, il est possible de distinguer deux catégories d'entités environnementales sur lesquelles l'interaction se produit. Nous parlons alors de médiatisation : 1) *artefact-based* et 2) *spatial-based* (voir figure 2.7).

2.4.2.1 Environnements avec artefacts

Le concept *artefact* a été utilisé dans différents domaines. A l'origine dans la théorie de l'activité (AT) [69], l'artefact est considéré comme un outil servant de médiateur d'interactions tout en permettant aux participants d'effectuer leurs différentes actions. D'une part, les artefacts médiatisent l'interaction entre les individus, mais aussi entre les individus et leur environnement. D'autre part, les artefacts englobent une partie de l'environnement pouvant servir de support pour les différentes activités des participants. On peut distinguer deux types d'*artefact* : *physiques* ou *cognitifs*.

Ce concept d'*artefact* a été réadapté par le méta-modèle *A&A* Agent and artefact proposé par Ricci et al. [55]. Dans ce méta-modèle, les agents représentent des entités (pro)actives en charge des objectifs / tâches qui forment l'ensemble du comportement du SMA, alors que les artefacts sont les entités réactives fournissant les services et les fonctions permettant aux agents de travailler ensemble au sein du système, et que l'environnement de l'agent se forme selon les besoins du système.

La Figure 2.6 par exemple illustre une représentation abstraite d'un workspace (espace-de-travail), comportant quatre agents travaillant ensemble, tout en partageant et utilisant trois artefacts.

Les agents exécutent des actions afin d'utiliser les artefacts, et ce tout en exploitant leur perception. L'artefact de la partie supérieure de la figure inclut l'interface d'utilisation (usage interface) qui représente l'ensemble des opérations pouvant être déclenchées par les agents.

Ce type de médiatisation *artefact-based* s'articule donc autour de la conception et de l'implémentation d'*artefact* qui émule des objets concrets de l'environnement d'agents, dont le but est de permettre la communication entre ces entités autonomes. Dès lors que les *artefacts* influencent la façon avec laquelle les agents agissent et interagissent au sein du SMA, ils peuvent aussi potentiellement changer le raisonnement des agents sur les actions.

Nous pouvons alors considérer les artefacts de coordination comme une généralisation du médium de coordination [12]. L'exemple classique le plus connu dans ce contexte et qui est le premier à proposer un média d'interaction indirecte est le modèle architectural *Blackboard* [70] [71]. Ce dernier, a été largement utilisé pour aborder différents types de problèmes avec des aspects d'incertitude, et non déterministe. Ce pattern peut être appliqué comme solution à différents cas d'application tels que la communication, la coordination et la collaboration au sein d'un SMA [72].

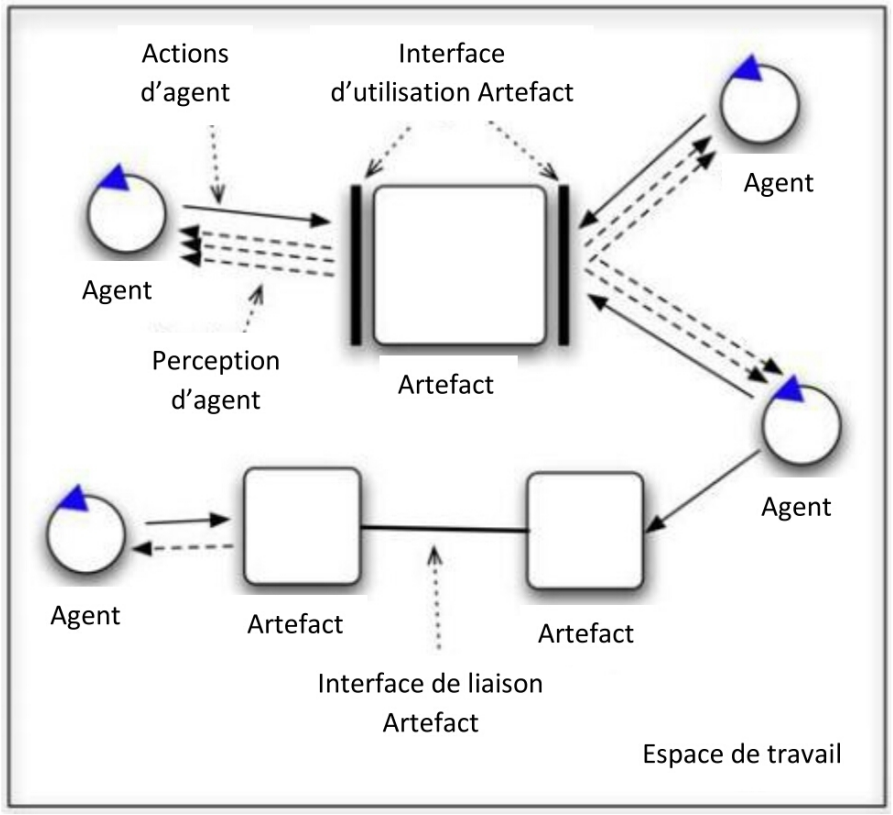


FIGURE 2.6 – Un exemple de représentation de l'abstraction espace-de-travail (workspace) avec des agents qui interagissent indirectement par le biais d'artefacts [12]

2.4.2.2 Environnements avec caractéristiques spatiales

Il existe des situations dans lesquelles les caractéristiques spatiales de l'environnement représentent un facteur clé ne pouvant être négligé dans l'analyse et la modélisation du système. C'est pourquoi cette catégorie d'environnement dite *spatial-based* est basée sur l'intégration des caractéristiques spatiales de l'environnement au niveau du modèle. Parmi les ancêtres des modèles qui fournissent une représentation de la structure spatiale de l'environnement, nous retrouvons le modèle mathématique d'automates cellulaires CA (Cellular Automata) [73] qui a été intégré à des approches multi-agents comme l'approche proposée par [74] où chaque agent est situé dans un espace simulé et basé sur la grille des automates cellulaires.

D'autres travaux de recherche proposent d'intégrer des modèles d'environnement spécifiques aux SMA comme Bandini et al. [66] qui proposent un modèle MMAS (Multilayered Multi-Agent Situated System) servant de support pour la définition d'architectures conceptuelles et fournissant un concept fort d'environnement d'agent, qui représente une abstraction d'un environnement physique éventuellement interfacé avec des représentations d'aspects conceptuels.

Leur modèle fournit deux mécanismes de base pour l'interaction d'agents (réaction et émission de champ) qui dépendent fortement de la structure spatiale de l'environnement.

2.5 Les interactions multi-agents

il existe plusieurs dimensions et modèles d'interactions d'agents pouvant servir à définir une taxonomie possible. Cependant, dans le cadre de ce travail de recherche, nous nous sommes basés sur la taxonomie conceptuelle d'agents réalisée à partir des différents modèles d'interactions existants (voir Figure 2.7).

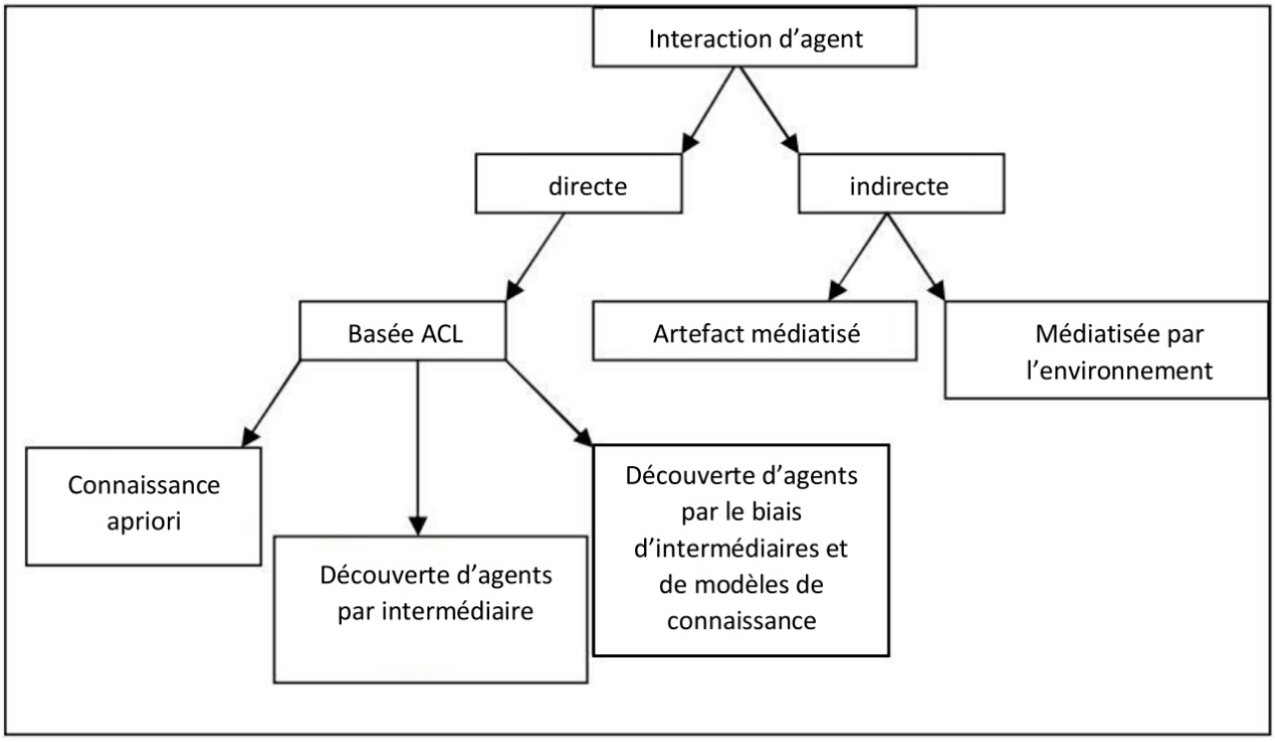


FIGURE 2.7 – La taxonomie d'interactions multi-agents [13]

2.5.1 Les interactions directes

Les interactions directes permettent aux agents de communiquer par le biais d'échanges de messages. Les modèles les plus adoptés dans le domaine étant ceux basés sur le langage de communication ACL (Agent Communication Language) dont la spécification a été proposée par FIPA et mieux connue sous le nom de FIPA-ACL [75]. Les interactions directes font intervenir

principalement les différents schémas d'interactions possibles entre les agents. La formalisation de ces échanges de messages est faite par le biais des protocoles d'interactions.

2.5.1.1 Les protocoles d'interaction

Le concept de *protocole d'interaction* fait partie de la plupart des méta-modèles du domaine multi-agents. La méthodologie Passi [76] par exemple associe le concept AIP (Agent Interaction protocol) à la communication entre agents. Dans Opera [77] le protocole d'interaction est une agrégation de messages. En effet,un protocole d'interaction n'est autre qu'un enchaînement prédéfini de messages échangés entre des agents qui interagissent. Rappelons que FIPA (Foundation of Intelligent Physical Agents) [1] le définit comme étant un pattern de communication. Il existe différents aspects liés aux protocoles d'interactions tel que les comportements qui définissent les rôles d'interactions (*d'initiateur* et *participant*), ainsi que l'objet d'interaction. Ces protocoles d'interactions standardisés par FIPA sont définis sur la base des messages FIPA-ACL et utilisent les actes de communication indépendants de la plateforme d'implémentation d'agents.

A titre d'exemple et dans ce qui suit, nous détaillons quelques protocoles d'interactions :

1. **Le protocole *Contract-Net*** Le protocole *Contract-Net* (CNP) [78], est un protocole à haut niveau supportant la communication entre les agents au sein d'un SMA tout en offrant un contrôle distribué de l'exécution de tâches coopératives et de négociations compétitives. La spécification proposée par FIPA pour ce protocole est représentée dans la Figure 2.8

   Dans ce protocole, les agents peuvent prendre deux rôles soit celui de l'initiateur ou celui du participant.

   L'initiateur est responsable de l'exécution d'un service donné telle que l'exécution d'une tâche, ainsi que du traitement des résultats de cette exécution. Le participant quant à lui fait une proposition à l'initiateur en réponse à sa sollicitation. En cas d'acceptation, il est responsable de l'exécution effective de ce service sur lequel porte la négociation.

   L'initiateur commence par envoyer un message de type CFP (call for proposal) aux m agents participants (potentiels contractuels). Cet envoi est réalisé tout en transmettant le service à réaliser tel que l'action à exécuter et ses contraintes s'il y-en a (timeout ou autre). Parmi les m récepteurs du CFP, n (avec $n<=m$) répondent après avoir fait appel à une *stratégie de proposition* soit par une proposition (PROPOSE), ou bien un refus (REFUSE).

   L'initiateur analyse ensuite les j réponses qu'il aura reçu tel que $j=n-i$, et choisit la proposition $l=j-k$ (ACCEPT), ce qui impliquera le rejet (REJECT) des k ($k<=j$) pro-

1. www.fipa.org

positions restantes. Dans tous les cas, l'initiateur devra répondre par l'acte de communication adéquat (ACCEPT ou bien REJECT) à l'ensemble des participants ayant fait une offre. Le nombre l d'agents sélectionnés ne peut comporter aucun des agents, ou un seul, ou quelques uns ou bien l'ensemble de tous les agents afin de réaliser le service en question. Les agents ayant été choisis pour le traitement de la tâche doivent informer l'agent initiateur du statut de ce traitement en envoyant un message INFORM. En cas d'annulation ou d'échec de l'exécution du service le participant devra alors répondre par un CANCEL ou un FAILURE relatifs aux deux situations respectives.

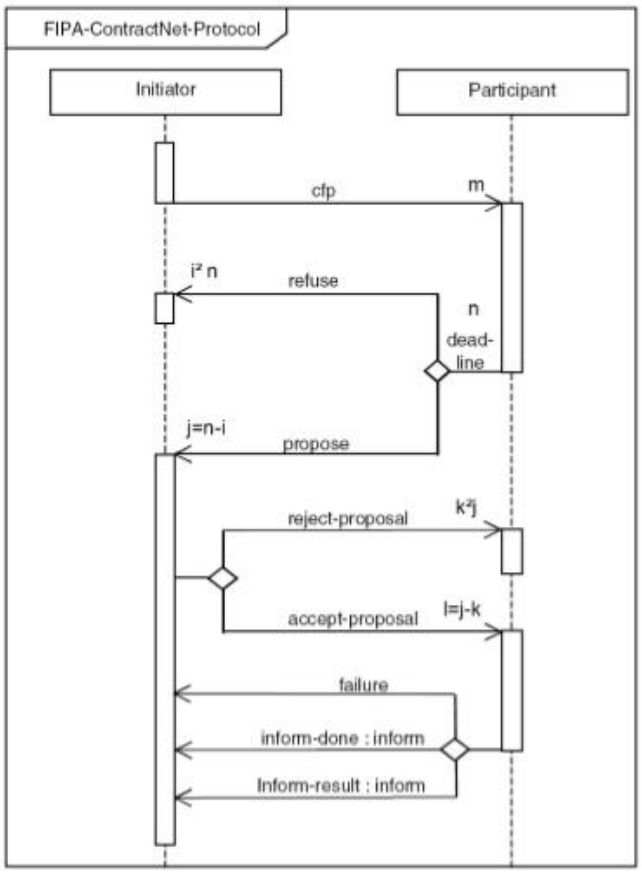


FIGURE 2.8 – Spécification du protocole FIPA Contract Net

2. **Le protocole *English-Auction*** La figure 2.9 représente la taxonomie d'enchères existantes, incluant l'ensemble des protocoles d'enchères existants. Parmi lesquels nous retrouvons le protocole d'enchère Anglaise ou English-Auction. Ce protocole permet d'effectuer des offres publiquement (connue par l'ensemble des participants) et de façon ascendante (prix croissant à chaque tour).

   La spécification FIPA du protocole d'enchère Anglaise (*English-Auction*) est représentée dans la Figure 2.10. Dans ce protocole [79] [80], le commissaire-priseur *auctioneer* cherche

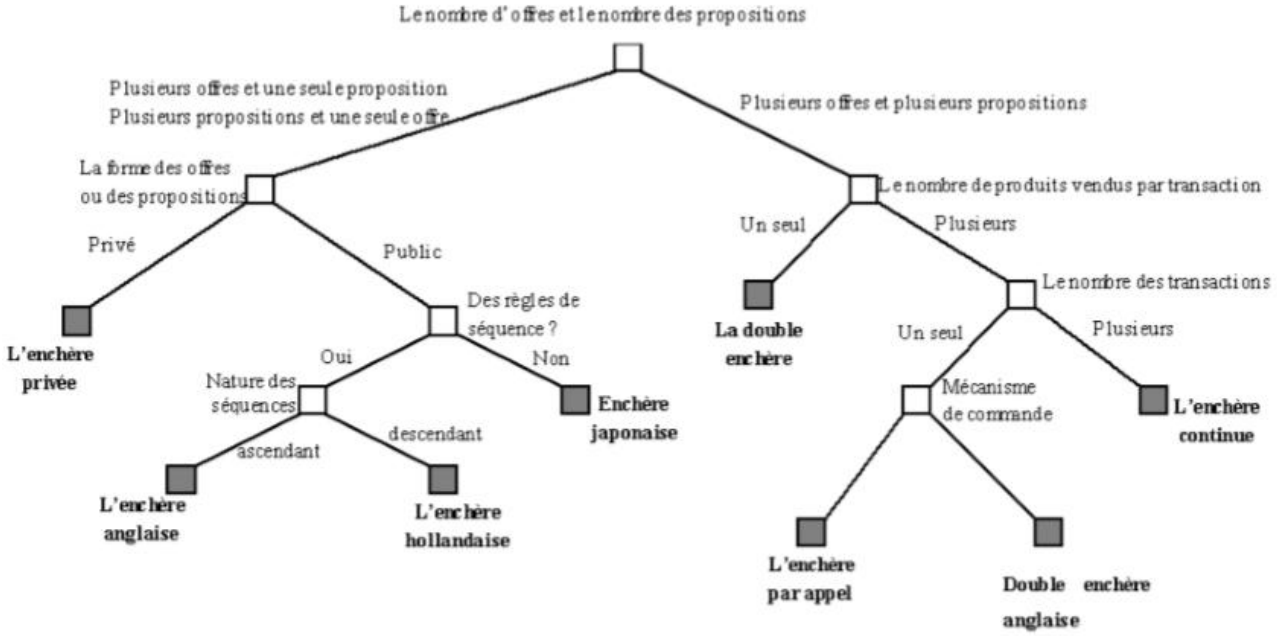


FIGURE 2.9 – Taxonomie des protocoles d'enchères [14]

à trouver le prix du marché d'un bien, en proposant initialement un prix inférieur à la valeur de marché supposée, pour augmenter ensuite, progressivement le prix. Chaque fois que le prix est annoncé, le commissaire aux enchères attend pour voir si les acheteurs (bidder) afficheront leur volonté de payer le prix proposé.

Dès que l'un des acheteurs indique qu'il accepte le prix, le commissaire priseur émet un nouvel appel d'offres avec le dernier prix proposé afin d'obtenir de meilleures propositions de la part des acheteurs. La vente aux enchères continue, jusqu'à ce que les acheteurs ne manifestent plus d'intérêt quant au prix proposé.

Les appels du commissaire-priseur, exprimés par envoi de message CFP, sont multicast pour tous les participants à la vente aux enchères. Les propositions soumises par les enchérisseurs concernent principalement le processus d'appel d'offres.

Un exemple d'échanges entre des agents qui suivent ce protocole pourrait être que suite à la réception d'un message CFP à soumettre des offres pour une vente aux enchères d'un produit X, une proposition de l'ordre : "Je propose le prix Z pour X " soit analysée puis comparée à l'ensemble de toutes les propositions reçues. Chaque proposition faite au commissaire-priseur sous-entend que le participant peut payer le prix sans s'engager à le payer jusqu'à l'annonce par le commissaire-priseur du gagnant. La transaction d'enchères est ainsi achevée, et la vente du bien X est finalisée avec seul le gagnant de l'enchère au prix accepté.

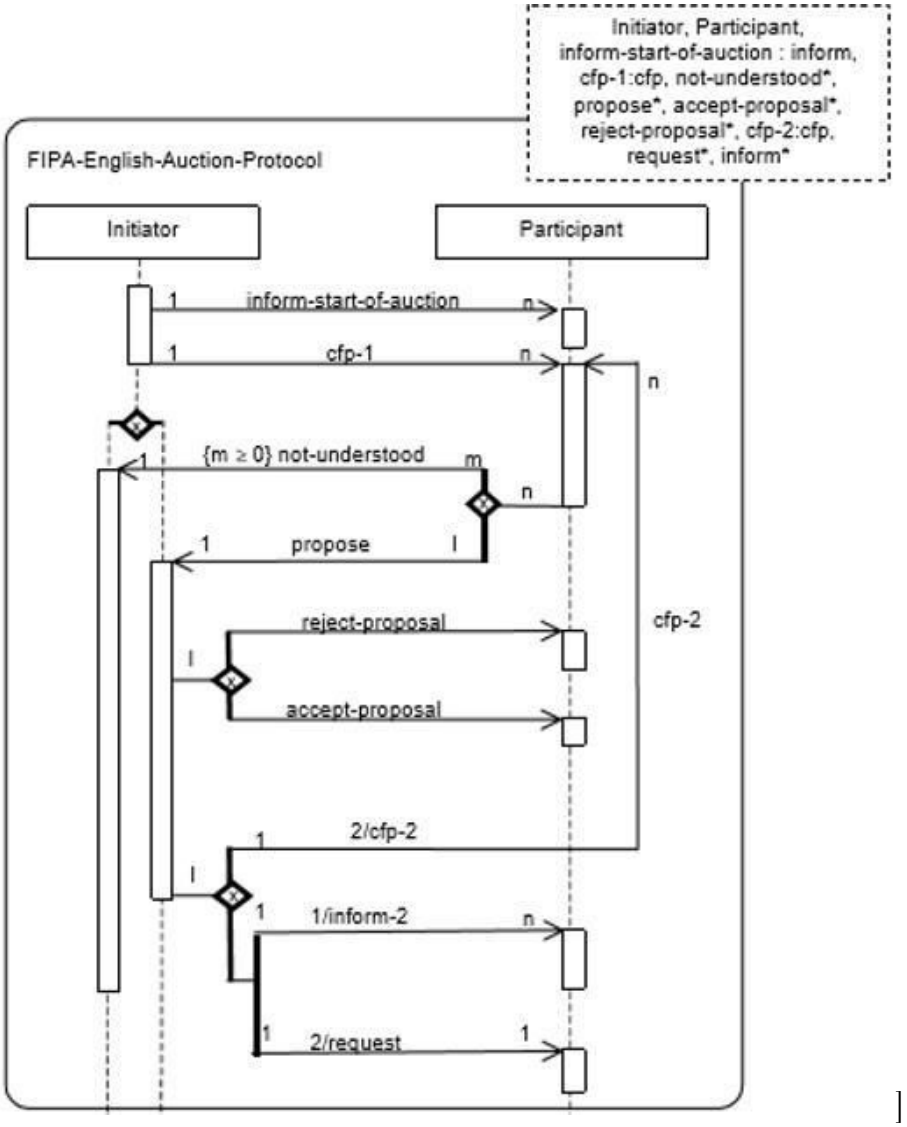


FIGURE 2.10 – Protocole d'enchères anglaise FIPA English-Auction

Les rôles présentés dans le protocole sont représentés par un rôle initiateur d'enchères représentant le comportement relatif à l'initiateur de l'enchère (commissaire-priseur), ainsi qu'un rôle participant qui représente le comportement adopté par un enchérisseur.

2.5.1.2 L'ontologie de protocoles d'interactions ONTOPRO

L'ontologie ONTOPRO est une ontologie de protocoles d'interactions à laquelle nous nous sommes intéressés pendant notre travail de recherche (voir chapitre 5). Avant de la présenter, nous allons donner les aspects les plus importants de l'*ontologie*, ainsi que quelques définitions :

— "An Ontology is a means of enabling communication and knowledge sharing by capturing

a shared understanding of terms that can be used both by humans and machine software" [81]

— "An ontology defines the basic terms and relations comprising the vocabulary of a topic area as well the rules for combining terms and relations to define extensions for this vocabulary" [82]

— "An ontology provides the meta-information to describe the data semantics, represent knowledge, and communicate with various types of entities (e.g., software agents and humans)" [83]

L'ontologie ONTOPRO [84] [15] dont les concepts sont représentés dans la Figure 2.11, est une ontologie qui a été conçue pour décrire les protocoles d'interactions. Elle est issue de l'analyse d'un ensemble de protocoles d'interactions standardisés par FIPA [2], soit les protocoles d'enchères anglaises [79] et du Contract-Net [78].

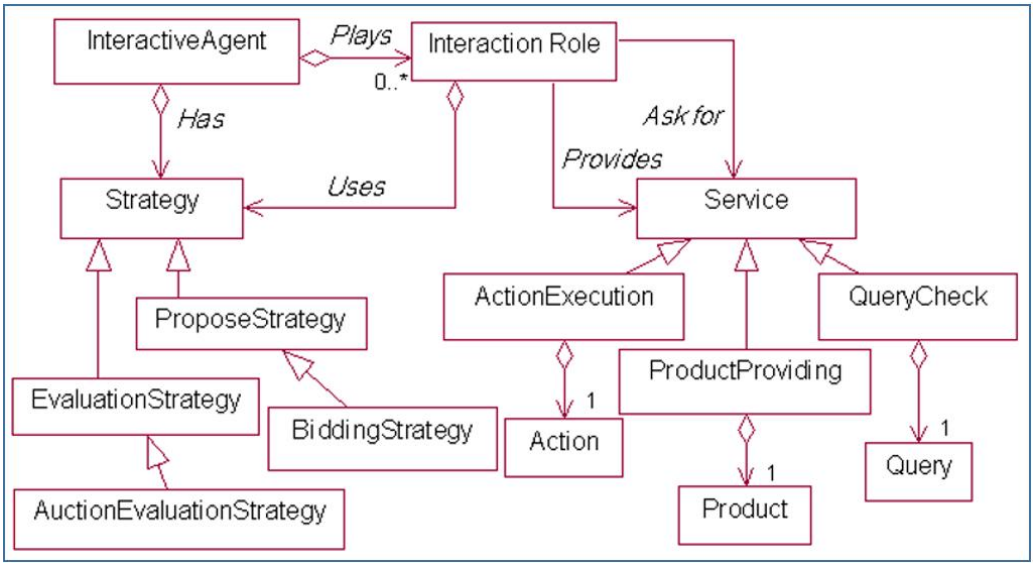


FIGURE 2.11 – Les concepts de l'ontologie ONTOPRO [15]

Les concepts de l'ontologie ONTOPRO sont principalement une réification des paramètres d'entrée des rôles d'interaction, elle comporte les concepts suivants :

1. ***Service :*** Ce concept fait référence à *l'objet de l'interaction.* Ce dernier offrant ainsi une solution d'unification des différents sujets possibles sur lesquels porte l'interaction. En effet, un objet d'interactions peut concerner l'exécution d'une tâche, la vérification d'un prédicat ou encore l'achat d'un produit. Dans les protocoles FIPA-Contract-Net, FIPA-Propose et FIPA-Request, l'objet de l'interaction représente l'éxécution d'une tâche,

2. www.fipa.org

tandis que dans les protocoles FIPA-English-Auction et FIPA-Dutch-Auction l'objet de l'interaction représente l'achat d'un produit.

2. ***Stratégie :*** Ce concept fait référence aux *modèles décisionnels*. Les rôles d'interactions peuvent utiliser un ensemble d'actions décisionnelles. Les concepteurs d'ONTOPRO ont défini une taxonomie de stratégies en distinguant deux grandes classes de stratégies :1) la *stratégie d'évaluation* qui permet de choisir une proposition de service parmi l'ensemble des propositions reçues, et 2) la *stratégie de proposition* qui est utilisée pour faire une proposition pour service donné.

2.5.2 Les interactions indirectes

Les interactions indirectes entre les agents peuvent être médiatisées de deux façons différentes (voir Figure 2.7) : par les artefacts inclus au sein de l'environnement ou bien par l'environnement lui-même.

La première catégorie de ces approches *Mediated artefacts* s'articule autour de la notion d'*artefact* qui sert de média de coordination lors de l'interaction indirecte [13]. La notion d'artefact dans ce contexte provient certes du modèle d'environnement d'agents A&A [55], mais joue également un rôle important lors d'interactions indirectes. La seconde catégorie se concentre principalement sur l'environnement à modéliser vu qu'il représente l'emplacement où auront lieu les interactions, et influencera de ce fait l'interaction et le comportement d'agents. La majorité des approches indirectes adopte une infrastructure de communication qui va fournir un mécanisme pour la mise en place de l'interaction.

L'environnement qui représente une classe d'abstraction de premier ordre joue entre autres le rôle de *médiation d'interaction* [16]. La Figure 2.12 représente un exemple mettant en avant une interaction d'agents médiatisée par l'environnement.

Dans cet exemple, les agents interagissent avec une structure à base de *phéromones* déployée au sein d'une structure de marché électronique (E-market). Parmi les agents se trouvant à droite de la figure, un agent désirant intégrer le groupe au sein de l'institution électronique, tandis que l'autre agent fixe le prix d'un produit particulier grâce à sa stratégie d'évaluation. La couche de médiation d'interaction reste cependant transparente pour certaines ressources. Tel est le cas par exemple pour l'abstraction d'*agenda*.

La figure 2.14 présente un second exemple d'interactions indirectes. Il s'agit dans cet exemple de médiation de l'interaction au sein d'un environnement présentant des caractéristiques spatiales. Cette figure montre un aspect important lié aux interactions indirectes. Il s'agit du principe de *Stigmerie*, qui représente l'un des mécanismes d'auto-organisation grâce auquel l'interaction indirecte se produit. Dans ce cas, les interactions se produisent en raison de changements effectués par les agents au sein de l'environnement. Ces changements sont perçus par

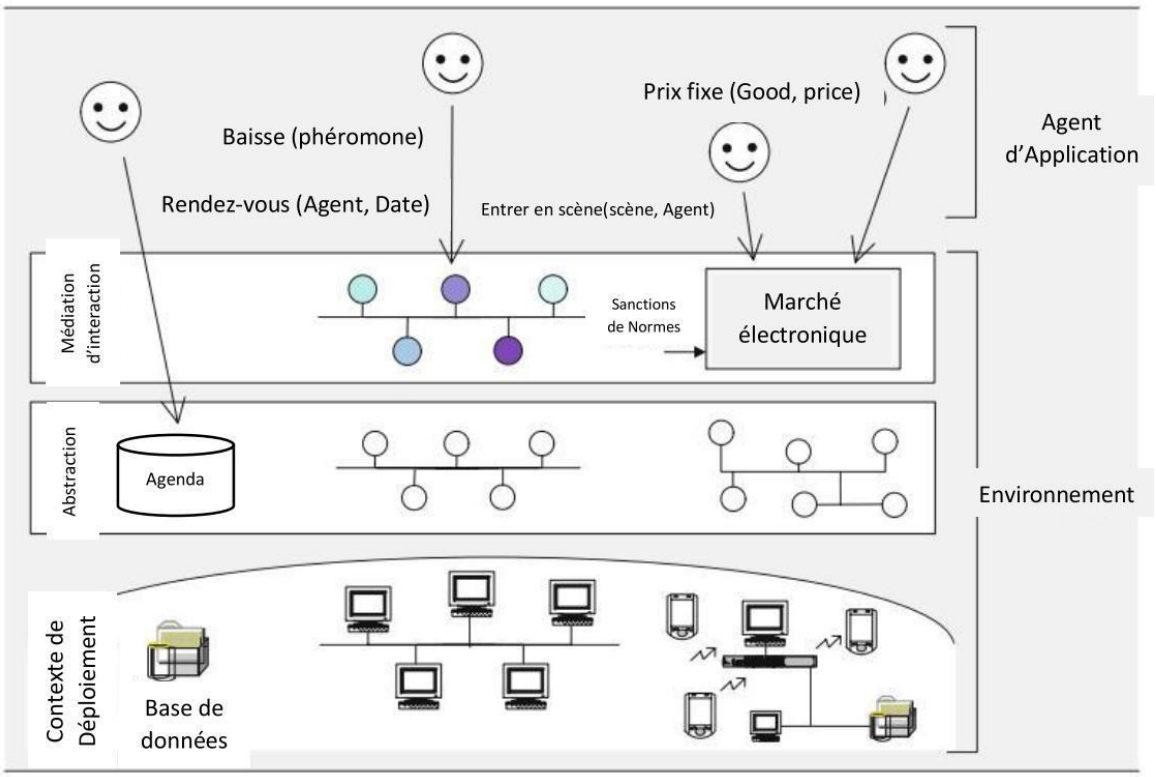


FIGURE 2.12 – Un exemple d'interaction médiatisée par l'environnement à travers des ressources [16]

d'autres agents qui modifient leur comportement en conséquence, et éventuellement mènent le système vers l'état global désiré.

Les interactions indirectes peuvent également exploiter d'autres mécanismes inspirés d'autres domaines tel que l'utilisation des *champs potentiels*. Ferber [17] précise que les objets présents dans l'environnement y compris les agents émettent des signaux dont l'intensité est fonction de la distance de l'émetteur. Ces signaux définissent des champs attractifs pour les buts, ou répulsifs pour les obstacles. Dans la figure 2.13 l'agent est au sein d'un environnement qui présente un tel champ comprenant un but ainsi que deux obstacles.

2.5.2.1 Champs potentiels

Les *champs potentiels* s'inspirent de la physique et reposent sur l'utilisation de champs scalaires ou, plus généralement, de champs vectoriels. De ce fait, chaque point de l'espace est associé à un vecteur. D'un point de vue pratique, ce type d'approche est principalement utilisé pour des tâches de navigation. Par exemple, Simonin [85] propose dans sa thèse un modèle de foraging dans lequel un agent ayant trouvé une source de nourriture émet un signal attractif pour

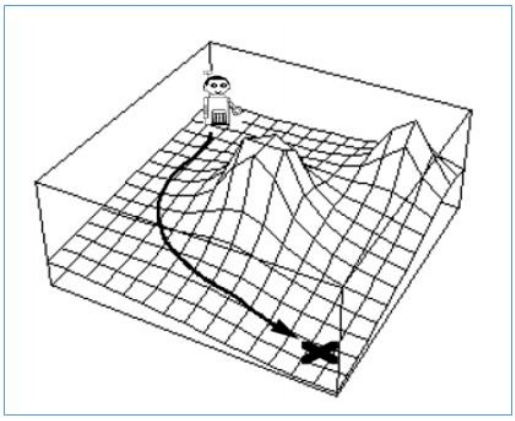


FIGURE 2.13 – Un exemple en 3D de champs potentiel avec un but et deux obstacles pour l'agent [17]

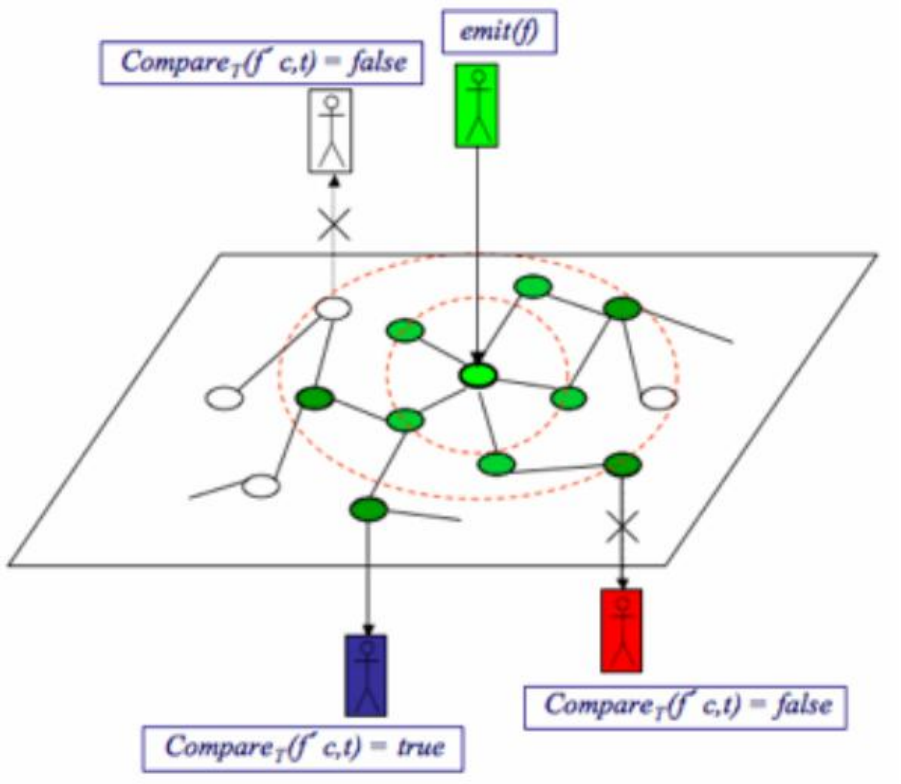


FIGURE 2.14 – Exemple d'interactions indirectes par médiatisation au sein d'une structure spatiale d'environnement [13]

attirer les autres agents. De façon plus générique, les agents peuvent en observant leur voisinage

(neighbourhood) déduire des informations de gradients utiles à la résolution du problème. Ce type d'approches a également été utilisé dans le domaine de la gestion de chaîne logistique ou supply chain management. Dans la thèse de Moujahed [86] par exemple, le modèle positionne des usines et des dépôts afin d'optimiser le coût de transport des marchandises des usines vers les dépôts, et des dépôts vers les clients.

2.5.2.2 Mécanismes d'interactions indirectes par Stigmergie :

La *Stigmergie* représente l'un des mécanismes de communication indirecte et de coordination au moyen de traces crées et détectées (via capteurs) dans un environnement partagé (voir figure 2.15).

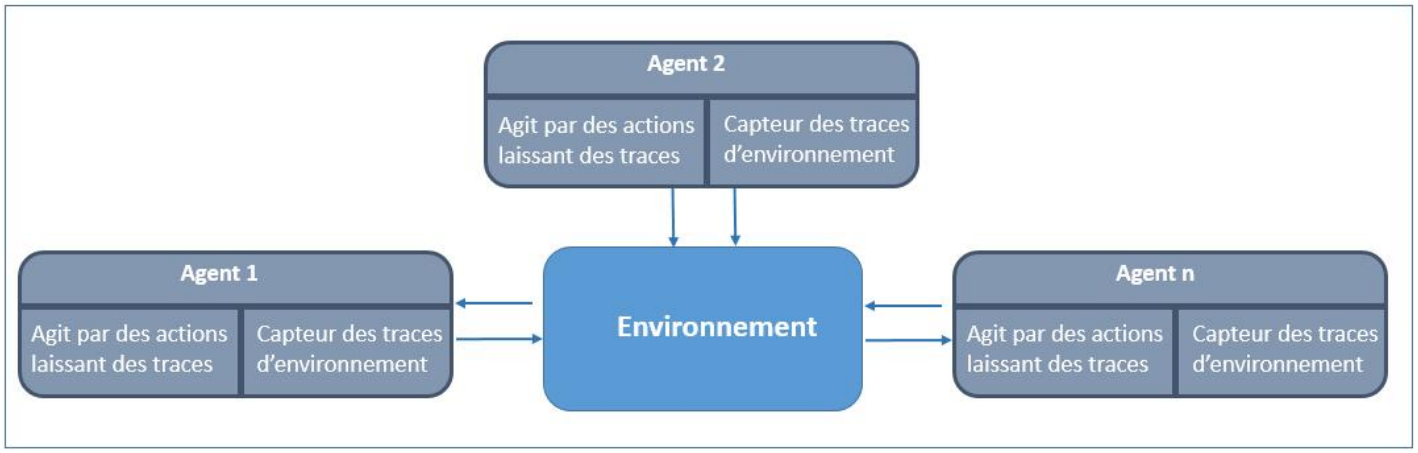


FIGURE 2.15 – Mécanisme de *Stigmergie* d'interactions indirectes [18]

Le concept de *Stigmergie* a été introduit par Grasse [87] afin d'expliquer la construction de nids dans les colonies de termites. Le concept indique que des entités individuelles *interagissent indirectement* à travers un environnement partagé. Un individu modifie son environnement et répond à toute modification de l'environnement en le modifiant à son tour.

Une récente définition de ce concept stipule que *"stigmergy is an indirect, mediated mechanism of coordination between actions, in which the trace of an action left on a medium stimulates the performance of a subsequent action"* [19].

La figure 2.16 [19] décrit le processus de *stigmergie* sous forme de boucle, où une *action* produit une *marque*, qui à son tour incite une *action* qui produira également une *marque*, et ainsi de suite.

En résumé, les actions stimulent leur propre exécution continue par l'intermédiaire des *marques* que ces actions produisent elles-même. Une *marque* sera considérée comme un effet perceptible, trace ou produit d'une action.

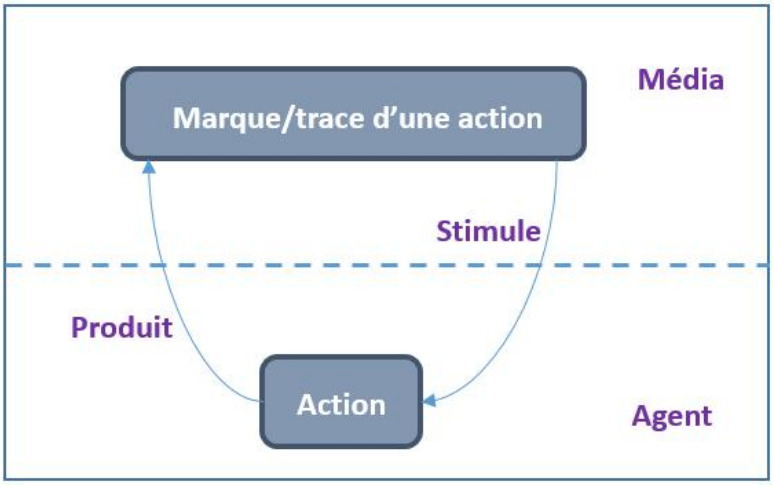


FIGURE 2.16 – La boucle de rétroaction stigmergique [19]

2.5.2.3 Modèles à base de phéromones digitales :

Les phéromones digitales sont le pendant artificiel des phéromones utilisées par les insectes, et que les agents peuvent déposer au sein d'un environnement pouvant être représenté sous forme matricielle.

En comparaison aux champs potentiels, les phéromones digitales offrent des mécanismes d'évaporation et de diffusion qui permettent d'une part une persistance (temporelle) limitée, et d'autre part une diffusion de l'information dans l'environnement. Le processus d'évaporation est un phénomène qui mène progressivement à la disparition de la phéromone. Ce processus a pour but de faire disparaître les informations obsolètes, et contribue à la création de gradients (traces orientées) que les agents pourront suivre. Ce processus peut être modélisé par une suite géométrique (l'évolution dans le temps de la quantité de phéromone sera donc discrète).

Différents algorithmes de fourragement, de recherche et de transport de ressources, ou encore d'optimisation de parcours au sein d'un graphe ont été proposés. Dans le modèle de Parunak et al. [88], les auteurs proposent des modèles de coordination entre drones (véhicules sans pilotes) pour la surveillance et la poursuite et qui sont fondés sur des mécanismes à base de phéromones digitales.

2.6 L'organisation d'un SMA

Dans cette section, nous nous intéressons uniquement aux SMA basés sur des modèles centrés sur l'ensemble du système *OCMAS* (organization centered multi-agent systems), et non pas sur les systèmes centrés autour des caractéristiques individuelles de l'agent *ACMAS*(agent centered multi-agent system). L'un des principes fondamentaux du niveau organisationnel est de décrire le "quoi" et non le "comment" des choses, et ceci est réalisé en proposant une structure au niveau d'un patron d'activités d'agents pour répondre au "quoi", mais sans décrire "comment" les agents se comportent [20]. En effet, les auteurs Ferber et al. indiquent donc que l'organisation du système ne contient pas de "code" à exécuter par les agents, mais que celle-ci fournit des spécifications utilisant une sorte de normes ou de lois, des limites et des attentes qui sont intégrées au niveau comportemental des agents. Dès lors que les architectures internes de l'agent n'interfèrent pas au niveau organisationnel,cela implique qu'une organisation peut tout aussi bien inclure des agents réactifs ou cognitifs qui agiront au sein de l'organisation. Par conséquent, mise à part les descriptions comportementales de l'agent, le niveau organisationnel ne tient pas compte des concepts que nous avons vu plus haut tels que les croyances (beliefs), les désires, ou encore les intentions. Concernant le concept de but (goal), il s'agit de s'intéresser aux buts de l'organisation qui peuvent être des buts collectifs.

L'ensemble des modèles existants doivent ainsi tenir compte des caractéristiques clés que nous venons de citer ainsi que d'autres éléments qui seront spécifiques à certaines situations et dont il faut tenir compte durant la phase de conception du système.

Nous reprenons dans ce qui suit une description de deux modèles organisationnels issus de l'ingénierie des SMA, que nous avons considéré dans ce travail de recherche.

Cette sélection met en avant la diversité des aspects et concepts qu'il est possible de considérer au sein d'une organisation, et aussi les éléments minimales qui constituent l'organisation.

Bien qu'il existe d'autres modèles d'organisation que nous avons inclus durant notre travail d'analyse du domaine des SMA, tel que le modèle CRIO sur lequel s'appuie la méthodologie Aspecs [89](voir chapitre 5), nous nous sommes appuyés principalement sur les deux modèles qui suivent pour deux raisons : d'une part pour identifier les composants minimales de l'organisation car nous ne pouvons parler d'organisation d'un SMA sans parler des composants de base de cette dernière ; et d'autres part, pour avoir une vision plus étendue de l'organisation.

2.6.1 L'organisation selon AGR :

Dans ce contexte le modèle AGR [20] est un modèle organisationnel basé sur les trois concepts suivants : *Agent/Group/Role*, que nous retrouvons au sein du méta-modèle présenté

dans la figure 2.17.

D'une façon générale, l'agent représente l'entité active et communicante qui joue un ou plusieurs rôles au sein de groupes. Les rôles représentent les fonctions associées à l'agent au sein d'un groupe (partition de l'organisation). Un groupe définit les lieux de communication possibles entre les agents, tel que les agents faisant partie d'un même groupe partagent des caractéristiques communes. Bien que ce modèle permet d'une part de décrire l'organisation d'un système de façon générique indépendamment de l'architecture interne des agents, et d'autre part d'inclure des éléments indispensables au sein d'une organisation, ce modèle reste restreint car il n'inclut pas certains concepts qui pourraient être nécessaires comme le fait de prévoir des normes au sein de l'organisation en question.

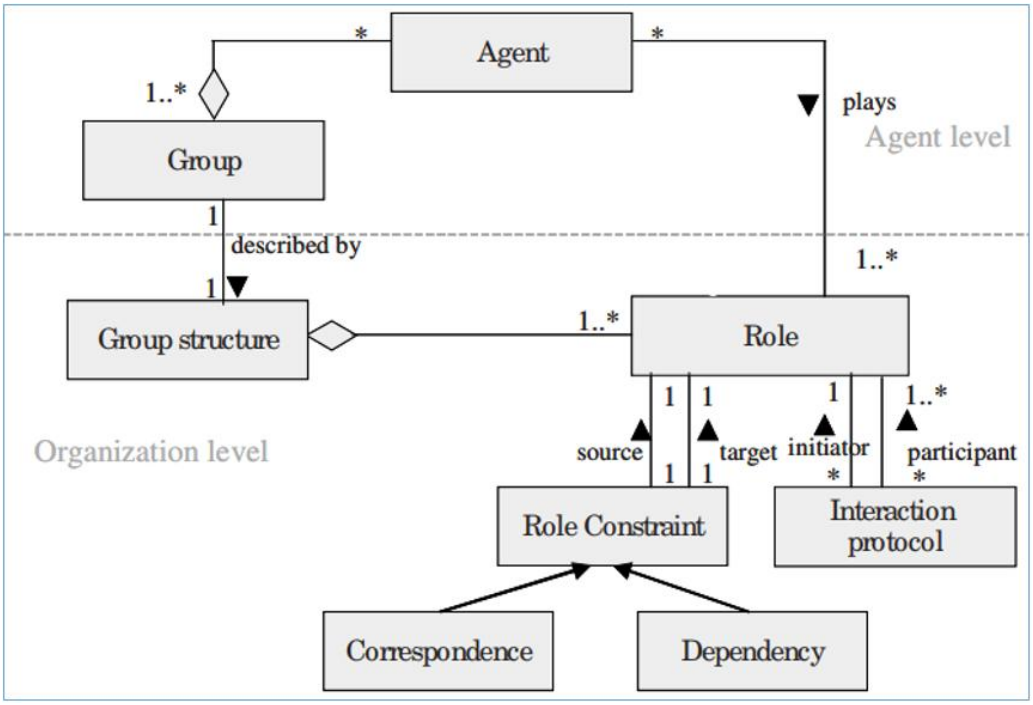


FIGURE 2.17 – Le méta-modèle d'AGR [20]

2.6.2 L'organisation selon Moise+ :

L'une des vues de l'organisation les plus intéressantes et étendues est celle proposée dans *Moise +* [21] où l'organisation est considérée selon trois points de vue complémentaires : *structural*, *fonctionnel* et *normatif*. Ce modèle apporte à l'organisation une dimension normative ou *déontique* en comparaison au modèle AGR que nous avons présenté juste avant. Moise+ permet ainsi de définir de façon très complète l'organisation à plusieurs niveaux. Notre travail de

recherche reprend d'ailleurs ce modèle comme ligne directrice pour l'analyse des autres caractéristiques des organisations multi-agents (voir chapitre 5). De façon plus précise, l'organisation est spécifiée à travers plusieurs niveaux comme suit : en premier, au niveau individuel grâce aux comportements d'agents qui sont autorisés pour chaque rôle ; en second, au niveau social par le biais des interactions entre les différents rôles ; et enfin au niveau collectif grâce aux structures résultantes des interactions possibles entre les rôles.

La figure 2.18 présente le méta-modèle de Moise+ qui montre comment l'organisation est décomposée en un ensemble de tâches à compléter (task) et qui sont elles-mêmes décomposées en parties appelées missions qui devront être assurées par les agents. Il est important de mentionner que pour chaque tâche, il faudra spécifier les actions, les ressources et les autorisations nécessaires.

Dans ce modèle, le groupe représente un ensemble de rôles et de missions permises dont l'exécution sera régulée par un but (goal) que l'agent devra atteindre.

Certes, ce modèle tout comme celui d'AGR [20] définit l'organisation de façon statique, cependant le présent travail de recherche n'inclut pas une étude sur les organisations réorganisées ou auto-organisées.

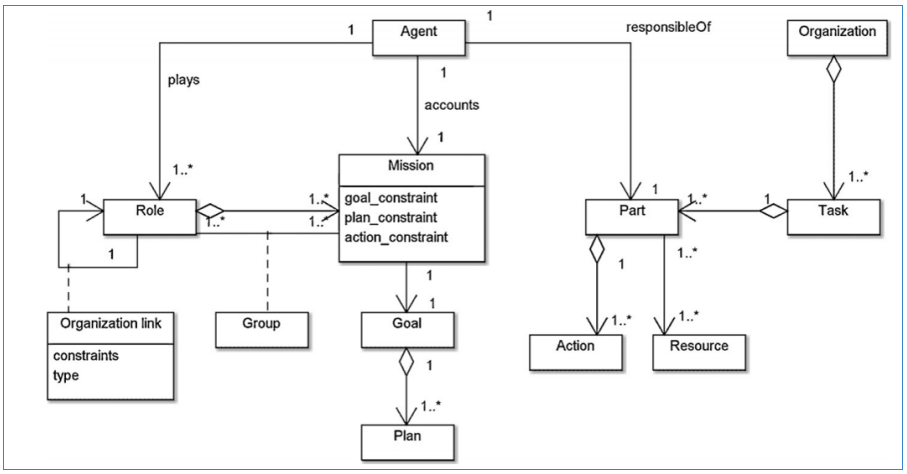


FIGURE 2.18 – Le méta-modèle de Moise+ [21]

2.7 Conclusion

Dans ce chapitre, nous avons présenté les principaux concepts d'un système multi-agents et les avons présentés en nous appuyant sur l'approche VOYELLES [28], qui est la méthode que nous avons choisie d'utiliser dans notre travail de recherche (voir chapitres 4 et 5).

Cette approche qui décrit un SMA par les dimensions voyelles (A "pour Agent", E "pour Environnement", I "pour Interaction", et O "pour Organisation"), nous a permis de couvrir les composants essentiels à la conception d'un système multi-agents, et d'approfondir le contenu de chacune des voyelles.

Pour la première voyelle A (Agent), nous avons présenté entre autres l'architecture interne délibérative BDI, ainsi que les types de perceptions d'agents existantes.

Pour la seconde voyelle I (Interaction), nous avons entre autres présenté les protocoles d'interactions utilisés lors d'interactions directes, ainsi qu'un ensemble de mécanismes pouvant être exploités durant des interactions indirectes entre agents.

Pour la troisième voyelle E (Environnement), nous avons présenté les éléments que nous retrouvons dans la plupart des modèles d'environnements, tels que les "ressources". Nous avons également présenté d'autres aspects de l'environnement d'agents comme la possibilité de le décrire par des caractéristiques spatiales durant la modélisation du système.

Concernant la dernière voyelle O (Organisation), nous avons présenté deux visions distinctes de l'organisation et qui sont celles du modèle AGR [20], et Moise+ [21]. Nous avons montré que certains éléments de l'organisation sont indispensables tels que les "rôles" qui doivent être identifiés au sein de tout type d'organisation. D'autres éléments vont dépendre du contexte comme le fait d'intégrer ou pas des normes au sein de l'organisation.

Après avoir abordé pour chacun des domaines de lignes de produits logiciels et de systèmes multi-agents les définitions, modèles et concepts les plus importants et pertinents dans le cadre de cette thèse, le prochain chapitre sera consacré à la partie état de l'art des travaux de recherche portant sur les approches d'ingénierie de lignes de produits multi-agents MAS-PL (Multi-Agent System Product Line). Ces approches sont issues de l'introduction des techniques d'ingénierie de lignes de produits logiciels au sein de l'ingénierie des SMA.

Troisième partie

État de l'art

Chapitre 3

Les approches d'ingénierie de lignes de produits multi-agents

3.1 Introduction

Le présent travail de recherche, rappelons-le, a pour objectif principal de faciliter le développement d'applications multi-agents similaires.

Ces versions ou variantes qui sont associées à un même domaine d'application, peuvent s'avérer être difficiles à implémenter. En effet, le simple passage d'une variante à l'autre peut engendrer des changements à divers niveaux, qui pour la plupart devront se faire manuellement (conception, code, etc.). Nous avons donc pensé à résoudre ce problème en minimisant l'écart existant entre les modifications à apporter et leur implémentation.

Pour cela, nous nous sommes intéressés à la branche d'ingénierie de lignes de produits multi-agents MAS-PL (Multi-agent system product line) ; où la ligne de produits représente une famille de SMA ou d'applications multi-agents similaires. Cette branche d'ingénierie MAS-PL est issue de l'ingénierie de lignes de produits logiciels que nous avons présentée en détail au niveau du chapitre 1, et qui a été appliquée à l'ingénierie des SMA afin de faciliter, et de supporter le développement de lignes de produits multi-agents.

Dans le présent chapitre, nous proposons de présenter en premier lieu les principales caractéristiques des pratiques de réutilisation mises en place dans les approches MAS-PL, en comparaison avec des pratiques de réutilisation plus classiques.

En second lieu, nous retraçons l'évolution des approches MAS-PL depuis leur émergence ; en donnant les avantages mais aussi les limites que ces approches ont rencontrés au fil du temps. En suite, nous donnons une classification des approches MAS-PL que nous avons réalisée sur la base de différents critères, tels que le degré de réutilisation lors du développement de chaque type d'approche ; ainsi que la méthode adoptée pour introduire des techniques de réutilisation de

LdPL (extensions, compositions, et adhoc). Nous présenterons quelques unes de ces approches de façon détaillée ; en indiquant pour chacune d'entre elles la phase d'ingénierie au niveau de laquelle elle intervient (ingénierie du domaine et/ou d'application) ; ainsi que les principales avantages et inconvénients.

Nous aborderons brièvement les travaux en relation uniquement avec la phase d'ingénierie d'application (outils de dérivation), car ils ne demeurent pas pertinents dans le cadre de notre travail de recherche.

Nous terminerons par une analyse et étude comparative des approches MAS-PL, sur la base d'un ensemble de critères choisis, en donnant les raisons de tels choix.

3.2 Principales caractéristiques des approches MAS-PL

Plusieurs travaux se sont préoccupés de la réutilisabilité dans le domaine des systèmes multi-agents. Ces approches peuvent intervenir à un ou plusieurs niveaux allant de la réutilisation des besoins à l'implémentation des SMA.

La réutilisation dans les systèmes multi-agents permet principalement d'optimiser le temps, le coût et l'effort de conception et/ou de développement d'un système sans démarrer de zéro.

L'objectif de cette section est de mettre en avant les principales caractéristiques des approches MAS-PL, qui font qu'elles se distinguent des autres pratiques de réutilisation :

1. **Réutilisation planifiée et non opportuniste**

   Cette caractéristique qui est typique des lignes de produits logiciels, et qui fait donc partie intégrante des approches MAS-PL, permet d'éliminer toute confusion susceptible d'avoir lieu avec d'autres pratiques de réutilisation.

   Afin de mieux comprendre cette caractéristique prenons pour exemple les travaux de recherche visant à réutiliser un code source, une conception, une spécification ou encore des jeux de tests. Dans ce contexte, les architectures modulaires d'agents sont flexibles et faciles à étendre tout en permettant une certaine aisance à réutiliser des composants multi-agents, comme la possibilité d'intégrer différentes librairies ou bibliothèques par le biais de composants réutilisables lors de l'implémentation d'un système multi-agents, facilitant ainsi l'activité du développeur, ainsi que le passage du modèle vers son implémentation. Ceci peut être réalisé en se basant sur des spécifications pouvant être transformées en implémentation de plateformes de développement multi-agents telles que Madkit [90] qui implémente l'un des premiers modèles organisationnels multi-agents existants [91] [20], ou encore Janus [92], qui correspond à l'implémentation des principes d'Aspecs [89] pour le développement de SMAH (Systèmes multi-agents holoniques).

   A l'inverse de ces pratiques de réutilisation dites *"opportunistes"* ; telles que les librairies

qui sont réutilisées si l'opportunité se présente ; la réutilisation en MAS-PL est "planifiée, forcée et donc activée". Tous les éléments nécessaires sont conçus pour être réutilisés et non dans l'espoir qu'ils le soient si l'occasion se présente, et sont optimisés pour être réutilisés dans plusieurs systèmes. Par exemple, l'approche MAS-PL proposée par Nunes et al. [93] prévoit de manière anticipée différents produits logiciels cibles, auxquels il est prévu d'ajouter un comportement autonome utilisant la technologie d'agent. Par conséquent, tous les composants permettant de construire des SMA dans ce contexte d'applications web seront forcément réutilisés. Il s'agit d'un développement "pour la réutilisation" (phase d'ingénierie du domaine).

2. **Ligne de produits multi-agents entant que bloc et non en tant que produits distincts**

   Lors du développement d'un nouveau système multi-agents qui semble similaire à un autre déjà construit préalablement ; il est possible de réutiliser, dans la mesure du possible, ce qui peut l'être à partir de celui déjà réalisé (voir la figure 3.1), et procéder ensuite aux modifications. Certes, cette façon de faire permet de gagner du temps, en apportant d'autres avantages économiques par la réutilisation d'une partie d'un autre système. Cependant, le résultat final sera l'obtention de deux systèmes entièrement différents mais pas deux systèmes construits à partir d'une base commune. Ceci est du au fait que les produits sont conçus, et développés un à un ou à la limite de façon parallèle par des intervenants différents.

   A l'inverse, dans les approches MAS-PL, les composants sont conçus explicitement pour une réutilisation. Par conséquent, la ligne de produit est traitée en tant que bloc et non de façon individuelle pour chaque système. Tous les systèmes multi-agents de la ligne de produits sont construits à partir de la même base commune.

   La figure 3.2 schématise ce principe de réutilisation pour les familles de SMA, en permettant aux ingénieurs d'utiliser un ensemble partagé d'artefacts (assets) qui représentent les artefacts associés aux différents systèmes multi-agents de la ligne de produits (besoins, spécifications, code source, configurations .etc). Les artefacts (assets) sont conçus de façon à pouvoir être partagés à travers la ligne de produits multi-agents.

   Dans l'approche MAS-PL GOSPEL [24] par exemple, dès le départ les composants réutilisables du domaine de l'ioT (internet of things) sont intégrés dans la définition du modèle de la ligne de produits multi-agents. Par conséquent, la première version du modèle supporte d'ores et déjà le développement de plusieurs versions similaires de SMA applicables au domaine de l'ioT. Ces derniers sont construits à partir d'une base commune d'artefacts relatifs à l'implémentation de ces variantes, et non obtenus par la réutilisation de façon individuelle.

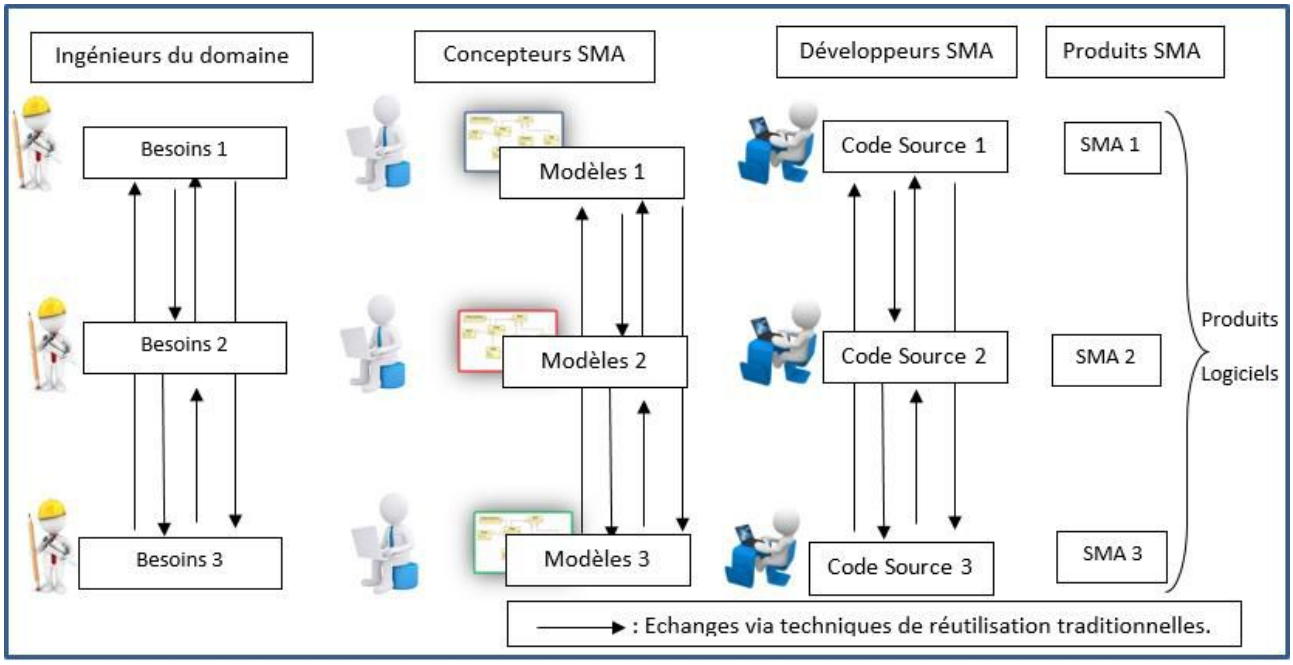


FIGURE 3.1 – Principe de réutilisation classique pour les systèmes multi-agents

3. **Facilité à remonter à la source d'une erreur au niveau du code source**

   Les pratiques de réutilisation illustrées dans la figure 3.1 reflètent que les intervenants dans les phases de réalisation d'un SMA communiquent par des échanges de requêtes de réutilisation d'un code source, de besoins, de spécification, de diagramme conceptuel etc. Cependant, ce processus présente un inconvénient majeur. Le nombre élevé d'échanges possibles peut par exemple conduire à des applications multi-agents défectueuses dans le cas de propagation d'une erreur. De plus, la communication et la coordination nécessaires au développement d'une ligne de produits multi-agents sera encore plus élevée. Supposons qu'une portion de code comporte une erreur, et que celle-ci soit réutilisée pour l'implémentation d'un autre produit multi-agents, et que ce même produit soit réutilisé à son tour, et ainsi de suite. Cela va engendrer une dégradation considérable de la qualité des produits multi-agents de la ligne. Sans oublier que la détection de l'origine de l'erreur restera difficile à réaliser.

   Une solution envisageable serait d'augmenter le nombre de concepteurs et de développeurs, mais cela reste insuffisant vu la complexité croissante des versions d'applications futures. Ce qui engendre une perte considérable en termes de temps, de coût et d'effort.

   A l'inverse, comme mentionné ci-dessus, l'ingénierie de lignes de produits logiciels offre un ensemble commun et partagé d'artefacts (assets) accessibles par les membres de l'équipe, et par conséquent les approches MAS-PL héritent de cette caractéristique.

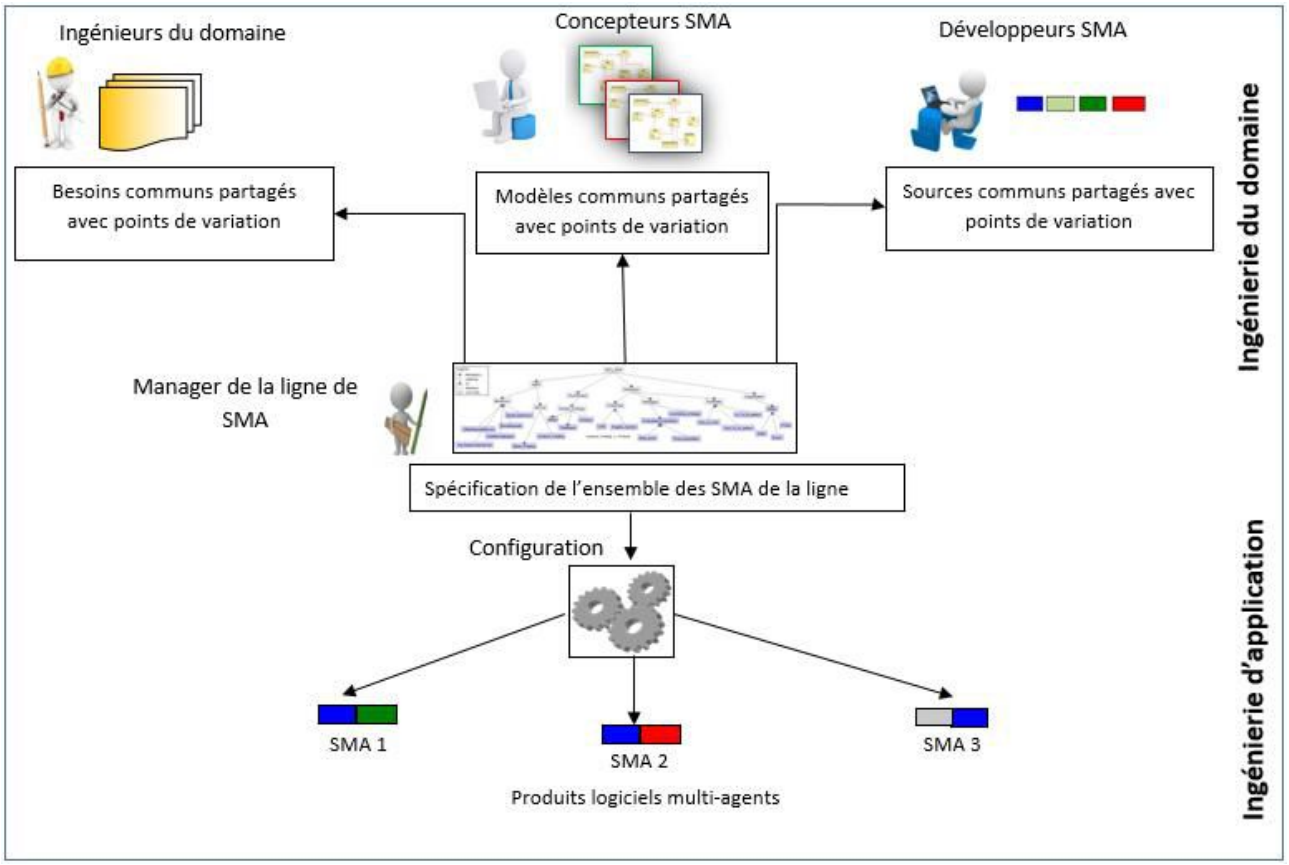


FIGURE 3.2 – Principe de réutilisation avec les techniques de lignes de produits logiciels pour les familles de systèmes multi-agents

L'accès est donné par un gestionnaire et en cas d'erreur, la source est facilement détectée. Comme nous l'avons vu dans le chapitre 1, l'ingénierie de lignes de produits logiciels permet d'intégrer le concept de points communs et variables dès les *premières phases de la conception* ; ce qui n'est pas le cas pour une approche classique.

4. **Construction des SMA à partir de caractéristiques communes et variables et non à partir de problèmes récurrents**

   Parmi l'une des pratiques de réutilisation les plus utilisées et connues du domaine et provenant de l'ingénierie logicielle est celle des patrons de conception *(design patterns)* qui s'inspirent de problèmes récurrents. Dans le domaine multi-agents plusieurs travaux de réutilisation s'articulent autour de ce principe de design patterns.

   Dans ce contexte, Lind et al. [94] proposent un des premiers travaux de réutilisation qui présente l'avantage de donner une structure possible d'un catalogue de patrons, tout en introduisant un schéma de description de patron qui répond à des besoins d'agent spécifique. Cependant, ces premières tentatives de réutilisation par le biais de patrons

de conception ne proposaient aucune directive accompagnant les développeurs dans leur démarche. Seules quelques approches ont été proposées par la suite pour la réutilisation au sein d'un environnement de développement orienté agents [95] [96].

L'approche MASORG–MAS Organisations [97] a été proposée pour supporter la réutilisation de systèmes multi-agents décrits de façon générique en se focalisant sur la réutilisation de rôles spécifiques au domaine d'application. Ces derniers ont séparé le comportement générique de l'agent de celui de l'application. L'inconvénient majeur de ces approches est que la majorité d'entre elles propose des patrons relatifs à la définition de protocoles uniquement, en négligeant des aspects relatifs aux architectures d'agents. De plus, certains patrons de conception proposés, s'articulent seulement autour des framework multi-agents. Dans ce contexte, Sabatucci et al. [98] proposent les patterns "GenericAgent", "Request", et "Query", qu'ils considèrent comme les plus utilisés dans la majorité des cas d'études. Plus encore, les détails relatifs à l'implémentation des patrons sont délégués au développeur, et dépendent ainsi de sa propre expérience.

Les aspects de leur approche pour décrire et composer ces designs patterns est très intéressante. Néanmoins, ils sont d'avantage utilisés pour la représentation de connaissances déclaratives, que pour servir de patron de conception.

D'autres travaux plus récents proposent un schéma de patrons de conception des modèles d'ontologie permettant la réutilisation des connaissances d'ontologies existantes pour le développement des SMA [99]. Cette proposition est très intéressante dès lors qu'elle permet l'identification des modèles d'ontologie pertinents au problème de conception en question, par le biais du schéma de classification des modèles. Ces modèles peuvent être ensuite réutilisés et partagés entre les communautés du domaine multi-agents pendant la phase de développement du système. Leurs résultats ont bien montré que cette proposition permet de réduire le temps de développement lorsqu'il s'agit d'applications spécifiques à un domaine en question. Le schéma qu'ils proposent facilite également la recherche et la réutilisation des modèles pendant l'analyse, la conception ainsi que le développement d'un SMA. Cependant, il aurait été intéressant de relier ce principe à un plus haut niveau d'abstraction en l'appliquant pour les ontologies indépendantes du domaine d'application.

Nous retrouvons également des travaux intéressants, s'appliquant à un niveau d'abstraction élévé, parmi eux les patrons de conception spécifiques à des agents négociateurs [100], qui étudient et reprennent entre autres les protocoles de négociations les plus utilisés.

A l'inverse de ces pratiques, les approches MAS-PL ne se basent pas sur une réutilisation des solutions données à des problèmes récurrents résultant de l'expertise des praticiens du domaine, mais plutôt sur une analyse des similitudes et de la variabilité du domaine

considéré. Cette analyse est faite par des experts du domaine, afin de mieux cibler les caractéristiques communes et donc réutilisables, mais aussi afin d'exploiter cette variabilité en tant que force et non en tant que contrainte. En effet, la variabilité devient une force au sein d'approches des lignes de produits de façon générale et des lignes de produits multi-agents de façon spécifique, car celle-ci est obligatoirement spécifiée et donc considérée dès la première activité du processus d'ingénierie de la ligne de produits (phase d'ingénierie du domaine). Par la suite, la variabilité sera exploitée pendant la phase d'ingénierie d'application, où chaque version du système est obtenue "par la réutilisation" (dérivation automatique), en exploitant la variabilité (multitude de choix possibles pendant une configuration).

A titre d'exemple, l'approche GAIA-PL proposée par Dehlinger et al. [101] est une approche MAS-PL qui fournit un patron de spécification des besoins qui permet de capturer les changements de configuration (points de variation), ainsi que les potentielles réutilisations possibles des spécifications de besoins. Suite à leur analyse des caractéristiques communes et variables des spécifications des besoins, ils ont réussi à intégrer les différents points de variation d'agents associés aux protocoles, activités, permissions et responsabilités.

Leurs résultats ont montré un gain de temps considérable (48 pour cent) lors de la conception et la documentation. Quant au pourcentage des spécifications réutilisables capturées, il est de 36 pour cent sur un ensemble de 160 agents.

Le principe de réutilisation qu'ils appliquent n'est pas basée sur l'expérience des praticiens du domaine, mais plutôt sur l'analyse des caractéristiques communes et variables durant l'analyse et la spécification des besoins. Le nombre de produits de la ligne augmente au fur et à mesure que le nombre de caractéristiques variables augmente ; et cela grâce au fait que le nombre de combinaisons possibles entre les caractéristiques communes et variables augmente lors de la configuration du produit multi-agents à dériver. Ceci est valable à condition que la validité de la configuration soit assurée par les contraintes nécessaires à intégrer au sein du modèle.

5. **Plus qu'une architecture reconfigurable**

   Un avant dernier point important, est que les architectures de référence et les frameworks du domaine multi-agents existants sont conçus pour être réutilisés par des systèmes multi-agents multiples et à être *reconfigurées si nécessaire*. Par conséquent, ces solutions permettant de s'adapter aux familles de systèmes multi-agents même si elles n'utilisent pas de techniques provenant de l'ingénierie de lignes de produits logiciels.

   La réutilisation de ces structures architecturales n'est certes pas une première dans le domaine de la réutilisation pour les SMA, et reste une bonne idée car l'architecture est un élément central de tout système et un élément coûteux à construire. D'ailleurs,

beaucoup de travaux sur les systèmes multi-agents visent à la réutilisation de modèles et de briques logicielles en s'appuyant sur le principe *"Don't Repeat Yourself" (DRY)* en association avec la *Separation Of Concerns (SOC)*. Ceci a été réalisé en analysant d'éventuelles possibilités d'isoler de façon explicite des *MAS-concerns* par le biais d'abstractions orientées objet. Par exemple, Garcia et al. [102] ont réalisé une étude empirique qui évalue dans quelle mesure les abstractions associées d'une part aux abstractions OO (telles que les classes et objets), et d'autres part aux designs patterns dont nous avons parlé précédemment, permettent la modularisation des aspects d'un SMA. Leurs résultats ont montré que l'utilisation d'une conception et programmations par *Aspects* permettait de construire un SMA avec une meilleure modularisation et une diminution de l'effort de développement ; en sus d'une réduction dans le nombre de composants conceptuels présentant un degré de couplage réduit de façon considérable.

Même si de premier abord, l'architecture et sa reconfiguration semblent similaires lorsqu'il s'agit d'approches de lignes de produits multi-agents ; il y-a pourtant une différence majeure. En effet, en MAS-PL, l'architecture est conçue pour *prendre en charge la variation* requise par les autres produits multi-agents de la ligne, il est donc logique de la reconfigurer.

A titre d'exemple, l'approche MAS-PL proposée par Peña et al. [23] permet la compréhension, la description et l'analyse de systèmes évolutifs. Leur approche est basée sur le cœur de l'architecture de la ligne de produits multi-agents qui prend en charge les variations. Le système évolue donc obligatoirement par le biais de reconfigurations, et par conséquent de changements effectués au niveau des caractéristiques (features) incluses hors du cœur architectural.

Ainsi, un nouveau produit multi-agents de la ligne est associé à chaque état du système qui évolue dans le temps, mais qui reprend toujours le coeur de l'architecture qui a été conçue et qui est capable de supporter tous ces changements.

3.3 Évolution des approches MAS-PL

Dans cette section nous retraçons l'évolution, au fil du temps, des approches MAS-PL depuis leur apparition :

— **Avant 2005 :**

A cette période, les approches de lignes de produits multi-agents n'avaient pas encore vu le jour. Bien que les travaux sur la réutilisation qui les précédent n'entrent pas dans le cadre de notre travail de recherche, nous avons cité dans la section précédente quelques exemples reprenant les aspects les plus importants de l'ingénierie logicielle. Certaines pratiques sont encore utilisées de nos jours, tels que les patrons de conception qui pro-

viennent de solutions apportées à des problèmes récurrents. Ces derniers présentent bien des avantages, comme le fait de pouvoir être appliqués à des systèmes différents indépendamment du domaine d'application. Cependant, ils n'incluent pas des mécanismes clairement définis pour la gestion des similitudes et de la variabilité comme expliqué plus haut.

— **En 2005 :**
Durant cette année, les premiers efforts concrets en lignes de produits multi-agents MAS-PL ont vu le jour.
Ces premiers travaux de recherche ont tenté de capturer le potentiel de réutilisation des systèmes multi-agents dans le cadre de l'analyse des besoins et des spécifications d'agents [25]. Cependant, certains travaux étaient encore imprégnés de techniques qui n'étaient pas en relation directe avec les lignes de produits logiciels. Par exemple, il a été démontré que les "design patterns" pouvaient être réutilisés pour des systèmes orientés agents [94]. Les travaux réalisés par Dehlinger et al. [103] étaient similaires à un précédent travail qu'ils avaient réalisé [25] mais au niveau duquel le template de spécification des besoins était plus détaillé afin de capturer d'avantages de connaissances du domaine d'application.

— **De 2006 à 2008 :**
A cette période, les travaux de recherche étaient avantageusement imprégnés des ligne de produits logiciels. Dans ce contexte, Peña et al. ont commencé à utiliser des modèles en relation directe avec les lignes de produits logiciels [104]. Par exemple, ils ont adapté des modèles de similitudes et de variabilité en représentant les buts (Goals) des agents sous formes de caractéristiques (features) au niveau du modèle de caractéristiques (feature model). Même si certaines de leurs approches se focalisaient plus sur les parties communes que la variabilité, ils ont néanmoins rendu possible la construction du corps de l'architecture de la ligne de produit multi-agents capable de supporter la variabilité. Entre 2007 et 2008, les méthodes multi-agents telle que GAIA, PASSI et MaCMAS ont été étendues par le biais de mécanismes plus ou moins différents [105] [23] [106]. De plus, les approches MAS-PL sont devenues de plus en plus adaptées aux besoin des lignes de produits multi-agents en s'inscrivant dans le cadre du "Situational Method Engineering (SME)". Dans ce contexte, une approche intéressante [23] a repris des fragments de la méthodologie MaCMAS tout en documentant les besoins orientés But (Goal), des modèles de rôles et des diagrammes de traçabilité afin de construire une première version du modèle de système, pour ensuite y intégrer des aspects liés à la réutilisation des points communs et la gestion de la variabilité.

— **De 2009 à 2010 :**
Au fil des années, nous remarquons l'introduction de plus en plus d'aspects provenant

de l'ingénierie de lignes de produits logiciels tel que l'exploitation de stéréotypes réalisée principalement par la méthode PLUS [107].
De plus, les chercheurs ont commencé à s'intéresser à la granularité des caractéristiques. Dans ce contexte, Nunes et al. [108] ont montré qu'une granularité fine (croyances, buts et plans des agents) est essentielle pour l'extraction des caractéristiques d'un SMA ; car cela permet entre autres de réduire les réplications de code source ou encore d'améliorer sa lisibilité.
Beaucoup de travaux ont porté également sur des outils de dérivation (phase d'ingénierie d'application), comme l'outil GenArch proposé par Cirilo et al. [109], et qui permet l'instanciation et la personnalisation pour une dérivation automatique des lignes de produits multi-agents. Ces travaux ne sont cependant pas pertinents dans le cadre du présent travail de recherche.

— **En 2011 :**
Durant cette année, beaucoup d'approches de lignes de produits multi-agents ont été proposées ou encore mises à jour en y introduisant plus de détails concernant l'implémentation de la ligne de produits multi-agents. Comme résultat obtenu, une couverture totale du cycle de vie de développement de la ligne de produits multi-agents. Cependant, ces approches ne donnent toujours pas suffisamment de détails liés aux implémentations. Par exemple, bien que PASSI-PL [93] couvre toutes les étapes de l'analyse des besoins à l'implémentation ; la correspondance (mapping) entre les caractéristiques et leurs implémentations réutilisables n'est pas expliquée. De plus, la seconde phase du processus d'ingénierie couvrant entre autres la définition des besoins d'utilisateurs n'est pas détaillée. Pour finir, aucune indication n'est donnée pour cibler une sélection de caractéristiques optimale et adéquate afin d'obtenir une configuration valide.

— **De 2012 à nos jours :**
Bien que les approchent MAS-PL apparues à cette période proposent de donner plus de détails sur l'implémention, elles restent limitées à des domaines bien précis. En effet, nous avons constaté que la spécification de la variabilité est spécifique au domaine d'application, et de ce fait aucune réutilisation des artefacts ou de modèles n'est possible indépendamment du domaine d'application. Entre 2015 et 2019, les travaux qui ont suivi ont été étendus à d'autres domaines tels que les approches proposées pour l'internet d'objets ou Internet of Things (IoT) [110], mais également pour les systèmes ambients intelligents AmI(Ambient Intelligent) et plus précisément les systèmes AAL (Ambient Assisted Living) [111]en introduisant des lignes de produits (SPL) dynamiques pour l'adaptation du comportement d'agents durant son exécution (runtime). De plus, d'autres types de modèles [24] ont été combinés aux modèles de similitudes et de variabilité, tels que les modèles Goal-driven basés sur les buts d'agents et ce afin d'améliorer

la sélection des caractéristiques en fonction des besoins des différents utilisateurs, en tenant compte de l'ensemble des variations possibles des buts d'agents dès les premières étapes de la modélisation de la ligne de produits multi-agents.

— **Synthèse :**

Le tableau 3.1 présente un récapitulatif des points les plus importants à retenir sur l'apparition et l'évolution des approches MAS-PL. Ce récapitulatif est constitué d'une description générale ainsi que des limites rencontrées par les approches à chaque période :

Période	Descriptions	Limites
Avant 2005	Design patterns ; SOC (Separation of Concerns) tel que l'utilisation d'Aspects (AOP).	Aucune prise en compte de la variabilité ; Techniques de réutilisation pour la construction d'un seul système multi-agents ou de systèmes distincts.
2005	Extensions des méthodes multi-agents ; Développement de familles de SMA ; Approches construites de zéro.	Pas de documentation des similitudes et de la variabilité (pas de Feature Model) ; Aucune capture des changements entre les versions (différences) ; Aucune contrainte ni contrôle sur la validité de sélection de rôles dans le système.
De 2006 à 2008	Compositions d'approches ; Spécification de variabilité avec un FM ; Utilisation d'outils de vérification de configurations valides.	Exploitation restreinte des techniques LdP ; Focus sur l'évolution du système et non sur l'analyse initiale pour la spécification de la variabilité dès le départ ; Variabilité générique non exploitée dans le FM.
De 2009 à 2010	Exploitation des méthodes LdP (PLUS, FAST) ; Stéréotypes pour la variabilité ; Nouvelles Extensions plus riches.	Approches annotatrices lourdes ; Approches ne couvrant pas l'ensemble des étapes du cycle de développement.
2011	Couverture totale du développement des MAS-PL ; Quelques détails d'implémentation MAS-PL.	Pas assez de détails sur l'implémentation ; Aucun association (mapping) entre les caractéristiques et leurs implémentations.
De 2012 à 2020	Approches pour ioT et AAL ; Introduction des lignes de produits dynamiques pour l'adaptation du comportement d'agent durant le l'exécution (runtime) ; Combinaisons des modèles de variabilité avec des modèles dirigés par les buts (Goal-driven) ; Variabilité spécifiée par le CVL (Commonality and Variability Language)	Spécification d'une variabilité relative au domaine d'application ; Aucune réutilisation possible du modèle de similitudes et de variabilité pour un autre domaine ; Aucune réutilisation possible des artefacts d'implémentation indépendamment du domaine.

TABLE 3.1 – Évolution temporelle des approches MAS-PL

3.4 Classification des approches MAS-PL

Bien qu'elle ne regroupe pas toutes les approches multi-agents existantes, la figure 3.3 permet de donner une vue d'ensemble des méthodologies orientées agents, en montrant où viennent se greffer les approches multi-agents à base de LdP aux approches conçues pour le développement d'une seule application multi-agents.

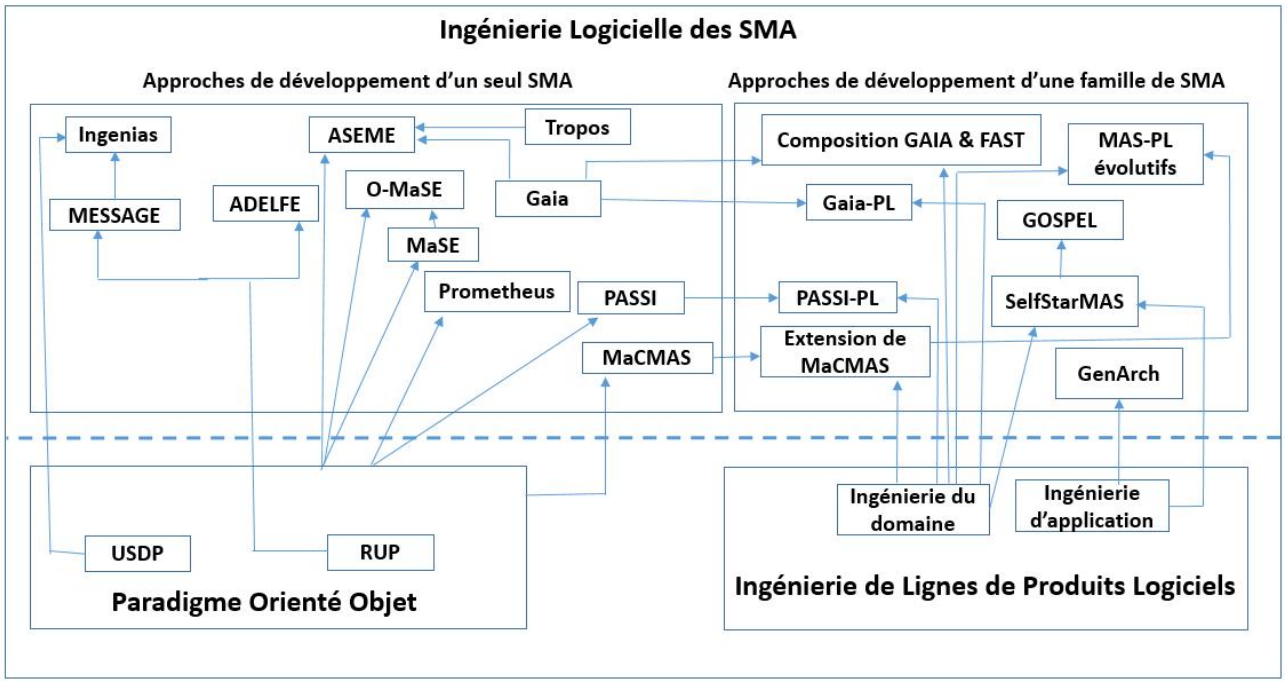


FIGURE 3.3 – Positionnement des approches MAS-PL dans l'ingénierie multi-agents

Cette figure est issue d'une adaptation de [112] à laquelle nous avons intégré quelques approches MAS-PL dont nous avons évoqué les grandes lignes plus haut.

En analysant l'ensemble de ces approches, nous avons constaté qu'il était possible de distinguer trois grandes catégories : Les *extensions*, les *compositions* et les approches construites de zéro ou *adhoc*.

Cette classification a été réalisée en fonction du degré de réutilisation, et de la pratique adoptée pour l'introduction de principes d'ingénierie de LdP logiciels au sein de l'ingénierie des familles de SMA. Dans ce qui suit, nous donnons une description détaillée de chacune de ces classes d'approches accompagnée de quelques exemples :

3.4.1 Extensions de méthodologies multi-agents

La majorité des approches MAS-PLs se placent dans cette catégorie. Cette classe d'approches propose des extensions de méthodologies multi-agents par des aspects provenant des LdP (Lignes de Produits logiciels). Ces travaux ont étendu les méthodologies multi-agents PASSI [76,113] , GAIA [67,114,115], et MaCMAS [116]. Nous présentons et analysons quelques-unes d'entre elles dans ce qui suit :

3.4.1.1 L'extension PASSI-PL

1. **Description**

   Les premiers travaux de Nunes et al. [22] ont proposé l'approche PASSI-PL (PASSI Product Line) qui étend la méthodologie multi-agents PASSI [76], en y introduisant une nouvelle phase préliminaire. Cette nouvelle phase modélise les caractéristiques (features) relativement au domaine d'application. Le reste de leur approche consiste à étendre les notations utilisées dans PASSI, comme l'ajout de stéréotypes au sein des diagrammes UML utilisés dans le but d'indiquer les abstractions et composants variables du système.

   La figure 3.4, représente un exemple de la façon avec laquelle l'extension a été réalisée. Le diagramme d'identification d'agents utilisé habituellement dans la méthodologie PASSI [76], a été étendu par le biais de nouveaux stéréotypes pour l'annotation de la variabilité, qui est spécifiée sous forme d'*options*, ou encore de *choix alternatifs* à faire entre les fonctionnalités du SMA. Dans cet exemple associé au cas d'étude OLIS (OnLine Intelligent Services), qui est une application capable de fournir différents services aux utilisateurs ; la fonctionalité "Remind event" associée à l'egent est optionnelle. Tandis que le choix entre les fonctionnalités "check interest in event academic" et "check interest in event travel" de l'agent, est un choix alternatif.

   Les parties réutilisables proposées dans cette approche sont détectables une fois que le modèle évolue. Les configurations possibles d'agents reflètent tous les points de variations à prendre en considération lors de la modélisation de l'ensemble des produits de la ligne.

   Les derniers travaux de Nunes et al. [93] apportent de nouveaux aspects à l'extension de la méthodologie PASSI [76], en proposant un processus d'ingénierie du domaine pour leur développement. Ce processus est doté de notations (UML et MAS-ML [117]) afin de modéliser et documenter la variabilité d'agents. Ils utilisent également la méthode PLUS [107] qui emploie des stéréotypes afin de spécifier et de permettre une traçabilité de la variabilité présentant ainsi l'un des points forts de l'approche. Leur approche est la seule extension qui couvre l'ensemble des phases d'ingénierie multi-agents allant de la conception à l'implémentation.

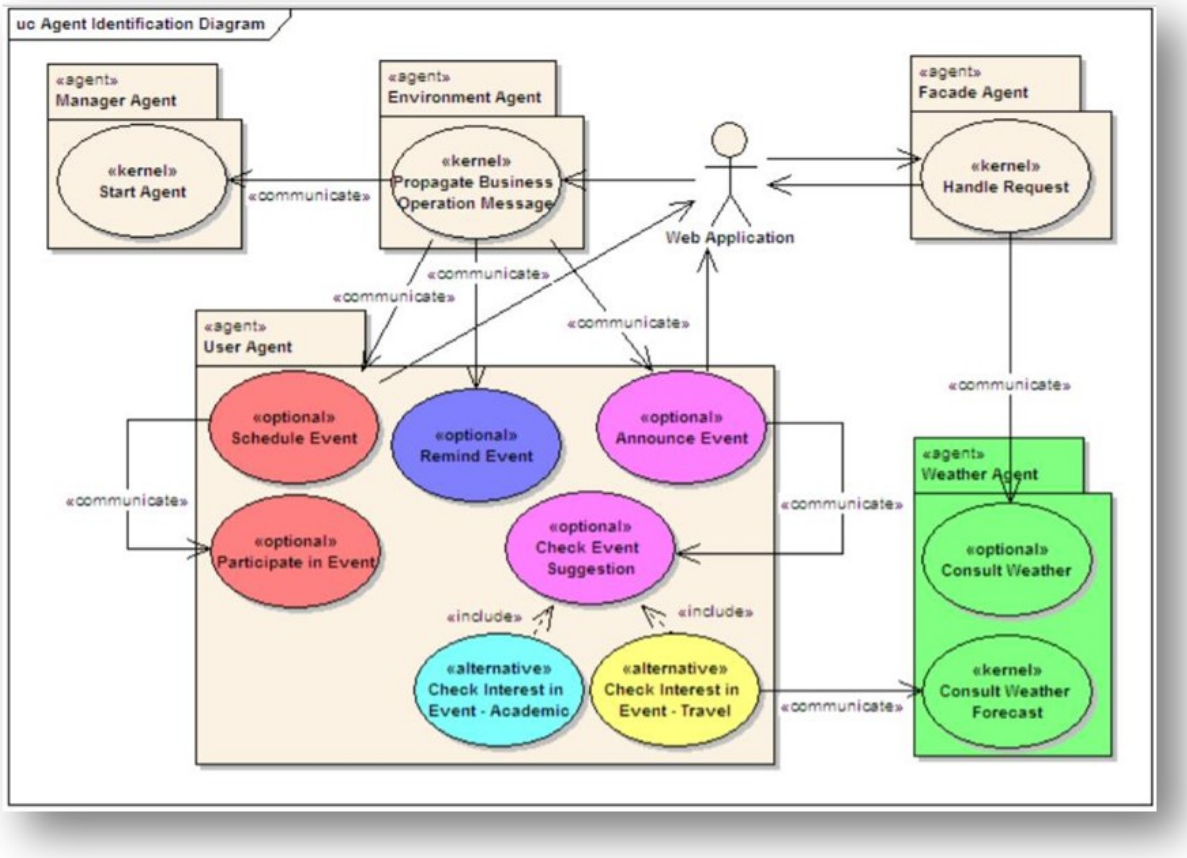


FIGURE 3.4 – Exemple d'extension du diagramme d'identification d'agents dans PASSI-PL [22]

2. **Avantages et inconvénients**

   Un des avantages des premiers travaux sur l'extension PASSI-PL [22], est qu'elle offre un patron de spécification des besoins réutilisable indépendamment du domaine d'application. Le second avantage est qu'un diagramme préliminaire à toute étape est établi afin de spécifier la variabilité liée au domaine d'application. Cependant, aucune variabilité indépendante du domaine d'application n'est spécifiée. Il aurait été intéressant d'établir une toute autre analyse afin que les abstractions relatives aux systèmes n'apparaissent pas ultérieurement de façon non regroupée tel qu'il est proposé dans l'approche.

   Un autre avantage est que lors de l'identification de rôles, l'idée d'utilisation de fragments UML pour croiser les caractéristiques (features) est intéressante. Cependant, nous pensons que l'utilisation de diverses couleurs et notations pour la spécification de la variabilité des systèmes, peut devenir assez lourde et rendre les modèles difficiles à maintenir lors d'un développement incrémental ou encore dans le cas où le système comporterait

un grand nombre de diagrammes conceptuels.

Le cas d'utilisation qu'ils proposent permet le développement d'une ligne d'applications web comportant des services Web variables et donc de façon spécifique à ce domaine. La particularité de leur proposition est qu'au départ lors de la conception et implémentation de la première variante du système, celle-ci ne comportant aucun service intelligent et donc aucun comportement autonome du système. Hors lors de la construction d'une ligne de produits multi-agents, la première variante doit être un système multi-agents. Du moins, la variante comportant les caractéristiques basiques devrait être un SMA. Ce n'est qu'à partir de la seconde variante du système que le modèle est raffiné par l'introduction de la technologie d'agents ; et donc de services intelligents. Nous pensons que la version la plus simple inclue dans la ligne et qui va représenter le cœur de la famille multi-agents devrait inclure au moins un agent minimaliste afin d'assurer le service intelligent minimal.

Pour finir, nous constatons que bien que les derniers travaux sur l'extension PASSI-PL [93] proposent des directives pour le développement, ils n'en contiennent pas suffisamment sur la manière d'implémenter les SMA, ni sur la façon de les générer jusqu'à pouvoir les déployer. De plus, aucune indication n'est donnée sur le type d'implémentation choisie (approche annotatrice, compositionnelle ou combinatoire) ; ce qui peut rendre difficile l'adoption de l'approche pour la communauté multi-agents ; à moins que les développeurs et experts en SMA soient également experts en lignes de produits logiciels.

3.4.1.2 L'extension GAIA-PL

1. **Description**

   Dehlinger et al. [105] ont proposé GAIA-PL (GAIA Product Line), qui étend la méthodologie multi-agents GAIA [114].

   Leur approche se focalise sur la documentation ainsi que la réutilisation de spécification de besoins pour un MAS-PL. Ils utilisent des techniques de lignes de produits pour réaliser le transfert des rôles d'agents dans la méthodologie GAIA à des rôles réutilisables au niveau du code source.

   Dans leur dernière proposition [101], les auteurs intègrent leur approche à la méthode FAST [31], qui divise le processus d'une ligne de produit en trois sous-processus : qualification du domaine, ingénierie du domaine et ingénierie d'application.

   Cette approche lignes de produits logiciels est basée sur l'investigation pro-active des ressources [31] ; et propose de diviser la conception et le développement d'une ligne de produits selon les deux phases d'ingénierie du domaine et d'application tel qu'ils ont été

définis dans le framework général du SPLE [3].

2. **Avantages et inconvénients**

   Leur principale contribution est de fournir un patron (pattern) de spécification des besoins pour capturer les changements de configuration (points de variation), ainsi que les potentielles réutilisations possibles des spécification de besoins. La spécification des besoins qu'ils proposent de réaliser, permet de faciliter leur réutilisation. Une autre contribution, est d'avoir intégré les différents points de variation d'agents en capturant les protocoles, les activités, les permissions et les responsabilités au sein du système multi-agents. Bien que ces points de variations soient des concepts génériques aux systèmes multi-agents, ils restent spécifiques aux rôles d'agents. De plus, malgré un gain de temps considérable (48 pour cent) lors de la conception et la documentation, le pourcentage des spécifications réutilisables capturées est de 36 pour cent sur un ensemble de 160 agents. Un autre inconvénient est qu'aucun détail n'est fourni concernant l'activité d'implémentation, ni comment les éléments réutilisables qu'ils proposent sont transformés au niveau du code source.

3.4.1.3 L'extension de MaCMAS

1. **Description**

   Peña et al. [23] ont proposé une approche MAS-PL qui étend la méthodologie MaCMAS (Methodology for analysing Complex Multiagent Systems) [116], qui est un fragment de méthodologie d'analyse des systèmes multi-agents complexes. Leur approche fournit un support pour les lignes de produits multi-agents.

   La figure 3.5, représente toutes les étapes proposées par l'approche. La première étape est celle de la conception d'un plan d'évolution qui illustre bien l'extension de la méthodologie MaCMAS, et qui doit représenter tous les produits multi-agents, incluant les caractéristiques qui devraient être ajoutées ou retirées afin de passer d'un produit à l'autre.

   En d'autres termes, chaque état du plan de variation va représenter un produit, et chaque transition représente l'ensemble des features à ajouter au produit pour le passage d'une variante à l'autre.

   A partir de cette information, les modèles de rôles et de plans seront composés ou décomposés, ainsi que les modèles de plans correspondants à chaque fonctionnalité du produit actuel ; afin d'obtenir la partie du modèle correspondante à l'organisation "acquaintance organization". La dernière étape de l'approche consiste à déployer les rôles dans l'organisation de la connaissance du produit sur les agents du système.

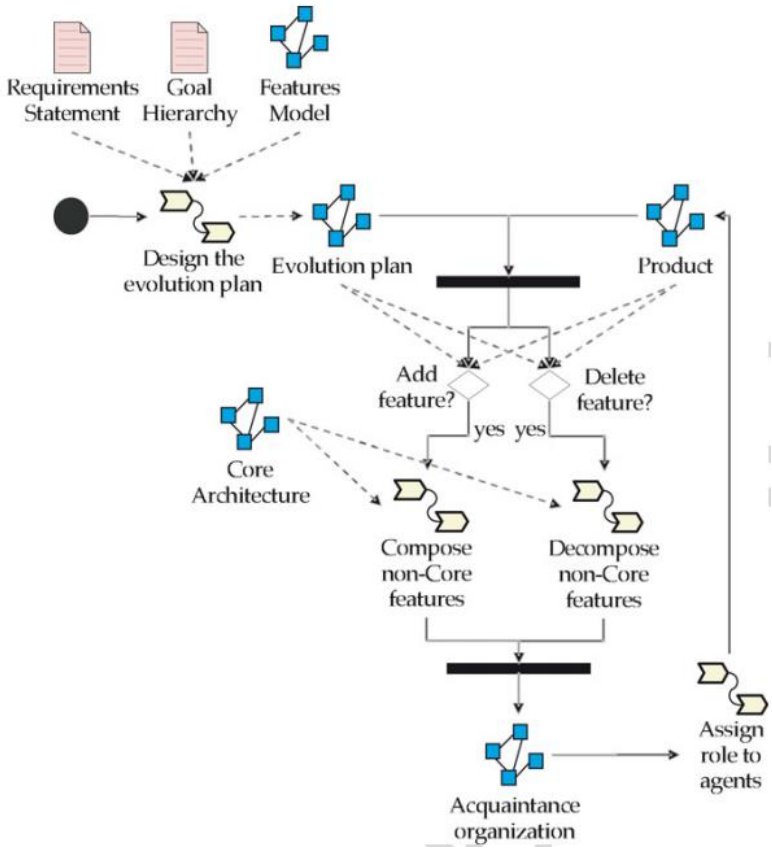


FIGURE 3.5 – Les étapes de l'approche MAS-PL associée à l'extension de MaCMAS [23]

2. **Avantages et inconvénients**

   La premier avantage de l'approche proposée est qu'elle permet la compréhension, la description et l'analyse de systèmes évolutifs.

   Un second avantage est que contrairement à un précédent travail (présenté ci-dessous dans la catégorie d'approches adhoc) [104], celle -ci [23] ne se base pas seulement sur le cœur (similitude) de l'architecture de la ligne de produits multi-agents qui représente les caractéristiques communes entre les produits de la ligne ; mais également sur les caractéristiques non communes représentant la variabilité. Le système évolue par le biais de changements sur les caractéristiques incluses dans l'ensemble des caractéristiques n'appartenant pas à celles du cœur architectural. A chaque état du système qui évolue dans le temps est associé un produit multi-agent de la ligne.

   Nous pensons également que l'idée de représenter les différents états du système est intéressante. Cependant, l'ajout et/ou suppression de caractéristiques qui déclenchent un changement d'état, est insuffisant pour inclure toutes les variantes possibles, sachant que d'autres événements déclencheurs pourraient mener à des variantes similaires susceptibles

de représenter d'autres points variables.

L'extension du modèle d'évolution de plan peut prédire partiellement les produits de la ligne. Tandis que l'utilisation et l'intégration de ces caractéristiques au sein du modèle de caractéristiques, permettrait de détecter des combinaisons supplémentaires de caractéristiques de façon plus adéquate et plus facile. Certes les conditions de transitions liées à l'environnement sont incluses mais des aspects variables relatifs à l'organisation ou aux interactions du système demeurent inchangées et sont considérées comme non variables lors de l'évolution du système. Ce qui exclue des variantes supplémentaires de la ligne de produits qui ne figurent donc pas au sein du plan de l'évolution du système. Un dernier inconvénient de l'approche est que le plan d'évolution du système est établi une seule fois au départ et leur approche ne permet pas son raffinement. Seuls les rôles du système sont raffinés avant d'être composés ou assignés aux agents afin d'obtenir un nouveau produit. L'émergence de nouveaux rôles apparaît lors de la composition de modèles de rôles qui ne sont pas indépendants. Cependant, leur répartition à travers plusieurs modèles de rôles composés, ainsi que les plans relatifs, qui seront également composés, peut s'avérer assez lourde dans le cas d'un nombre de modèles de rôles élevé. Le modèle composant les plans qui servira à représenter l'ordre d'exécution des modèles de rôles va également dépendre du nombre de nouveaux rôles émergents lors de l'évolution du système ; ce qui rend l'utilisation de l'approche lourde dans le cas d'une ligne de produits comportant un plus grand nombre que la ligne qu'ils utilisent comme cas d'étude de leur approche.

De plus, le fait de devoir exclure des caractéristiques du modèle afin de retourner à des produits précédents se fait par une décomposition de modèles, qui combinés à des notations UML ne capturant pas explicitement les caractéristiques d'agents par une modélisation séparée ; mais plutôt combinée rend l'approche difficile à comprendre.

Un des avantages de leur proposition, est qu'elle donne des directives quant à la capture et la réutilisation du cœur architectural de la famille de systèmes multi-agents, qui représente les similitudes entre les membres de la ligne de produits. De plus, le processus proposé au sein de l'approche est pris en charge par la plateforme d'implémentation.

Cependant, parmi les inconvénients majeurs soulevés, existe le fait qu'il n'y a pas d'association (mapping) directe entre la couche d'implémentation et le modèle de caractéristiques FM. De plus, aucun détail n'est donné quant à l'implémentation de la lignes de produits multi-agents. Nous remarquons aussi, que bien que le cœur architectural ait été construit et validé, aucune description de la variabilité n'est donné par l'approche qui se focalise principalement sur les similitudes.

Nous constatons également que l'approche en question peut induire, une mauvaise modularité des caractéristiques d'agents. Nous pensons que ceci peut être du à l'analyse de

la variabilité qui est réalisée non pas avant la modélisation du système, mais après. Le contraire tel que préconisé en ingénierie LdP, aurait permis une meilleure spécification des caractéristiques d'agents.

3.4.2 Compositions d'approches multi-agents

Ces approches s'inscrivent dans le cadre du SME (Situational Method Engineering), et reprennent donc plusieurs approches ou portions d'approches (multi-agents et lignes de produits confondues) et les composent. Une fois ces portions d'approches composées, elles peuvent être représentées par exemple sous forme de pattern réutilisables.

3.4.2.1 L'approche GOSPEL

1. **Description**

   L'approche la plus récente nommée GOSPEL (Goal-oriented Software Product Lines) [24] ne s'est pas contentée de couvrir les deux phases d'ingénierie des lignes de produits logiciels, mais a aussi combiné les modèles issus du processus de développement avec les modèles Goal-driven orientés buts. L'objectif de cette combinaison est de montrer comment les caractéristiques d'une variante d'un produit multi-agents peuvent assurer la correspondance avec les besoins des utilisateurs du système généré ; et par conséquent comment les buts (goals) peuvent être utilisés pour conduire et améliorer la sélection des caractéristiques de façon pertinente.

   Cette approche utilise en tout deux types de modèles : istar qui gère la variabilité des buts (Goals), et CVL (Commonality and Variability Language) pour spécifier les similitudes et la variabilité de la ligne de produits. Les algorithmes proposés par les auteurs propagent toutes les variations possibles au sein de la ligne à l'ensemble des agents logiciels.

   L'approche GOSPEL s'appuie également sur la réutilisation des agents issus de l'approche Self-StarMAS [110] que nous présentons ci-après.

2. **Avantages et inconvénients**

   L'approche GOSPEL offre de nombreux avantages. Nous pensons que le fait d'inclure un modèle de buts (Goal-model) au niveau du cycle de vie de la ligne de produits multi-agents est une idée intéressante, car ce type de modèles fait référence à une famille de modèles de buts. Ainsi la variabilité associée aux caractéristiques reliées est orientée vers les buts d'agents (goals), et est supportée par l'approche grâce au langage spécifique GRL (Goal Requierements Language).

   Un autre avantage de l'approche est qu'elle permet de gérer un ensemble considérable de modèles istar. Cependant, nous pensons que ces changements au niveau des buts

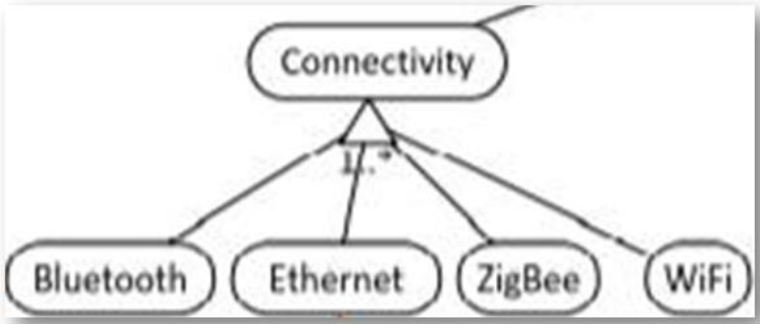


FIGURE 3.6 – Un extrait du modèle de variabilité de la connectivité spécifiée par l'approche GOSPEL [24]

pourraient être optimisés en étant inclus au niveau du modèle de similitudes et de variabilité. Par exemple, les modèles istar traitent des concepts d'agents réutilisables tels que les rôles, tâches, buts et ressources qui peuvent être associés à des caractéristiques génériques.

Le fait que l'approche GOSPEL s'appuie sur la réutilisation des agents à partir de Self-StarMAS [110] présente l'avantage que les agents en question soient auto-adaptatifs. Rappelons-le, ces derniers ont été développés de façon spécifique pour des dispositifs ou appareils légers tels que les capteurs de mouvements. Par conséquent, les agents présentent l'avantage d'adapter leurs comportements en fonction des appareils connectés au niveau desquels ils sont intégrés. De ce fait, leurs actions restent relatives à leurs buts, tel que les agents intégrés au niveau des appareils portables, ou encore les agents réactifs qui baseront leur comportement sur les règles fixées tout comme les agents intégrés dans les capteurs. En dépit de ces avantages, l'inconvénient principal de l'approche reste sa spécificité au domaine de l'internet d'objets (ioT).

Un autre avantage est celui d'avoir construit des modèles de variabilité tenant compte de plusieurs variations possibles telles que le support de protocoles de transport hétérogènes (Wifi, Bluetooth et ZigBee). Cependant, ce modèle dont un extrait est représenté dans la figure 3.6, n'est pas forcément réutilisable ni applicable à d'autres domaines d'application car ces derniers restent spécifiques au domaine considéré.

3.4.2.2 La composition de GAIA et FAST

1. **Description**

   Dehlinger et al. [25] reprennent un sous-ensemble d'étapes de la méthodologie multi-agents GAIA, en la reliant à des besoins d'ingénierie établis au préalable. Cette approche permet à GAIA de supporter la capture et la modélisation de l'ingénierie des besoins

premiers. Ceci est réalisé en composant une portion de la méthodologie multi-agents GAIA à une portion issue de la méthodologie de ligne de produits logiciels nommée FAST (Family-oriented Abstraction Specification and Translation). De façon plus précise, les portions de la méthodologie FAST sont intégrées afin d'être appliquées pour l'analyse et la conception des Rôles d'agents. Une analyse préliminaire des similitudes et de la variabilité CVA (Commonality and Variability Analysis) [118] est réalisée.

Cette combinaison est représentée dans la figure 3.7 où nous retrouvons bien les étapes de la méthologie multi-agents Gaia [67] qui ont été combinées au sous processus CVA de la méthodologie FAST comme expliqué plus haut ; ce qui a permis d'obtenir sur la base de la documentation des besoins (requierements), des schémas de points de variations des rôles d'agents (analyse et conception).

2. **Avantages et inconvénients**

Dans un de leurs travaux, Dehlinger et al. [103] proposent un template extensible de spécification des besoins pour les systèmes distribués avec support de réutilisation sur des besoins dérivés à partir de systèmes multi-agents. Leur approche ne reprend que quelques portions de la méthodologie GAIA. Cependant, les éléments qu'ils proposent de réutiliser représentent les configurations possibles d'un agent qu'ils capturent à travers l'évolution du système. Il s'agit de ces mêmes configurations ou points de variations qu'ils introduisent au sein des spécifications de besoins en utilisant des templates de spécification de besoins.

Leur approche présente une certaine flexibilité malgré que l'adoption de CVA ne soit pas la plus utilisée ni la plus adaptée des techniques de modélisation des besoins. La flexibilité réside dans la possibilité de remplacer cette activité par des approches orientées caractéristiques (features) ou orientées but (goal).

Les points de variations qu'ils capturent sont en relation directe avec les différences pouvant exister au sein d'un même rôle du système. Bien que le rôle pouvant présenter des points de variation au niveau des protocoles, des activités, des permissions et des responsabilités ; cette variabilité reste spécifique au domaine d'application. Par contre la détection de points de variations indépendante du domaine d'application présente des aspects intéressants réutilisables comme par exemple, définir des degrés différents de variation d'intelligence d'un rôle. Tel qu'une configuration d'un rôle serait de niveau d'intelligence I4 pour l'exécution/réception de commande ; tandis qu'une intelligence de niveau I3 représenterait un niveau d'intelligence I4 qui serait en plus capable de réaliser une planification locale. Ces points de variations permettant de configurer des rôles de niveaux d'intelligence variable représentent un aspect très important et très pertinent qui pourrait apparaître au niveau du modèle de similitudes et de variabilité. Leur exploitation des points de variations ne se fait qu'une fois que l'agent exécute un rôle donné.

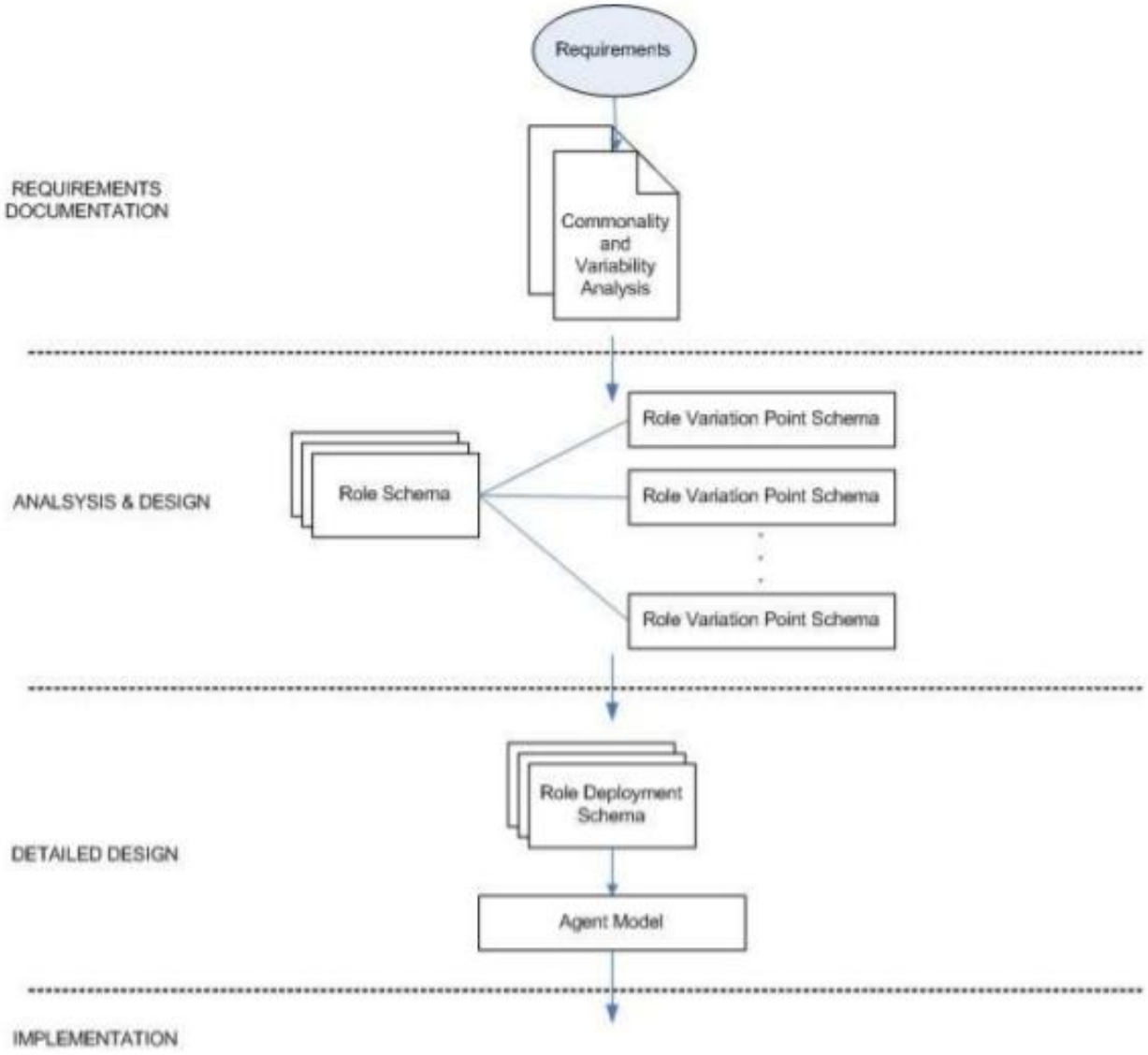


FIGURE 3.7 – Approche MAS-PL combinant Gaia et Fast [25]

Nous pensons qu'au lieu de représenter ces points de variations uniquement au niveau de l'exécution d'un agent (runtime), il serait intéressant d'introduire les caractéristiques (features) relatives à toutes les configurations possibles au sein du modèle. Par exemple ces caractéristiques pourraient tout aussi bien inclure : "Receive/execute commande", "Local-Planning", "Interaction", "Cluster-knowledge" ("full" ou "partial") entant que caractéristiques variables au niveau conceptuel. Tel que chaque configuration donnerait des rôles de niveaux d'intelligence différents. Ces points de variations permettraient de modéliser les systèmes multi-agents entant que ligne de produits. Les auteurs parlent de

configuration d'un agent relativement à ses rôles, et ses points de variation. Pourtant, ils ne représentent pas la base sur laquelle cette sélection de caractéristiques (configuration) sera réalisée ; alors que ces points de variations ne figurent pas au sein du modèle, et considèrent que ces points de variations seront particuliers à chaque application. Toutefois, ceci ne s'applique pas d'un autre niveau d'abstraction. Certes, ils précisent que d'avantages de caractéristiques telles que : "passive" ou "active" pourraient concerner seulement des applications spécifiques. Cependant, ceci n'empêcherait pas de les intégrer de façon variable au niveau du modèle qui serait réutilisable indépendamment d'une application spécifique. Les éléments qu'ils considèrent réutilisables représentent les configurations d'agents pouvant être réutilisés lors du développement initial du système, et ses mises à jours.

Quant à l'approche relative aux template extensible de spécification des besoins [103], elle permet de fournir une technique d'analyse de sûreté, du moment qu'elle est sensible aux configurations dynamiques au sein des composants du système qui représentent les agents et qu'elle permet de supporter une réutilisation de façon sûre. Cependant, les aspects réutilisables dépendent de l'application en question, car l'évolution du système ne passe pas forcément par les mêmes ajouts lorsque nous considérons d'autres domaines d'applications, et d'autres configurations possibles.

3.4.3 Approches construites de zéro (adhoc)

Cette dernière catégorie est celle des approches construites de zéro, et ce afin de répondre à des besoins spécifiques de la ligne de produits multi-agents ; ou encore à des problèmes spécifiques à certains domaines d'application. Bien qu'elles soient construites de zéro, nous allons voir qu'elles peuvent néanmoins s'inspirer d'approches déjà existantes.

3.4.3.1 L'approche MAS-PL de construction du coeur de l'architecture

1. **Description**

   Peña et al. [104] ont proposé une approche construite de zéro dans le but est de construire le corps de l'architecture d'une ligne de produits multi-agents par le biais de composition de modèles de rôles d'agents. Cette approche n'est cependant pas axée sur l'association existante entre cette composition de modèle réalisée lors de l'analyse du domaine d'application et les artefacts d'implémentation correspondants.

   Pour la spécification des similitudes et de la variabilité, elle est réalisée par le modèle de caractéristiques FM. Cependant, le reste des étapes de leur approche s'inspire de la méthodologie MaCMAS en utilisant un autre type de diagramme, celui-ci étant un diagramme de traçabilité ; qui est un diagramme hiérarchique assez proche du modèle

de caractéristiques FM. Ce dernier produit un ensemble de modèles de rôles relatif à chaque but (Goal) du système. Les modèles d'acquittance qu'ils produisent sont utilisés entant que support de construction de la structure de l'organisation du système. Ce support regroupe ainsi les caractéristiques communes entre ces modèles une fois la ligne multi-agents conçue. Il s'agit de ces mêmes caractéristiques communes qui représentent le corps de l'architecture réutilisable. Les étapes extraites de MaCMAS qui ont été certes reprises mais également réadaptées en incluant les différentes vues de l'organisation du système multi-agents : (i) Vue statique : représente principalement les aspects statiques relatifs à l'organisation du système en utilisant par exemple des modèles de Rôles, afin de montrer comment chaque Goal du système est matérialisé (ii) Vue dynamique : représente les aspects relatifs au comportement de l'organisation du système, en représentant par exemple le plan relatif à chaque rôle du système (iii) Vue de traçabilité : pour représenter la façon avec laquelle les modèles de niveau d'abstractions différents sont reliés, mais se focalisent souvent sur les relations entre les interactions car elles représentent le cœur du modèle.

2. **Avantages et inconvénients**

   La première approche présentée ci-dessus a comme avantage de représenter une approche de départ d'une ligne de produits multi-agents ; dès lors qu'elle se focalise principalement sur le processus à adopter pour capturer le corps de l'architecture d'un système multi-agents qui servira de base ou de référentiel par la suite lors de l'évolution du système à travers les différentes variantes de la ligne. Cependant, elle ne donne aucune indication quant à la façon de greffer de nouvelles caractéristiques au niveau de l'architecture de base. De plus, aucune directive couvrant l'ingénierie d'application n'est donnée, que cela concerne l'implémentation ou encore les configurations possibles et la dérivation de code.

   Pour finir, le modèle organisationnel exploité au sein de l'approche est certes riche vu qu'il aborde l'organisation sous différentes vues. Par contre, qu'en est-il de l'exploitation des autres modèles d'organisations ? En effet, il se peut que le besoin d'un nouveau produit multi-agents soit accompagné de changements conceptuels d'un niveau organisationnel, ou encore de l'introduction de nouveaux modèles non pris en charge par l'approche, tel que le support d'avantages de topologies

3.4.3.2 L'approche SelfStarMAS

1. **Description**

   L'approche SelfStarMAS construite de zéro et qui a été proposée par Ayala. et al. [119] est une approche de développement de lignes de produits multi-agents pour l'internet d'objets ou ioT(internet Of Things). Les auteurs ont mis en place cette approche en pro-

posant un processus de développement d'agents auto-adaptatifs pour le domaine de IoT. Ils n'utilisent pas de modèles de caractéristiques au niveau de l'ingénierie du domaine pour la spécification des similitudes et de la variabilité, mais ont opté plutôt pour le CVL (Common Variability Language) [44]. Leur approche SelfStarMAS fournit un ensemble d'agents réutilisables, qui adapte et étend les agents FIPA-compiliant [1] ; et ce afin qu'ils soient réutilisables dans le domaine de l'ioT en question. Ces agents réutilisables développés sont représentés sous différentes versions tout en pouvant être intégrés au niveau de dispositifs Android ou autre tels que Libelium waspmotes. Une caractéristique principale de ces agents est qu'ils sont adaptés aux ressources des appareils ou dispositifs où les agents seront intégrés.

2. **Avantages et inconvénients**

   Le premier avantage de l'approche est qu'elle permet de couvrir l'ensemble des phases de développement d'une ligne de produits multi-agents.

   Cependant l'inconvénient majeur est que la plateforme SOL multi-agents [111] au niveau de laquelle les agents sont déployés a été conçue pour le support de d'agents spécifiques au domaine d'application uniquement.

   Cette approche a été étendue par une approche de Lignes de produits dynamiques qui présente l'avantage d'adaptation du comportement d'agents lors de l'exécution [110]. Cette adaptation présente l'avantage de gérer la variabilité à un niveau de l'éxécution (runtime). Cependant son inconvénient majeur est qu'elle reste également spécifique au domaine d'application et rejoint donc le même inconvénient qui se pose dans la première version de l'approche. De plus, bien que cette approche couvre l'ensemble des deux phases d'ingénierie, il y a une focalisation sur l'ingénierie d'application aux dépens de l'ingénierie du domaine au niveau de laquelle l'apport principal est de proposer une sorte d'ontologie relativement à la variabilité des concepts IoT.

   Une autre approche MAS-PL a été proposée pour les systèmes ambiants intelligents et plus précisément les systèmes AAL(Ambient Assisted Living) [120]. L'approche en question réutilise le même processus que celui proposé au préalable [110]. En effet, leur proposition permet de gérer la variabilité des agents ainsi que les dépendances logicielles et matérielles issues du domaine concerné. De plus, cette proposition permet d'intégrer différents degrés d'auto-adaptation d'agents en fonction des besoins des systèmes AAL ainsi que les dispositifs physiques où l'agent sera intégré. L'approche reprend le processus général du développement de lignes de produits logiciels en y apportant des modifications au niveau de l'ingénierie d'application et ce en précédant la génération de variante par un processus supplémentaire, ce dernier nommé "Weaving process" permet de tisser le

1. `http://www.fipa.org`

modèle de buts (goals) des agents avec l'architecture générée dans la phase d'ingénierie d'application.

Concernant l'extension qui a été faite par la suite [110], nous pouvons citer comme avantage principal l'obtention d'une consistance architecturale pour les variantes. Mais, le même problème de sa spécificité se pose. De plus, les variantes obtenues en sortie à la fin de la phase d'ingénierie d'application ne peuvent être directement déployées vu qu'elles ne représentent qu'une des variante architecturales possibles.

3.5 Outils de dérivation de lignes de produits multi-agents

Certains travaux sur les MAS-PLs proposent des outils de dérivation de lignes de produits multi-agents. C'est dire que leurs contributions se situent au niveau de la phase d'ingénierie d'application (voir GenArch dans la figure 3.3). Certains outils sont développés de zéro dans le but d'être spécifiquement dédiés à la dérivation des systèmes multi-agents. D'autres proposent des extensions d'outils. Etant donné que ces travaux n'entrent pas dans le cadre de notre travail de recherche, nous les décrivons brièvement dans ce qui suit :

3.5.1 Outils de dérivation Adhoc

Cirilo et al. [121] proposent l'outil GenArch. Bien que leur travail ne couvre pas l'ingénierie du domaine, les modèles de dérivation qu'ils proposent offrent des avantages à l'ingénieur du domaine tel que la possibilité d'utiliser des features modèles en utilisant des approches annotatives lors de l'implémentation du domaine pour y faire référence. En effet, GenArch fournit des annotations JAVA qui peuvent être utilisées au niveau de la variation LdP logiciels ainsi que des points de variation pour permettre une construction automatique des modèles (FM, modèle d'implémentation et modèle de configuration). Le but principal de ces annotations étant de permettre un mapping entre une feature au niveau du modèle et une feature au niveau des assets de code JAVA pour caractériser la feature en utilisant par exemple l'annotation @Feature. Un autre exemple est de caractériser un point de variation par l'annotation @Variability. Bien que les approches annotatrices permettent d'effectuer un lien direct entre l'ingénierie du domaine et de l'application sans que cette approche ne couvre l'ingénierie du domaine, ces annotations restent pauvres et ne permettent pas de faire une distinction entre les différents niveaux de granularité des caractéristiques et ce en considérant les caractéristiques à granularité fine et grossière comme étant au même niveau sans aucune distinction lors de la dérivation.

3.5.2 Extensions d'outils de dérivation

Les approches MAS-PL qui étendent les outils de dérivation tentent principalement de définir un processus léger durant l'ingénierie d'application. Dans ce contexte, Nunes et al. [122] proposent un outil de dérivation de systèmes multi-agents étendant l'outil GenArch [121]. La particularité de leur approche est qu'ils proposent des modèles multi-niveaux afin de supporter la spécification des connaissances de configuration, ainsi que la dérivation automatique de produit MAS-PL. Un autre avantage de leur processus d'automatisation de la dérivation de produits multi-agents est qu'ils exploitent et combinent les avantages des mécanismes d'approches orientées code ainsi que les mécanismes concernant les connaissances des configurations spécifiques au domaine. L'un des inconvénients de cette approche est que l'outil qu'ils proposent ne génère que des agents ayant une architecture BDI et s'exécutant sur Jadex [123].

3.6 Comparaison des approches MAS-PL

Dans cette section, nous proposons un ensemble de critères de comparaison des approches MAS-PL présentées et analysées dans les sections précédentes. Nous expliquons nos choix quant aux critères de comparaison, et donnons un résumé au niveau d'un tableau comparatif. Pour finir, nous discutons de l'ensemble des points abordés en présentant de façon synthétique les principaux avantages et inconvénients de ces approches.

3.6.1 Choix des critères de comparaison

Les critères de comparaison des approhes MAS-PL ont été choisis de façon à mettre en avant les aspects que nous estimons nécessaires de retenir, et que nous avons jugés comme étant importants suite à notre analyse.

Chaque point est pertinent car il permet de répondre à des questions primordiales afin de délimiter les principaux axes autour desquels devrait s'appuyer une approche MAS-PL ; et qui sont les suivants :

— Quelle méthodologie multi-agents choisir ?
  Le premier critère choisi est celui de la méthodologie multi-agents qui a été étendue ou encore réutilisée en partie afin de construire l'approche de lignes de produits multi-agents. Le choix de la méthodologie permet entre autres de délimiter les concepts multi-agents couverts par l'approche, mais aussi de voir comment celle-ci a été étendue dans certains cas.

— Quelles sont les phases couvertes par l'approche ?
  Le second critère choisi est celui de la ou des phases couvertes par l'approche. En effet, pour comprendre une approche MAS-PL, il est important de comprendre où se situe

la contribution mais aussi quelles phases sont couvertes par l'approche. Comme nous l'avons vu certaines approches telles que Gospel [24], ou encore PASSI-PL [93] couvrent l'ensemble des phases d'ingénierie du domaine et d'application. D'autres approches telles que Gaia-PL [101], interviennent au niveau de l'ingénierie du domaine uniquement. Et enfin, nous retrouvons des outils de dérivation de produits, et donc qui interviennent pendant la phase d'ingénierie d'application comme l'outil GenArch [109].

— Quels concepts et quel type de granularité choisir pour les caractéristiques ?
Le troisième critère est celui qui rejoint en partie le premier critère mentionné ci-dessus. Il est en relation avec un qualificatif important des concepts qui sont couverts par l'approche, et qui est lié directement à la granularité des caractéristiques. Comme nous l'avons expliqué dans ce chapitre, la modularité d'une approche est un point important. Celle-ci va être déterminée en partie grâce à la granularité considérée au sein des caractéristiques faisant partie du modèle.

Extension				***Phase d'ingénierie***		**Modularité**		**Approches et outils SPL**				
Approches MAS-PLs	PASSI	GAIA	MaCMAS	Ingénierie du domaine	Ingénierie d'application	Granularité fine	Granularité grossière	Feature Model	CVL	FAST	PLUS	SPL Dynamique
[23]	-	+	-	+	-	-	+	+	-	+	-	-
[105]	-	-	+	+	-	-	+	+	-	+	-	-
[101]	-	+	-	+	-	-	+	+	-	+	-	-
[93]	+	-	-	+	-	+	+	+	-	-	+	-
[22]	+	-	-	+	-	-	+	+	-	-	-	-
[124]	-	-	-	+	-	-	-	+	-	-	+	-
[108] [125]	+	-	+	+	-	+	+	+	-	-	-	-
[126]	-	-	+	+	-	-	+	+	-	-	-	-
[25]	-	+	-	+	-	-	+	+	-	+	-	-
[104] [23]	-	-	+	+	-	-	+	+	-	-	-	-
[110]	-	-	-	+	+	-	+	-	-	-	-	+
[119]	-	-	-	+	+	-	+	-	+	-	-	-
[120]	-	-	-	+	+	-	+	-	+	-	-	-
[24]	-	-	-	+	+	-	+	-	+	-	-	-
[122]	-	-	-	-	+	+	-	-	-	-	-	-

TABLE 3.2 – Synthèse sur les approches MASPLs

Les caractéristiques à granularité fine peuvent donner plus de détails ; et devenir ainsi des caractéristiques atomiques pouvant être décomposées. En d'autres termes, plus la granularité est fine lors de la modularisation, plus les caractéristiques seront spécifiques et par conséquent la probabilité de les réutiliser en caractéristiques alternatives et va-

riables augmente. En revanche, les modularisations à granularité plus épaisse minimise la réutilisation des caractéristiques.

Les caractéristiques à granularité grossière, peuvent être implémentées et encapsulées au sein d'une unité spécifique telle qu'une classe ou un agent. Nous retenons donc ce critère comme important lors du choix des caractéristiques à considérer au niveau d'un modèle pour la ligne de produits multi-agents.

— Quels modèles choisir pour la spécification des similitudes et de la variabilité de la ligne de produits multi-agents ?

Les derniers critères sélectionnés sont en association directe avec les approches provenant de l'ingénierie de lignes de produits logiciels. Ils concernent les choix d'approches de modélisation telles que les modèles de caractéristiques FM (Feature Model),ou encore CVL (Commonality and Variability Language), ainsi que d'autres approches utilisées ou avec lesquelles les approches multi-agents ont été combinées telles que l'approche FAST associée au sous processus CVA (Commonality and Variability Analysis).

Ces éléments regroupent les points essentiels à retenir de chacune des approches MAS-PL présentées ; mais aussi les choix possibles et effectués afin de permettre le développement d'une ligne de produits multi-agents ; et pour lesquels les choix des approches MAS-PL ont été analysés.

3.6.2 Analyse et discussion des avantages et inconvénients

Sur la base des critères choisis, le tableau 3.2 présente un récapitulatif comparatif des approches MAS-PLs que nous avons analysées, et pour lesquelles nous avons discuté avec détail les avantages ainsi que les inconvénients dans les sections précédentes.

De ce tableau comparatif, il ressort que la majorité des approches MAS-PL utilisent le modèle de caractéristiques FM (feature model) pour la spécification de la variabilité et des similitudes. Peu d'approches utilisent d'autres types de modélisation tel que CVL (Commonality and Variability Language).

Quelques approches seulement s'inscrivent dans le contexte du SME, et ont repris FAST ou encore PLUS. La majorité des approches MAS-PL se situe dans la catégorie d'extensions d'approches.

Nous remarquons également que les seules méthodologies issues de l'ingénierie des SMA, et qui sont reliées à une approche MAS-PL sont respectivement PASSI, GAIA et MaCMAS. Par conséquent, l'utilisation des approches MAS-PL est restreinte à l'utilisation uniquement des concepts couverts par ces méthodes.

Pour finir, nous remarquons que la majorité des approches utilisent une granularité grossière des caractéristiques, et que seules deux approches utilisent les deux types de granularité. Ce

qui induit comme expliqué au préalable, à une mauvaise modularité, et une faible probabilité de réutilisation des caractéristiques. Nous pensons, que ceci réduit entre autre l'impact de la réutilisation associée directement aux concepts. En d'autres termes, le fait d'inclure d'avantages de choix possibles des concepts avec une granularité plus fine pour les caractéristiques, permettrait d'introduire un niveau de détail des caractéristiques, qui se répercuterait au niveau de l'implémentation, tout en permettant d'augmenter la probabilité d'une éventuelle réutilisation.

D'une façon générale, toutes ces approches présentent les avantages suivants :

1. Les extensions des méthodologies multi-agents ont rendu possible le développement d'applications multi-agents similaires, et ont ainsi amélioré la réutilisation au sein de familles de SMA. Gaia-PL [101] a permis par exemple la réutilisation des spécifications des besoins. Un autre exemple est celui de PASSI-PL [93], qui a permis la spécification des similitudes et de la variabilité entre les produits multi-agents par le biais d'introduction de nouveaux stéréotypes au niveau des modèles UML.
2. Couverture du cycle complet de développement de la ligne de produits multi-agents en couvrant les deux phases d'ingénierie du domaine et d'application.
3. L'amélioration du processus d'ingénierie des lignes de produits multi-agents grâce à la prise en compte des contraintes liées au domaine d'application. Par exemple, SelfStar MAS [119] a intégré au sein du modèle spécifique à l'ioT (internet of things) la contrainte relative aux architectures d'agents qui peuvent être supportées (agents réactifs), dans le cas de sélection des caractéristiques de capteurs ioT. Sans oublier que l'introduction des approches FAST, ou encore PLUS par exemple a permis de construire des compositions d'approches plus complètes.
4. Pour finir, certaines approches telles que Gospel [24] offrent une flexibilité pour la modélisation des similitudes et de la variabilité des produits de la ligne de produits multi-agents, et qui peut donc se faire par le biais de modèles FM, ou encore de modèles CVL.

Bien que les avantages des approches MAS-PL existantes soient multiples, nous avons relevé les inconvénients suivants :

1. Aucune des approches ne propose un point de départ pour la construction du modèle de la ligne de produits multi-agents. Par conséquent, le modèle de similitudes et de variabilité est spécifique au domaine d'application, et doit être construit de zéro. En effet, comme présenté tout au long de ce chapitre, les modèles construits par les approches ne sont pas forcément applicables à d'autres domaines d'application. Par exemple, ces derniers ne permettent pas d'utiliser des protocoles d'interaction connus et très répandus dans le domaine comme le protocole du Contract net.
2. Seuls les concepts supportés par la méthodologie issue de l'ingénierie des SMA ; et donc concernée par l'approche MAS-PL sont supportés. Par conséquent, si un nouveau produit

à inclure nécessite des concepts issus d'autres méta-modèles, ces derniers ne seront pas supportés par l'approche en question. Comme nous avons pu le voir, parmi toutes les approches MAS-PLS, les seules méthodologies qui ont été étendues sont : PASSI [76,113] , GAIA [67,114,115], et MaCMAS [116].

3. Aucun point de départ n'est donné pour l'implémentation de la ligne de produits multi-agents. Celle-ci est donc développée de zéro, et demande donc plus d'effort de la part du développeur. Comme nous avons pu le voir par exemple, l'approche PASSI-PL [93] permet de réutiliser le code source de façon spécifique au domaine d'application, mais celle-ci est réalisée de zéro.
4. La traçabilité de la variabilité par les annotations peut devenir difficile lorsqu'il s'agit d'un grand nombre de modèles utilisés.

3.7 Conclusion

Dans ce chapitre, nous avons présenté les travaux de recherche relatifs aux approches de lignes de produits multi-agents MAS-PL. Nous avons mis en avant leurs principales caractéristiques en comparaison avec des pratiques de réutilisation classiques. Nous avons également retracé leur évolution au fil du temps. Suite à cela, nous avons présenté une classification des approches existantes incluant les extensions, les compositions ainsi que les approches construites de zéro. Nous avons entre autres présenté les avantages et inconvénients de ces approches. Pour finir, nous avons comparé les approches analysées sur la base de critères jugés importants. De notre analyse, il ressort que ces approches ont permis d'améliorer la réutilisation au sein de familles de SMA ; comme le fait de proposer des approches qui facilitent la conception et le développement des lignes de produits multi-agents (PASSI-PL [93], GAIA-PL [101], GOSPEL [24] etc.). De plus, d'autres approches ont facilité la réutilisation, en offrant la possibilité de construire le coeur de l'architecture d'une ligne de produits multi-agents.

Cependant, le principal inconvénient qui attire notre attention est que les modèles de caractéristiques et artefacts d'implémentation ne sont pas réutilisables car ils sont spécifiques du domaine d'application.

Par conséquent, l'écart reste considérable entre les changements à apporter au niveau de la ligne de produits multi-agents et de leur implémentation.

De plus, la variabilité analysée peut parfois se faire après la modélisation du système multi-agents [104], ce qui mène à un fort couplage entre les caractéristiques communes et variables suite à une modularisation qui peut s'avérer inappropriée.

Pour pallier aux problèmes cités, il faudrait mettre en place une approche MAS-PL capable d'une part de réutiliser le modèles de similitudes et de variabilité indépendamment du domaine

d'application ; et d'autres part de réutiliser des implémentations de façon indépendante du domaine d'application. C'est ce que nous proposons dans le prochain chapitre, qui présente notre approche MAS-PL, et qui tente de résoudre le problème de développement d'applications multi-agents similiaires tout en repoussant les limites des approches existantes.

Quatrième partie

Contributions

Chapitre 4

Vers une approche d'ingénierie MAS-PL incrémentale indépendante du domaine d'application

4.1 Introduction

A ce jour bien qu'il existe plusieurs approches de lignes de produits multi-agents MAS-PL (Multi-agent System Product Line) ayant facilité le développement des familles de SMA (applications multi-agents similaires) ; aucune d'entre elles ne propose des points de départ pour les modèles de caractéristiques ; ou encore pour l'implémentation de la ligne de produits multi-agents.

En effet, bien que les approches présentées dans le chapitre précédent permettent une réutilisation des modèles et de leurs implémentations, ces derniers doivent être réalisés de zéro car ils sont spécifiques au domaine d'application ; et par conséquent ils ne sont pas forcément réutilisables indépendamment du domaine d'application.

Nous avons donc pensé, à proposer une approche MAS-PL capable d'offrir deux points de départ : un premier pour la construction du modèle de caractéristiques ; et un second pour les implémentations associées. L'objectif de cette démarche, est que le modèle de caractéristiques soit applicable à un plus grand nombre d'applications multi-agents possible.

Pour réaliser cela, notre approche s'articule autour d'un modèle de caractéristiques constitué de deux couches (niveaux), représentant une ligne de produits multiple (voir chapitre 1).

Le premier niveau représente les similitudes, et la variabilité des concepts issus de notre analyse du domaine des SMA (FM générique), qui sera présenté en détail dans le prochain chapitre. Le second niveau, représente les similitudes et la variabilité issues de l'analyse d'un domaine d'application spécifique (FM spécifique). Il en est de même pour les implémentions

associées à chacun des deux niveaux du modèle de caractéristiques. Notre approche propose un premier niveau d'implémentation lié aux artefacts réutilisables du domaine des SMA ; et un second niveau relatif au domaine d'application, qui est développé en réutilisant les artefacts du premier niveau.

Dans ce chapitre nous présentons notre approche MAS-PL [127, 128], par l'esquisse d'une vue d'ensemble, l'évocation de ses principes et fondements. Nous allons ensuite la situer parmi les approches MAS-PL existantes, que nous avons présentées et analysées dans le chapitre de de l'état de l'art.

Nous pourrons alors passer à une description plus détaillée de notre approche, en donnant l'ensemble des choix que nous avions à faire ; tout en justifiant ces derniers. Nous présenterons de façon détaillée chacune des phases d'ingénierie du domaine et d'application ainsi que les activités qui constituent notre approche. Chaque activité sera illustrée par le biais d'exemples.

4.2 Vue d'ensemble de l'approche

Afin de résoudre le problème de développement d'applications multi-agents similaires, et de pallier aux problèmes soulevés au niveau des approches MAS-PL existantes, nous proposons l'approche MAS-PL dont la vue globale est représentée dans la figure 4.1.

4.2.1 Principes et fondements

Notre approche se base sur les principes et fondements suivants :

1. **Adaptation du processus général des lignes de produits logiciels :**
   Comme présenté au niveau de la figure 4.1, notre approche s'appuie sur le processus général du développement de lignes de produits logiciels [3], et couvre donc les deux phases *d'ingénierie du domaine* et *d'ingénierie d'application* (voir chapitre 1).
2. **Découpage des activités d'analyses (resp. d'implémentation) du domaine :**
   Bien que notre approche reprend intégralement le processus référentiel de l'ingénierie de lignes de produits logiciels, l'originalité de notre proposition est qu'elle permet une réutilisation indépendante du domaine d'application. C'est pourquoi nous avons procédé à un découpage des activités d'analyse et d'implémentation, de façon à avoir d'une part une analyse (resp. implémentation) indépendante du domaine d'application, et donc générique, et d'autres part une analyse (resp. implémentation) spécifique au domaine d'application.

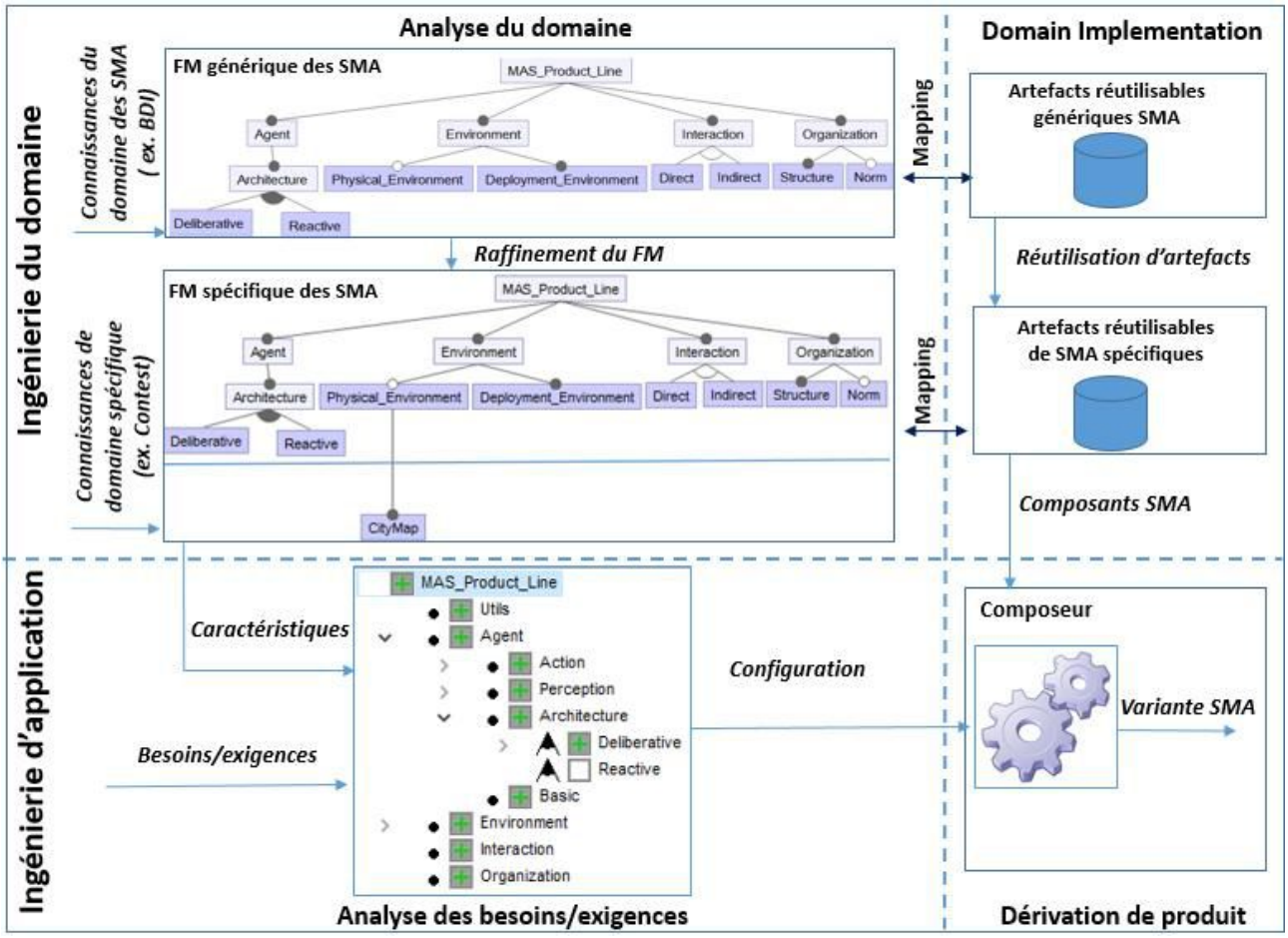


FIGURE 4.1 – Vue d'ensemble de l'approche MAS-PL

3. **Deux niveaux pour le modèle de caractéristiques FM et les artefacts :**
   La distinction d'activités d'analyse (resp. d'implémentation) du domaine a été réalisée dans le but d'obtenir des modèles de caractéristiques FM (resp. artefacts) à deux niveaux. Le premier niveau du FM (resp. artefacts) est en relation avec des similitudes et variabilité relatives aux concepts issus des connaissances du domaine des SMA (méta-modèles, ontologies, etc.). Le second niveau, est issu de l'analyse des connaissances du domaine d'application, et est obtenu par raffinement du modèle indépendant du domaine d'application issu de la première couche. Un produit multi-agents final, sera donc obtenu à partir de caractéristiques (resp. artefacts) en provenance des deux niveaux.
4. **Approche VOYELLES pour le premier niveau du FM :**
   Afin de permettre une réutilisation des caractéristiques (resp. artefacts) issues du domaine des SMA, nous nous sommes appuyés sur l'approche VOYELLES (A.I.E.O) [28] (voir chapitre 2) au premier niveau du modèle FM (niveau générique) ; et avons procédé

à un développement incrémental par raffinement du modèle.

L'approche VOYELLES nous a permis d'organiser notre modèle de caractéristiques des SMA. De plus, l'analyse des similitudes et de la variabilité de chacune des dimensions de l'approche (A : agent, E : environnement, I : interaction, et O : Organisation), nous a permis de proposer une extension de l'approche VOYELLES en approche MAS-PL (voir la figure 4.2).

4.2.2 Description générale des activités de l'approche

Notre approche MAS-PL est constituée de l'ensemble des activités suivantes :

1. ***Analyse du domaine des SMA (niveau générique)***

   Cette première activité d'analyse du domaine *générique*, concerne l'analyse des connaissances du domaine des SMA. Ce domaine fait référence à l'ensemble des connaissances en termes de concepts impliqués dans les méthodes, les modèles, et ontologies.

   Les concepts sont extraits, puis leurs sémantiques est comparée, afin qu'il n'y est pas de redondance au sein du modèle à construire. Par exemple, les concepts *AIP* (issu du méta-modèle de la méthodologie PASSi), et *Protocol* (issu du méta-modèle de la méthodologie GAIA) ont la même sémantique. Ils seront donc représentés par une même caractéristique.

   Mise à part leurs sémantiques, les concepts doivent être reliés à la caractéristique du *nœud père*. Les caractéristiques relatives à ces concepts, seront donc considérées comme des *nœuds fils*. Par exemple, le concept de rôle représenté au sein de notre modèle par la caractéristique *Role*, est un *nœud fils* de la caractéristique représentée par le *nœud père Organisation*.

   De plus, les concepts sont également analysés en termes de points communs (similitudes) et variables (différences). Par exemple, le concept*Role* est un concept commun entre les méta-modèles issus de notre analyse. Par conséquent, la caractéristique *Role* (nœud fils), doit être liée à la caractéristique *Organisation* (nœud père) par la relation de type *obligatoire (mandatory)* suivante :

   $mandatory(Organisation, Role) \equiv Role \leftrightarrow Organisation$.

   Cette activité produit en sortie un *Modèle de caractéristiques génériques des SMA / Generic MAS Feature Model (FM)* (voir la figure 4.3).

   Ce modèle de caractéristiques, peut être raffiné et étendu, en élargissant la délimitation du domaine analysé. Il est donc possible d'inclure d'autres concepts issus d'analyse de modèles non inclus dans notre présente étude.

2. ***Implémentation d'artefacts indépendants du domaine d'application (niveau générique)***

   La deuxième activité est celle de l'implémentation d'artefacts multi-agents réutilisables, qui sont indépendants du domaine d'application. Ces artefacts sont directement associés aux caractéristiques concrètes du modèle FM construit durant la première activité.

   Ces implémentations, sont donc constituées d'artefacts : d'agents, d'environnement, d'interaction et d'organisation. C'est pourquoi, tout comme les caractéristiques du même niveau, elles sont réutilisables indépendamment du domaine d'application.

   Afin de les implémenter, nous avons recherché entre autres quels étaient les artefacts réutilisables capables d'implémenter les fonctionnalités dites *génériques*.

   Par exemple, nous avons recherché une implémentation réutilisable pour les concepts de *perception*, et d'*action* de l'agent, qui proviennent du premier niveau du modèle de caractéristique (FM générique). C'est ainsi que nous avons réutilisé APLTK (Agent Programming Toolkit) [129]. Ces implémentations réutilisables sont directement associées (mapping) aux caractéristiques *perception* et *Agent_ action* de notre modèle (voir la figure 4.3).

3. ***Analyse du domaine d'application (niveau spécifique)***

   Cette troisième activité est celle de l'analyse du domaine *Spécifique*. Cette analyse, permet d'identifier les points communs et variables entre les membres d'une famille d'applications multi-agents spécifiques à un domaine d'application. Nous illustrons cette activité par le biais de trois cas d'études : le Multi-Agent Contest [1], la vente de livres (book trading issue de JADE) [2] ; et la gestion d'un emploi du temps, que nous présentons avec détail dans le chapitre 6.

   Cette activité produit en sortie un *Modèle de caractéristiques spécifiques/ Specific FM*. Contrairement aux approches MAS-PL existantes, la notre propose de construire ce modèle non pas de zéro, mais en réutilisant le modèle de caractéristiques génériques issu de l'activité précédente. Les caractéristiques de ce niveau (FM spécifique) peuvent représenter des nœuds fils des caractéristiques du premier niveau (FM générique), lesquelles représentent des nœuds pères. Par exemple, la caractéristique relative au rôle spécifique *Buyer* du domaine domaine du multi-agent contest, est représentée par un nœud fils de la caractéristique *Role* qui joue le rôle d'un nœud père dans ce cas (voir la figure 4.15).

1. https ://multiagentcontest.org/
2. https ://jade.tilab.com/

4. ***Implémentation d'artefacts dépendants du domaine d'application (niveau spécifique)***

   La dernière activité de la phase d'ingénierie du domaine, est celle du développement de *SMA spécifique/ Specific MAS*. Cette activité, permet de produire des artefacts réutilisables d'un domaine d'application spécifique. Ces artefacts sont associées directement aux caractéristiques qu'ils implémentent. Ces dernières, proviennent du second niveau de notre modèle (FM spécifique), issu de l'activité précédente.

   Dans les approches MAS-PL existantes, ces artefacts réutilisables sont implémentés à partir de zéro. A l'inverse, notre approche propose que l'implémentation *spécifique* au domaine d'application, puisse être réalisée en réutilisant les implémentations dites *génériques*, que nous avons regroupées pendant l'activité précédente.

5. ***Analyse des besoins et dérivation du produit multi-agent***

   Les deux dernières activités de notre approche, sont celles de l'ingénierie d'application. Ces activités concernent d'une part l'analyse des besoins du système multi-agents, et d'autres part sa dérivation.

   Cette phase produit en sortie une variante ou version possible d'un SMA, à partir de l'ensemble des produits multi-agents supportés par le modèle.

   Après l'analyse des besoins d'utilisateurs, une configuration correspondante est réalisée en sélectionnant les caractéristiques souhaitées au niveau de la variante multi-agents. Cette sélection de caractéristiques se fait à partir du modèle de caractéristiques *spécifiques*.

   Sur la base de cette configuration, le produit est obtenu par dérivation automatique.

   Notre approche, ne présente aucune originalité à ce niveau, car nous avons réutilisé un générateur de code existant.

   Nous nous intéressons aux approches d'implémentation basées sur la composition ou *Composition-based* [3], et non celles basées sur les annotations *Annotation-Based* (voir chapitre 1).

   Par conséquent, notre travail a nécessité l'utilisation d'un *Code Composer* pour dériver le produit. Cette activité de dérivation s'est donc effectuée par FeatureHouse composer [32], qui génère des variantes SMA automatiquement en composant des artefacts réutilisables relatifs à une configuration valide. Les artefacts proviennent donc des deux niveaux d'implémentation de la ligne de produits multi-agents.

4.2.3 Positionnement de notre approche MAS-PL

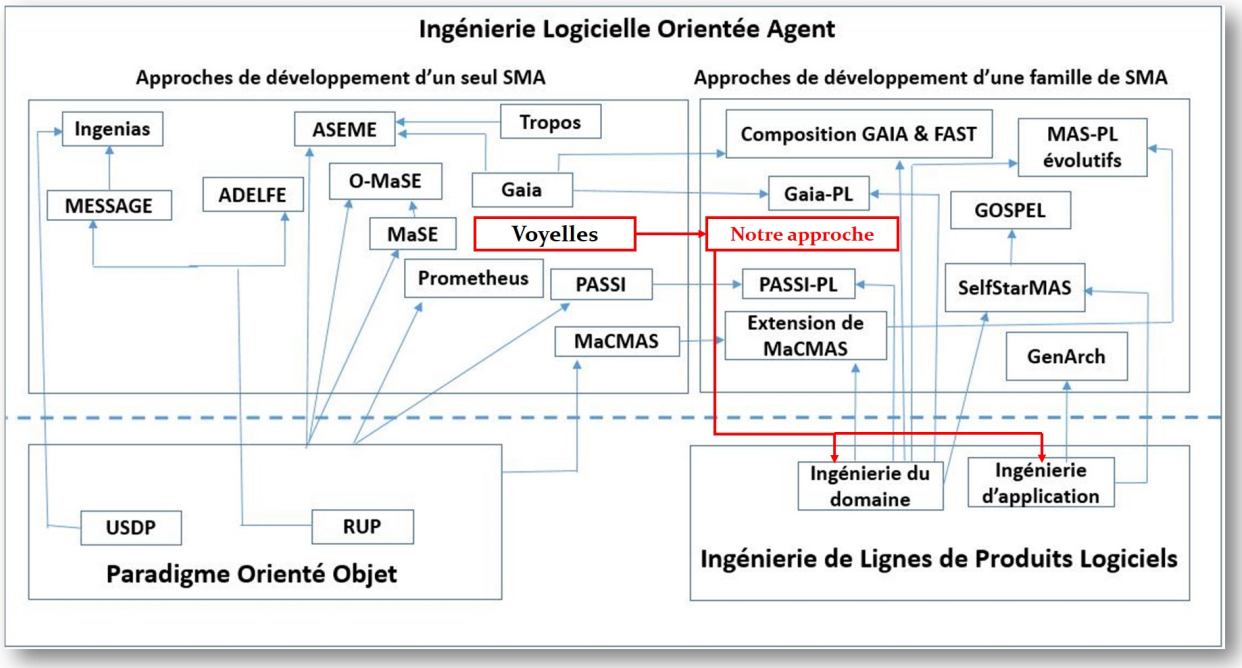


FIGURE 4.2 – Positionnement de notre approche MAS-PL dans l'ingénierie des SMA

Dans cette sous section, nous positionnons notre approche MAS-PL parmi les approches d'ingénierie des SMA, en mettant en avant les différences et similitudes qu'elle présente en comparaison avec les approches MAS-PL développées dans le chapitre précédent, et qui s'articulent autour des points suivants :

1. ***Contribution au niveau de la phase d'ingénierie du domaine :***

   En comparaison avec les approches MAS-PL existantes présentées dans le chapitre 3, notre approche (voir la figure 4.2), se positionne parmi celles couvrant l'ensemble des deux phases d'ingénierie de lignes de produits logiciels (voir la figure 4.1), et dont la contribution principale se situe au niveau de la phase d'ingénierie du domaine : spécification des similitudes et de la variabilité ; et implémentation d'artefacts indépendants du domaine d'application.

2. ***Basée sur les similitudes et la variabilité de plusieurs méta-modèles :***

   Contrairement aux approches qui se basent sur un seul méta-modèle limité à la méthodologie multi-agents choisie, notre approche offre une flexibilité au concepteur, qui peut choisir durant une configuration possible du produit multi-agents à dériver, les concepts

dont il aurait besoin en fonction de la méthodologie qu'il adoptera ; et ce afin de répondre aux besoins fonctionnels et non fonctionnels.

3. ***Classée parmi les extensions MAS-PL :***

   Étant donné que notre approche s'appuie intégralement sur le processus général de LdP logiciels SPLE (Software Product Line Engineering), elle ne se place pas parmi les approches construites de zéro. De plus, étant donné que nous n'avons pas composé ce processus avec une autre méthodologie ou fragment de méthodologie multi-agents, notre approche ne se positionne pas non plus parmi les compositions. Certes, le fait d'utiliser des concepts issus de plusieurs méta-modèles afin de construire notre modèle de caractéristiques des SMA, peut donner lieu à une confusion laissant penser à une éventuelle composition. Nous composons des concepts sous formes de points communs et variables représentés au sein d'un unique modèle (FM générique), et non les méthodologies correspondantes ni leur fragments.

   Contrairement aux approches MAS-PL qui représentent également des extensions, la notre ne se manifeste pas sous forme de stéréotypes pour l'annotation de la variabilité comme PASSI-PL [93], ou encore des patrons réutilisables spécifiant la variabilité des rôles d'agents [101], mais plutôt sous forme d'une nouvelle couche du FM ayant la particularité d'être indépendante du domaine d'application, et donc réutilisable en servant de point de départ pour toute méthode dont les concepts ont été couverts par notre délimitation du domaine durant notre analyse (voir les chapitres 2 et 5).

   Nous avons procédé à l'extension de chacune des dimensions de l'approche Voyelles [28] (habituellement utilisées pour la description d'une seule version du système), afin qu'elles puissent représenter les similitudes et la variabilité du domaine des SMA. Comme nous l'avons mentionné plus haut, notre approche permet de représenter l'approche voyelles sous forme d'approche de LdP multi-agents, qui a été étendue grâce à l'analyse de chacune des dimensions voyelles qu'elle englobe.

4. ***Modèlisation avec le modèle de caractéristiques FM :***

   Comme nous avons pu le voir dans le chapitre précédent, les approches MAS-PL n'utilisent pas toutes le même type de modélisation pour la ligne de produits multi-agents. Peña et al. [104] utilisent des modèles de caractéristiques FM (feature model) pour la représentation des buts (Goals) des agents sous formes de caractéristiques réutilisables.

   D'autres approches comme nous avons pu le voir, telles que GOSPEL [24] utilisent plusieurs types de modèles comme CVL (Commonality and Variability Language) et FM, avec la possibilité de passage d'un type de modèle à l'autre.

   Notre approche se situe parmi les approches utilisant les modèles de caractéristiques pour les raisons que nous expliquerons plus tard dans ce chapitre.

5. ***Une couche supplémentaire pour le modèle de caractéristiques FM :***

   L'originalité de notre extension, est qu'elle est la première à proposer deux couches au niveau du modèle de caractéristiques et de son implémentation. Notre approche permet de décrire une LdP multi-niveaux (deux couches), afin d'élargir l'étendue de la réutilisation des concepts, et de couvrir ainsi un plus large spectre de domaines d'applications. Par exemple, PASSI-PL [93] qui étend la méthodologie PASSI [76, 113], couvre le concept de *AIP (Agent Interaction Protocol)*, qui représente le concept de protocoles d'interaction ; mais la variabilité liée à un tel protocole n'est pas spécifiée par le modèle proposé au niveau de l'approche. A l'inverse, notre approche offre la possibilité de choisir le protocole d'interaction souhaité parmi un sous ensemble des protocoles d'interactions les plus utilisés (contract net, enchères etc.).

   De plus, aucun concept n'est associé au niveau organisationnel des extensions MAS-PL que nous avons analysées. Par exemple, le concept de *normes* n'est pas couvert par l'approche PASSI-PL, car ce même concept n'est pas couvert par la méthodologie orientée agent PASSI [76, 113]. Un autre exemple révèle qu'aucune variabilité liée aux permissions associées aux ressources n'est spécifiée.

   Gaia-PL [101], ne couvre pas non plus la variabilité de certains concepts du domaine ; tels que le concept d'agent *MentalState* et plus précisément le concept *Belief* supporté par le méta-modèle Ingenias [57].

   Il en est de même pour les caractéristiques environnementales, telles que celles *d'environnements physiques* qui ne sont pas supportées.

   A l'inverse, notre approche MAS-PL supporte une telle variabilité liée à tous ces concepts entre autres, que cela soit pour le concept de *Normes*, que nous avons repris du modèle Moise+ [21] ; d'environnement physique *physical World* issu du modèle AGRE [130], ou encore le concept *Belief* faisant partie du modèle BDI [58], et que nous retrouvons également au niveau du méta-modèle d'Ingenias [57]. Contrairement aux approches MAS-PL citées, la notre permet donc de choisir via une configuration, l'ensemble des caractéristiques désirées au niveau du produit multi-agents à dériver.

6. ***Choix d'une approche multi-agents flexible :***

   Notre choix s'est porté sur l'approche voyelles afin d'organiser notre modèle de caractéristques des SMA pour plusieurs raisons que nous expliquerons plus tard. La principale, est que cette approche, permet de décrire de façon indépendante chacune des dimensions d'un SMA, et c'est ce que nous avons fait. De plus, cette approche n'impose pas l'utilisation de modèles précis pour chacune des voyelles relatives aux dimensions : d'agent, d'environnement, d'interaction et d'organisation. Il a donc été possible d'inclure les modèles que nous souhaitions intégrer, afin d'effectuer un premier pas vers une approche

d'ingénierie multi-agents à base de lignes de produits logiciels.

De plus, notre approche peut être étendue en incluant d'avantages de modèles que nous n'avons pas inclus dans notre délimitation du domaine. A l'inverse, PASSI-PL, l'extension de MaCMAS ou encore GAIA-PL obligent de garder les modèles préconisés par chacune des approches multi-agents respectives qui ont été étendues. Par exemple, notre approche permet de développer des agents délibératifs avec la possibilité de paramétrer le choix de leurs caractéristiques. Il en est de même pour le paramétrage de la perception de l'agent qui peut être passive ou active, tandis que les approches existantes n'autorisent pas cela, du moins ces changements demanderaient plus d'effort de la part des concepteurs et développeurs, car elles ne proposent pas des implémentations de départ. En effet, aucune implémentation réutilisable n'est associée à ces caractéristiques comme nous l'avons fait en intégrant des artefacts réutilisables des protocoles d'interaction, d'environnements physiques ou encore d'architectures délibératives.

De plus, la spécificité du modèle de caractéristiques au domaine d'application ne permet pas une réutilisation étendue. Notre approche, contrairement à GOSPEL [24], ou encore SelfStar [110, 111, 120], n'est pas spécifique à un seul et unique domaine d'application.

4.3 Ingénierie du domaine

Cette section décrit de façon détaillée, les activités de la phase d'ingénierie du domaine de notre approche. Nous donnons pour chaque étape, l'ensemble des choix que nous avons fait.

4.3.1 Analyse indépendante du domaine d'application

Durant cette activité nous devions répondre à plusieurs questions. Il a fallu d'abord choisir un type de modèle pour représenter les similitudes et la variabilité des familles de SMA. En second lieu, nous avons procédé à une délimitation du domaine afin de déterminer les caractéristiques principales d'un SMA à intégrer au sein de notre modèle. Nous devions ensuite, trouver un moyen d'organiser les caractéristiques principales au sein de notre modèle. Et enfin, nous avons analysé les similitudes et la variabilité des SMA en termes de concepts provenant de notre délimitation du domaine. Dans ce qui suit, nous expliquons chacun des points cités.

4.3.1.1 Choix d'une modélisation pour la ligne de produits multi-agents

Afin de choisir une approche de modélisation adéquate de la ligne de produits multi-agents, nous avons comparé les principales approches issues de l'ingénierie de lignes de produits logiciels (voir le chapitre 1, et utilisées dans les approches MAS-PL (voir le chapitre 3).

Le tableau 4.2, regroupe les éléments relatifs à l'ensemble de ces modèles. Nous y mentionnons les aspects en relation avec leur structure, les notations qu'ils utilisent, ou encore sur quels concepts sont-ils centrés ("caractéristiques" ou "points de variations").

Principes	**OVM**	**CVL**	**FM**
Centré caractéristiques	-	-	+
Centré points de variations et variants	+	+	-
Un seul type de modèle	-	-	+
Plusieurs types de modèles	+	+	-
Syntaxe FODA	-	+	+
Standard OMG	-	+	-

TABLE 4.2 – Tableau comparatif des approches de modélisation des lignes de produits logiciels

Nous avons opté pour le modèle de caractéristiques FM (Feature Model), pour plusieurs raisons. Mise à part qu'elle soit l'approche la plus utilisée du domaine, la raison principale est que contrairement aux autres approches, les FMs s'articulent autour du concept clé de "caractéristiques" pour la spécification des similitudes et de la variabilité. Comme nous avons pu le voir au niveau du chapitre1, le concept de caractéristiques représente l'une des préoccupations principales autour de laquelle s'articule l'ingénierie de lignes de produits logiciels.

De plus, nous pensons que le fait qu'un FM regroupe l'ensemble des caractéristiques au sein d'un seul et unique modèle de la ligne de produits multi-agents, peut promouvoir son adoption par la communauté des SMA. A l'inverse, les modèles OVM (Orthogonal Variability Model), et CVL(Commonality and Variability Language) comme représenté dans le tableau 4.2, nécessitent plus de connaissances des modèles que leur approche emploie (Vspec, Vspec trees, etc.). Par exemple, il est primordial de spécifier les relations entre les modèles de variabilité et les modèles de base construits sous forme de liens. Il faut également, passer par une distinction entre les points de variations et les variantes dès la spécification. Sans oublier, que le modèle OVM tel que nous l'avons constaté au niveau du chapitre 1, représente chaque point de variation de façon séparée. C'est pourquoi, il faut effectuer un travail supplémentaire afin d'ajouter les contraintes de dépendances par la suite (voir la figure 1.10).

De plus, afin d'arriver à des modèles résolus du domaine (variantes), il est nécessaire d'exécuter une transformation basée sur le modèle de variation, le modèle de résolution et le modèle de base du domaine d'application considéré. Tandis qu'avec le modèle de caractéristiques, le choix de configuration relative à une variante souhaitée ne nécessite pas de passer par tous ces modèles.

Plus encore, même si CVL (Commonality and Variability Language) est un standard OMG (Object Management Group), il permet de représenter des modèles similaires aux FMs comme nous l'avions précisé au niveau du chapitre 1. Les deux types de modèles utilisent la syntaxe FODA (Feature Oriented Domain Analysis) [33], et par conséquent adoptent la même structure hiérarchique. Seules les notations graphiques diffèrent comme nous avons pu le voir au niveau des exemples présentés dans le chapitre 1.

4.3.1.2 Choix de l'approche VOYELLES

Une fois le type de modélisation choisi, il a fallu identifier les principales caractéristiques d'un SMA. Pour rappel, pendant l'activité d'analyse du domaine, les approches MAS-PL présentées dans le chapitre précédent, ont considéré uniquement un domaine spécifique (voir l'exemple de la figure 3.6).

Par conséquent, ces modèles ne sont donc pas forcément réutilisable pour d'autres domaines d'application. C'est pourquoi, nous avons donc pensé à construire un modèle de similitudes et de variabilité, qui lui serait réutilisable par un plus grand nombre d'applications possibles.

C'est ainsi, que nous avons décidé d'analyser le domaine multi-agents (modèles, méthodologies etc.), et d'identifier les principales caractéristiques du domaine.

Nous avons donc sélectionné un sous-ensemble de méthodologies et modèles. Comme préconisé en ingénierie de lignes de produits logiciels, pour une première version d'un modèle de caractéristiques, la délimitation du domaine peut contenir un sous-ensemble du domaine en question.

Pendant notre analyse, nous avons constaté que certaines approches étaient plus englobantes que d'autres en termes de concepts, et ce sont celles-ci qui nous intéressent. Parmi elles, l'approche VOYELLES [28,29] que nous avons choisie afin de nous permettre d'organiser l'ensemble des caractéristiques d'un SMA au sein de notre modèle.

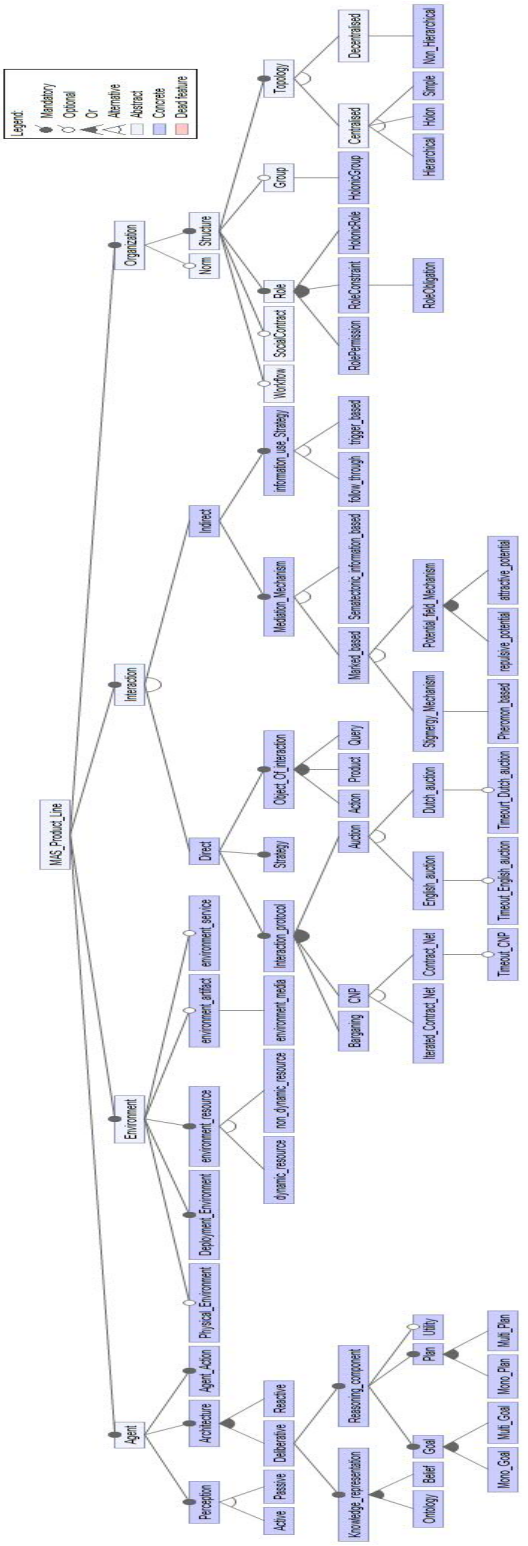


FIGURE 4.3 – Le modèle des caractéristiques indépendantes du domaine d'application pour les SMA

Notre choix s'est porté sur l'approche VOYELLES pour les raisons suivantes :

1. La première raison, est que cette approche a la particularité de proposer une séparation de chacune des dimensions d'agents (voyelle A), d'environnement (voyelle E), d'interaction (voyelle I) et d'organisation (voyelle O) pour la description d'un SMA.
2. La seconde raison, est qu'elle permet de rechercher de manière indépendante chacun des éléments descriptifs de chacune des dimensions du système. Il est donc possible de rechercher : la description d'architectures internes aux entités de traitement du système ; les éléments dépendants du domaine pour structurer l'interaction externe entre les entités du système ; les éléments pour structurer l'interaction interne entre les entités du système ; et les éléments pour structurer des entités au sein du SMA .
3. L'avant dernière raison, est que même si les dimensions d'un SMA peuvent être recherchées de manière indépendante, l'approche permet de former un modèle cohérent. C'est ce que nous avons obtenu en regroupant l'ensemble des résultats dont ceux issus de l'analyse des similitudes et de la variabilité de chaque dimension (voir le chapitre 5)
4. La dernière raison mais pas la moindre, est que cette approche n'impose pas l'utilisation de modèles spécifiques pour chacune des dimensions. Ce qui permet de ne pas restreindre notre représentation des caractéristiques relativement aux concepts de chacune des voyelles de l'approche. C'est ce point, qui nous a permis d'étendre l'approche voyelles en la représentant sous forme d'approche de lignes de produits multi-agents.

Une fois l'approche voyelles choisie, nous avons déduit les principales caractéristiques du domaine des SMA, les avons analysées en termes de points communs et variables. Notre objectif, étant de faire un pas vers un *modèle de caractéristiques générique / Generic MAS FM* ; qui est représenté dans la figure 4.3. Les détails de notre analyse, et de la construction de ce modèle sont présentés dans le prochain chapitre.

4.3.2 Implémentation indépendante du domaine d'application

Durant cette activité, nous avions un ensemble de choix à faire. Nous avons commencé par choisir un type d'implémentation pour la ligne de produits multi-agents, ainsi que l'ensemble d'outils nécessaires. Nous avons ensuite, cherché des implémentations réutilisables pour chacune des dimensions de l'approche voyelles, qui correspondent aux caractéristiques. Dans ce qui suit, nous donnons et expliquons chacun de ces choix, accompagnés d'une description des artefacts réutilisables que nous proposons au sein de notre approche.

4.3.2.1 Choix du type d'implémentation de la ligne de produits multi-agents

Une fois les principales caractéristiques des SMA déterminées (premier niveau du FM) sous forme de points communs et variables ; il a fallu choisir un type d'implémentation de la ligne de produits.

Pour rappel, tel que présenté dans le chapitre 1, il y-a trois types d'implémentations possibles pour la ligne de produits : *annotatrices, compositionnelles, et combinatoires.*

Après la comparaison de ces approches d'implémentation, notre choix s'est porté sur les approches compositionnelles pour plusieurs raisons. Mise à part qu'elles soient les plus utilisées, elles permettent une séparation modulaire des implémentations, sous forme de composants. Ainsi qu'une implémentation de la variabilité plus facile à tracer, à visualiser et à répartir par le développeur de la ligne. Le code source est associé à plusieurs unités de composition, qui implémentent les caractéristiques séparément.

4.3.2.2 Choix des outils d'implémentation

Une fois le type d'implémentation choisi, nous devions sélectionner les outils pour réaliser une telle implémentation.

Nous avons donc, utilisé l'environnement de développement feature IDE [131] sous éclipse.

Feature IDE, est open source, et permet d'intégrer toutes les phases du développement logiciel, axé autour des caractéristiques (feature-oriented software development).

De plus, comme nous avions opté pour une implémentation compositionnelle, nous avons utilisé le composeur de code FeatureHouse [32].

Pour l'implémentation multi-agents nous l'avons principalement réalisée avec JADE (Java Agent Development Environment) ; qui est un environnement logiciel permettant de développer des systèmes multi-agents d'agents conformément aux spécifications FIPA (Foundation for Intelligent Physical Agents) [3]. Par conséquent, JADE, garantit la conformité aux normes grâce à un ensemble complet de services et d'agents.

4.3.2.3 Description de l'activité d'implémentation du domaine multi-agents

La mise en œuvre des systèmes multi-agents, repose souvent sur les frameworks et outils existants. Les artefacts correspondent aux composants fournis par ces frameworks et outils. Dans ce contexte, il existe également plusieurs implémentations des modèles issus de l'ingénierie des SMA. Par exemple, MadKit [90] est l'implémentaion du modèle AGR [20].

Le choix de ces plate-formes dépend d'une part de l'expertise du développeur, et d'autres part des implémentations qu'il doit réaliser, et par conséquent des concepts associés.

3. http ://fipa.org/

Rappelons que, la principale question de recherche qui se cache derrière cette activité que nous proposons, est directement liée au problème de développement d'applications multi-agents similaires, que nous cherchons à faciliter . L'introduction de cette activité a pour but de faciliter le développement en minimisant l'effort d'implémentation de la ligne de produits multi-agents.

Comme nous avons pu le remarquer durant notre analyse des approches MAS-PL, un des inconvénients est que l'activité d'implémentation de la ligne de produits multi-agents ne propose aucun point de départ au développeur. De plus, les implémentations obtenues par le biais de ces approches MAS-PL, ne sont pas réutilisables pour d'autres domaines d'application. Par conséquent, le développeur devra fournir d'avantages d'efforts pour aboutir à une implémentation spécifique au domaine d'application.

Une dernière de nos motivations à vouloir intervenir à ce niveau d'implémentation, est que suite à de nouveaux besoins de la ligne de produits multi-agents, il est possible que le développeur se retrouve contraint à rechercher par lui même des implémentations réutilisables. Ces dernières, serviraient à satisfaire de nouveaux besoins d'une nouvelle instance d'un produit multi-agents (voir l'approche *réactive* de la figure 1.4 du chapitre 1). Par exemple, supposons que la première version de la ligne de produits multi-agents supporte n produits SMA. Suite à de nouveaux besoins en termes de produits, elle nécessiterait l'introduction d'un nouvel SMA, et par conséquent de passer à $n+1$ produits.

Supposons que cette mise à jour, impliquerait d'introduire de nouvelles fonctionalités au sein du modèle, dont certaines seraient indépendantes du domaine d'application. Par exemple, si jusqu'ici les n produits étaient à base d'interactions indirectes, et que le produit n+1, nécessiterait la mise en place d'interactions directes, il faudrait donc, en premier introduire ces nouvelles fonctionnalités au sein du modèle FM, et en second implémenter/rechercher une implémentation réutilisable des protocoles d'interactions (ex : vente aux enchères).

C'est pour toutes ces raisons, que nous avons pensé à permettre au développeur de réutiliser plus facilement les éventuelles implémentations dont il aurait besoin, et qui sont directement associées (mapping) au même niveau de notre modèle indépendant du domaine d'application (FM générique). En d'autres termes, la variabilité du domaine multi-agents qui est spécifiée au niveau de la première couche de notre FM, est reliée à la variabilité relative à l'implémentation du domaine en question. Ces implémentations, permettraient de réduire l'écart existant entre la variabilité des concepts du domaine, et leur implémentation. La variabilité liée à ces implémentations indépendantes du domaine d'application, n'étant pas supportée par les approches MAS-PL ; notre approche permettrait au développeur, de réutiliser ces implémentations, en les raffinant pendant la dernière activité de la phase d'ingénierie du domaine.

L'ensemble des artefacts réutilisables que nous proposons n'est pas statique. Il peut être enrichi en considérant d'autres frameworks non inclus dans notre travail de recherche, telque JaCaMo [132, 133].

Il est important de préciser, que nous ne proposons pas une variabilité liée aux plate-formes d'implémentation (variabilité technique) ; comme le passage d'une implémentation sous JADE à une implémentation sous Madkit. Nous proposons plutôt, une variabilité liée aux implémentations des concepts, que nous réutilisons à partir de ces plate-formes au niveau de JADE. Comme le passage d'une implémentation d'un agent doté d'une perception passive, à un agent doté d'une perception active. Nous visons les fonctionnalités principales du système, telles que les implémentations de modèles d'architectures délibératives.

En effet, ces artefacts réutilisables permettent d'implémenter les caractéristiques issues du premier niveau de notre modèle obtenu durant la première activité de notre approche. Par conséquent, ces implémentations correspondent directement aux artefacts relatifs à l'approche VOYELLES [28, 29], et sont représentés dans la figure 4.4. Ils permettent d'implémenter les principales fonctionnalités des systèmes multi-agents.

Il est important de souligner que certaines caractéristiques sont dites abstraites, et ne sont donc pas implémentées. En effet, certains concepts présents lors des phases de spécification, et de conception du système disparaissent au niveau le plus bas relatif à son implémentation. Par exemple, le concept *Role* disparaît au niveau de l'implémentation. Un rôle, va se traduire par un comportement de l'agent qui respecte l'ensemble des spécifications liées à un tel rôle, que ce soit au sein de son organisation ou encore durant les différentes interactions.

Chacun des artefacts que nous avons réutilisés et implémentés, pour chacune des dimensions d'agent, d'environnement, d'interaction et d'organisation, sont présentés dans ce qui suit :

Artefacts d'Agents (ex: packages BDI)	Artefacts d'Interactions (ex: packages pour Protocols)
Artefacts d'Environnement (ex: packages EIS)	Artefacts d'Organisation (ex: packages Structure)

]

FIGURE 4.4 – Les artefacts réutilisables génériques implémentant les caractéristiques multi-agents génériques

— **Artefacts de l'agent :**
Les artefacts réutilisables de l'agent permettent d'implémenter les caractéristiques *Agent* de notre modèle. Ces artefacts sont réutilisables indépendamment du domaine d'application. Ils correspondent aux architectures internes, ainsi qu'aux modules de perception et d'action de l'agent.
Notre approche, offre la possibilité de réutiliser les implémentations du modèle BDI [58], que nous avons présenté au niveau du chapitre 2 (voir la figure 2.3), et duquel nous avons

extrait les principales caractéristiques.
Nous avons cherché une implémentation réutilisable pour les différents concepts tels que *Belief*, *Goal* et *Plan*, ainsi que les concepts de *perception*, et *d'action* de l'agent. Il existe plusieurs platformes basées sur l'ensemble de ces concepts et plus précisément le modèle BDI. Dans ce contexte, nous avons réutlisé APLTK (Agent Programming Toolkit) qui représente une boîte à outils pour la programmation orientée Agent [129],et qui permet de faciliter l'implémentation du modèle BDI. Cet outil offre également une compatibilité des SMA avec les plate-formes 2APL, GOAL et Jason.
Il est important de souligner, que certains concepts n'apparaissent pas au niveau des caractéristiques, mais sont présents au niveau des implémentations. Ceci est du à la granularité des caractéristiques choisies. Rappelons, que le choix de ces caractéristiques, et de leur variabilité sont une hypothèse sur la façon avec laquelle les membres d'une famille multi-agents peuvent être distingués.
Par exemple, le concept *"capability"* n'apparaît pas au niveau des caractéristiques de notre modèle, mais il contribue à l'implémentation de la *perception*, et de *l'action*, et apparaît donc au niveau du code source.
En effet, la *perception* représente une aptitude (*capability*) sensorielle de l'agent. Tandis que le concept *d'action* , est considéré comme étant une aptitude effectrice de l'agent.
Ces principaux concepts réutilisables sont représentés au niveau la figure 4.5, et sont associés à leurs implémentations.
Nous avons également implémenté la variabilité des algorithmes de wooldridge [58], ainsi que les fonctions principales du modèle BDI. Nous avons ensuite, associé cette variabilité aux caractéristiques *Goal* et *Plan*.
La variabilité entre les algorithmes implémentés, concerne la cardinalité de l'ensemble B, D et I. Comme illustré dans la figure 5.2, que nous allons présenter dans le chapitre suivant, le premier algorithme représente la version la plus simple du BDI. Elle correspond, à des agents que nous qualifions dans notre modèle de mono-Goal (un seul Goal existant au niveau de la pile des Goals). Alors que le deuxième algorithme, concerne les agents qui doivent atteindre plusieurs objectifs. Nous qualifions ce type d'agents de multi-Goal (au moins deux Goals au niveau de la pile de Goals). Quant à la variabilité observée par rapport au dernier algorithme, elle propose la possibilité d'exécuter des plans dynamiques.
En effet, l'algorithme propose d'enrichir le plan de la bibliothèque pendant l'exécution. Cependant, ce type d'algorithme n'est pas utilisé en pratique pour deux raisons : (i) Le coût de son implémentation est élevé, et (ii) il est plus fréquent et simple d'utiliser en pratique une librairie de plans établie au préalable. L'agent devra ainsi sélectionner le plan approprié pour l'accomplissement de ses Goals. Ces implémentations sont directe-

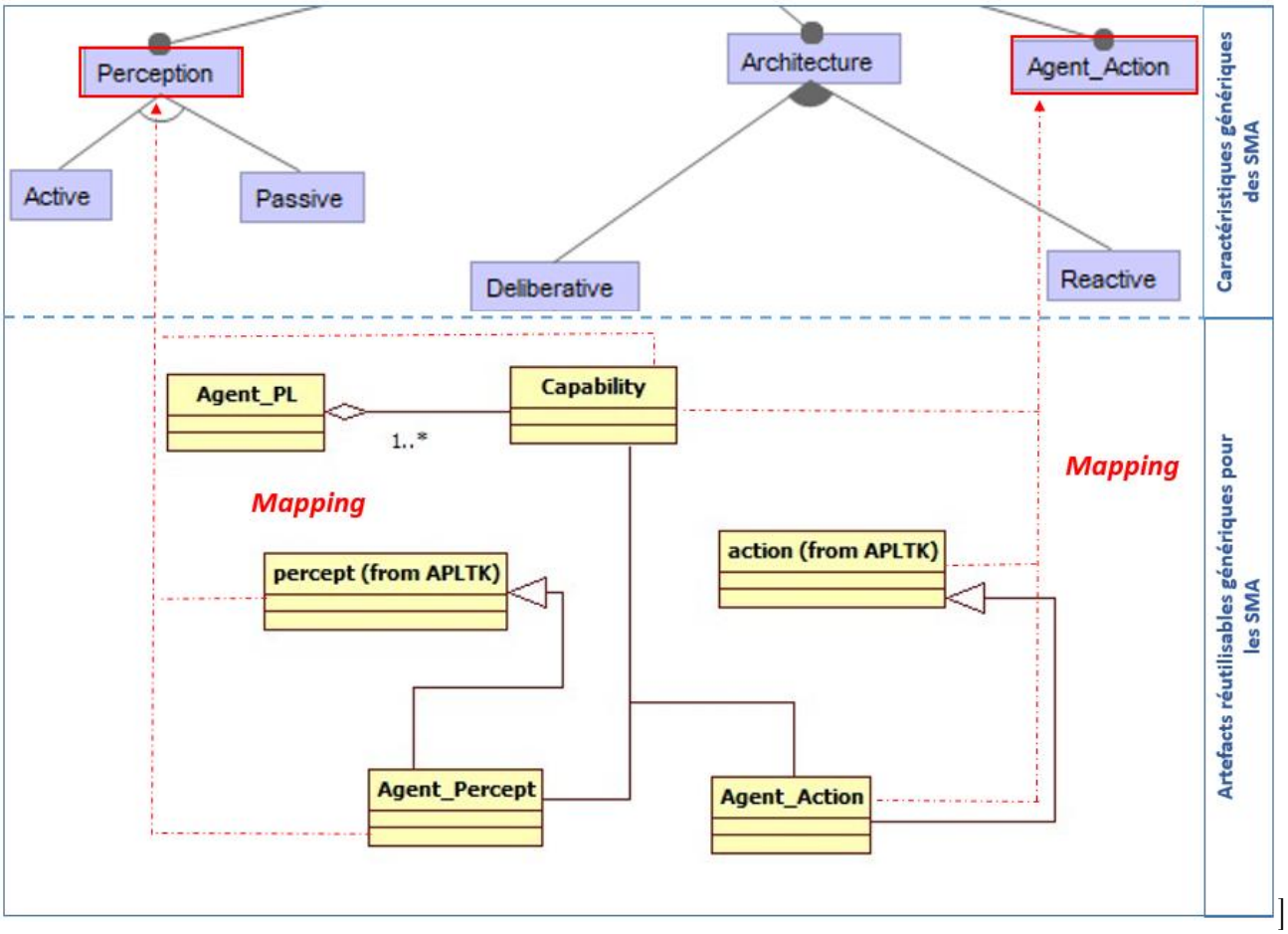


FIGURE 4.5 – un exemple d'association (mapping) entre les caractéristiques de perception et d'action de l'agent avec les artefacts réutilisables qui les implémentent

ment associées aux caractéristiques suivantes : *Mono_ Goal*, *Multi_ Goal*, *Mono_ Plan* et *Multi_ Plan*.

La plupart des implémentations des modèles BDI utilisent les fonctions *brf* (fonction de révision de croyance), *ogf* (fonction de génération d'option), *filter* et *asf* (fonction de sélection d'action). Cependant, la variabilité de l'algorithme a un impact sur ces fonctions. Par exemple, l'implémentation BDI *Mono _ Goal*, représentée dans la figure 5.2, montre que les agents n'utilisent ni la fonction *ogf*, ni la fonction *filter*. Par conséquent, les agents *Mono _ Plan*, n'ont pas besoin d'une fonction de sélection de plan.

Nous avons donc procédé à l'implémentation de la fonctionnalité *Mono _ Goal* avec des artefacts réutilisables, composés des fonctions proposées par le premier algorithme de Wooldridge [58], comme suit :

1. *gettingPercepts ()* : pour exécuter le get-next percept, et accèder aux perceptions,
2. *createBeliefsFromPercept ()* : pour créer et mettre à jour les croyances d'agents, cette

fonction correspond à la fonction *brf ()* ;

3. *checkAll-BeliefsForInsertGoal ()* : pour que la délibération de l'agent corresponde à la fonction *deliberate ()*,
4. *performActionGoal ()* : pour sélectionner un plan et l'exécuter.

La figure 4.6 illustre ces implémentations réutilisables, ainsi que leurs associations (mapping) aux caractéristiques concrètes, qu'elles implémentent. La caractéristique *Mono_ Goal* est implémentée par le premier algorithme de Wooldridge [58], tandis que la caractéristique *Multi_ Goal* est implémentée par le second algorithme.
Il est important de souligner que seules les caractéristiques concrètes (concrete features) du FM générique, sont implémentées. Les caractéristiques abstraites n'ont aucune correspondance (mapping) avec les artefacts d'implémentation réutilisables, car elles n'apparaissent que de façon abstraite lors de la spécification du système multi-agents. De ce fait, certains concepts disparaissent au niveau d'implémentation, comme la caractéristique *BDI*.

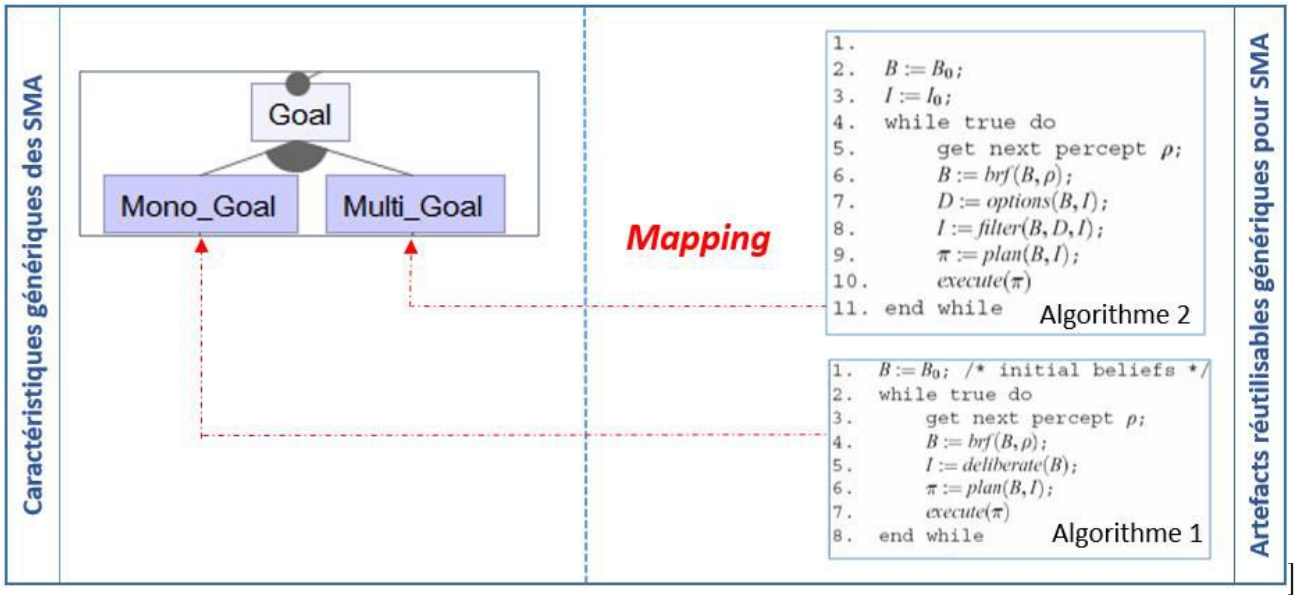


FIGURE 4.6 – un exemple d'association (mapping) entre des caractéristiques d'agents et les artefacts réutilisables qui les implémentent

— **Artefacts de l'environnement :**
Les agents faisant partie de l'environnement ; ce dernier leur fournit par exemple des moyens, ou des ressources pour permettre la communication.
Dans ce contexte, différents modèles d'environnements d'agents ont été proposés (voir le chapitre 5). Certains de ces modèles ont également été implémentés, et peuvent par conséquent être réutilisés durant cette activité proposée par notre approche.

En exemple, le modèle A & A (Agent et Artefact) [55] est intéressant, puisqu'il est pris en charge par CArtAgO [134]. Ce même modèle a été intégré au sein de jaCaMo [132, 133], qui n'a pas été construit en réutilisant uniquement CArtAgO, mais aussi les plate-formes Jason [135] (pour l'implémentation d'agents autonomes) ; et également Moise [136] (pour la programmation d'organisations d'agents).
Selon Behrens et al. [129], la normalisation de l'environnement devrait être réalisée, en séparant les concepts de moyens de communication, et de ressources. C'est pourquoi, ils ont proposé une approche générique pour connecter des agents à l'environnement, en considérant des artefacts environnementaux réutilisables, et aussi indépendants que possible d'une structure environnementale spécifique. C'est ainsi, qu'ils ont proposé l'API EIS (Environment Interface Standard) [129] ; que nous réutilisons. Celle-ci représente les artefacts environnementaux susceptibles d'être réutilisables, et que nous avons associés (mapping) à la fonctionnalité *Physical _ Environment* de notre modèle. Un des avantages de EIS, est qu'il réduit l'effort d'implémentation pour se connecter aux environnements d'agents (par exemple, les environnements de jeu Unreal Tournament UT3 et UT2004 et le Concours multi-agents).
Une de nos motivations principales pour la réutilisation de l'API EIS, est qu'elle a elle même réutilisé un maximum de concepts issus de plusieurs méta-modèles d'environnements d'agents. Ces derniers se sont avérés être des éléments génériques intéressants pour notre approche. De plus, les différents principes des plate-formes 2APL [137], GOAL [138] et Jason [135] ; ont été regroupés de façon à être réutilisés.

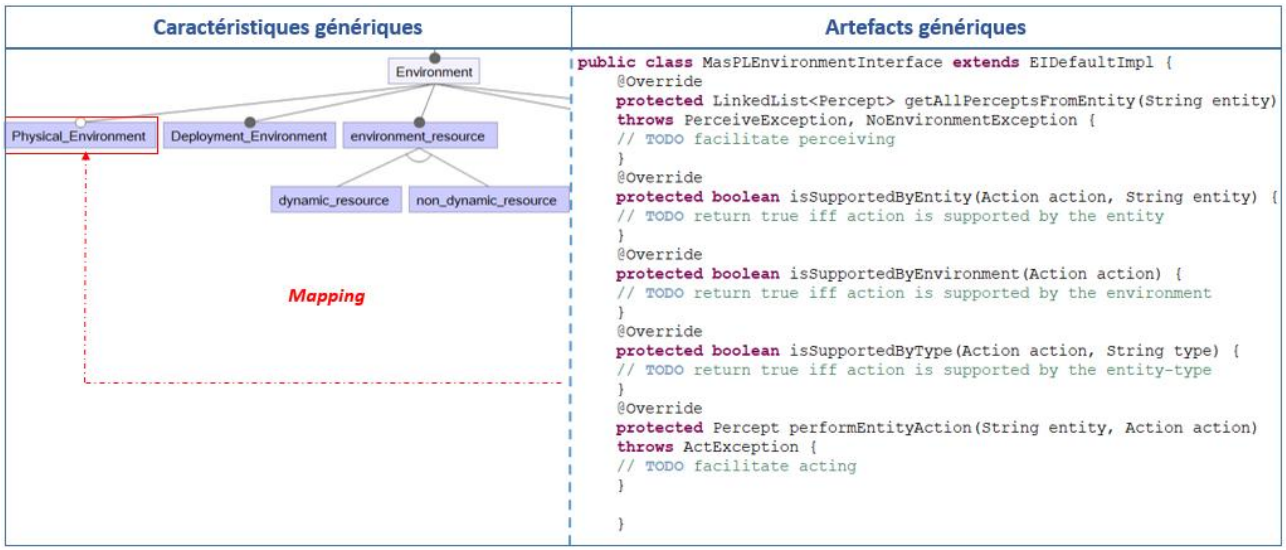


FIGURE 4.7 – un exemple d'association entre une caractéristique d'environnement d'agents et un extrait de son implémentation

Le fait d'être construit sur la base de 2APL [137], qui est l'extension de 3APL, a permis entre autres d'augmenter les choix possibles (variabilité) pour l'environnement. Par exemple, 2APL a permis d'intégrer des évènements (events), ainsi que de nouvelles actions possibles au sein de l'environnement. Ces dernières, permettent aux agents d'agir au sein d'un environnement physique, en déclenchant l'exécution de nouveaux plans, et d'actions communicatives. Sans oublier, l'introduction de nouvelles règles permettant de réparer des plans qui auraient échoué durant leur exécution au sein de l'environnement. De plus, un autre point intéressant, est que EIS permet d'avoir une connexion bidirectionnelles entre des plate-formes multi-agents arbitraires, et des environnements physiques arbitraires.
La figure 4.7, présente un exemple de réutilisation de EIS que nous avons associé à la caractéristique d'environnement physique au niveau de notre modèle. La classe *MasPLEnvironmentInterface* hérite de la classe abstraite *EIDefaultImpl*. Cette dernière, représente l'implémentation par défaut qui est requise par toute interface d'environnement physique.

— **Artefacts d'interactions :**

Caractéristiques/features génériques	**artefacts génériques**
interaction_protocol	Les protocoles d'interactions sont implémentés par les différents rôles compris dans le protocole. Chacun de ces rôles aura un comportement différent en fonction du protocole qu'il implémente. De ce fait les comportements auront comme point commun la classe *FSMBehaviour* fournie par JADE représentant le comportement de chacun des rôles d'interactions sous forme de machine à états.
CNP	Le protocole contract net est implémenté par quatre différentes classes. Nous réutilisons d'une part les classes *ContractNetInitiator* et *ContractNetResponder* fournies par JADE ; et d'autre part nos propres classes *ContractNetInitiatorRole* et *ContractNetRespondertRole* qui héritent des deux classes précédentes. Le but étant d'étendre les classes fournies par JADE afin de permettre d'y associer une variabilité de stratégies et de services pour faciliter la réutilisation au sein de la LdP.

Caractéristiques/features génériques	**artefacts génériques**
Strategy	La classe *Strategy* telle qu'elle est décrite au sein de l'ontologie ONTOPRO que nous réutilisons au sein de notre modèle.
Evaluation_Strategy	La classe *EvaluationStrategy* telle décrite au niveau de l'ontologie ONTROPO.
Proposition_Strategy	La classe *PropositionStrategy* telle décrite au niveau de l'ontologie ONTROPO.
Object_of_interaction	La classe *Service* qui représente l'objet sur lequel porte l'interaction. Tel cité précédement il peut représenter par exemple un produit, une requête ou encore une action.
Product	Les deux classes *ProductProviding* ainsi que *Product*. ces deux artefacts implémentent les objets d'interactions relatifs à la fourniture de produits quelconques.
Action	Les deux classes *ActionExecution* ainsi que *Action*. ces deux artefacts implémentent les objets d'interactions relatifs à l'exécution d'action.
Query	Les deux classes *QueryCheck* ainsi que *Query*. ces deux artefacts implémentent les objets d'interactions de type query ou requête de vérification.

TABLE 4.3 – Un exemple d'artefacts réutilisables implémentant les caractéristiques génériques de notre modèle

Les artefacts d'interaction réutilisables concernent principalement les protocoles d'interaction. Les artefacts d'interaction standardisés par FIPA sont disponibles pour le programmeur grâce à des abstractions pour développer un système multi-agents conforme à FIPA. Certaines implémentations FIPA réutilisables sont fournies par des frameworks, tels que JADE qui propose une implémentation des rôles du CNP.

Dans les différents cas d'application présentés dans le chapitre 6, nous expliquerons en détail comment réutiliser les artefacts multi-agents des protocoles d'interaction.

Nous présentons dans ce qui suit comment nous avons réutilisé les artefacts du protocole CNP (Contract Net Protocol), en reprenant le CNP-Initiator et CNP-Participant (ou responder) qui représentent les rôles relatifs à l'implémentation du protocole d'interaction. Nous avons réutilisé les classes JAVA fournies par l'API de JADE [4] afin de pouvoir les utiliser au sein de notre approche.

Un exemple d'association (mapping) entre les artefacts génériques d'interactions, et

4. `http://jade.tilab.com/doc/api/`

les caractéristiques correspondantes est représenté dans le tableau 4.3. Cet exemple, donne pour chacune des caractéristiques génériques, l'ensemble des artefacts qui l'implémentent.

L'extrait du diagramme de classe relatif à cette réutilisation est représenté dans la figure 4.8. La description de chacune de ces classes est comme suit :

1. La classe *ContractNetInitiator (from JADE)* :

   La classe issue de l'API de JADE permet d'implémenter le rôle de l'initiateur des protocoles Fipa-Contract-Net or Iterated-Fipa-Contract-Net.

   Cette classe hérite de la classe FSMBehaviour de JADE. Elle représente une implémentation générique du comportement de l'initiateur du contract net, elle peut aussi bien être utilisée pour des interactions 1 :1 ou 1 :N. Cette classe générique peut être raffinée notament pour les méthodes de type Handler pour chaque type d'acte communicatif : handlePropose (), handleRefuse (), handleNotUnderstood ().

   L'initiateur sollicite des propositions des autres agents en envoyant un message de type CFP qui spécifie l'action à réaliser, et si nécessaire les conditions relatives à son exécution. L'implémentation de la méthode prepareCFP() doit retourner un vecteur de messages à envoyer (un seul message à envoyer à plusieurs récepteurs).

   Les participants peuvent ainsi répondre en envoyant un message PROPOSE qui comporte les préconditions établies pour l'action en question, comme par exemple le prix. Dans le cas de refus, ils envoient un message de type REFUSE, ou encore un NOT-UNDERSTOOD dans le cas de problème de communication. JADE propose de gérer cette catégorie de réponses possibles par le biais de la méthode *handleAllResponses()*. Dans le cas où le programmeur désire gérer tous ces actes de communication séparément, il peut utiliser les méthodes suivantes : handlePropose(), handleRefuse(), handleNotUnderstood().

   L'initiateur peut évaluer toutes les propositions faites et décider quelle offre accepter. Ceci qui correspond à une stratégie d'évaluation peut se faire soit au niveau de la méthode *handlePropose()* c'est à dire à la réception d'une proposition, ou encore se faire une fois tous les messages de ce type reçus en utilisant la méthode *handleAllResponses()*. Dans le cas d'acceptation d'une proposition, le programmeur peut ignorer et ne pas attendre le reste des propositions en faisant appel à la méthode *skipNextResponses()* lors de l'attente d'un message de type PROPOSE, ce qui permettra de passer au prochain état.

   Une fois que les participants ayant reçu un ACCEPT-PROPOSAL complétent leur action, ils peuvent enfin répondre par un INFORM relatif au résultat de l'action réalisée, ou bien s'il y a un problème ils répondent par un FAILURE. Ces messages sont

traités au sein de la méthode handleAllResultNotifications(). Ils peuvent également être traités séparément au sein des méthodes : handleInform(), handleFailure().

2. La classe *ContractNetResponder (from JADE)* :

   Cette classe étend la Classe SSContractNetResponder qui implémente le comportement du rôle participant du contract net. Il est également implémenté par JADE.

   La classe représente l'implémentation du comportement du rôle Participant du protocole fipa-contract-net. Ce comportement permet ainsi de répondre aux CFP reçus par le rôle initiateur.

   Dès l'arrivée d'un message qui correspond au template passé en paramètres du constructeur, la méthode prepareResponse() est exécutée en incluant la réponse souhaitée telle que l'envoi d'un message PROPOSE.

   Dans le cas où l'initiateur accepte la proposition et de ce fait le participant recevrait un message ACCEPT-PROPOSAL, la méthode prepareResultNotification() sera exécutée et devra retourner un message comprenant le résultat de la notification, soit un INFORM ou un FAILURE. Cependant, dans le cas où l'initiateur rejette la proposition en envoyant un message REJECT-PROPOSAL au participant, ce dernier devra exécuter la méthode handleRejectProposal() afin de terminer le protocole.

3. La classe *ContractNetInitiatorRole* :

   Représente le comportement relatif au rôle d'interaction de l'initiateur. Cette classe étend la classe *ContractNetInitiator (from JADE)*, afin de permettre au rôle la définition de ses stratégies d'évaluation et services relatifs à l'objet de l'interaction.

4. La classe *ContractNetResponderRole :*

   Représente le comportement relatif au rôle d'interaction du participant. Cette classe étend la classe *ContractNetResponder (from JADE)*, afin de permettre au rôle la définition de ses stratégies de proposition et services relatifs à l'objet de l'interaction.

5. La classe *AbstractStrategy :*

   Représente le concept de Stratégie tel qu'il été présenté dans l'ontologie ONTOPRO que nous avons réutilisée (voir Chapitre 4).

6. La classe *EvaluationStrategy :*

   Représente les stratégies d'évaluation, comme celles utilisées par le rôle d'initiateur du contract-Net afin de décider comment évaluer une proposition faite par un participant. Elle représente l'un des conceptS que nous avons réutilisé à partir de l'ontologie ONTOPRO (voir chapitre 4).

7. La classe *ProposeStrategy :*

   Représente les stratégies de proposition, comme celles utilisées par le rôle du participant du contract-Net afin de décider de faire une proposition une fois qu'il est

sollicité par un appel d'offre. Elle représente l'un des concepts que nous avons réutilisé à partir de l'ontologie ONTOPRO (voir chapitre 4).

8. La classe *AbstractService :*

   Représente, comme nous l'avons développé dans le chapitre précédent, l'objet de l'interaction qui peut être en relation avec l'exécution d'une action,ou de fourniture de produits ou encore une requête de vérification.

9. *Classe Product :*

   La classe du domaine représentant un produit compris dans les différentes transactions d'achat/vente. Le reste des classes est décrit dans le chapitre précédent, et représente l'ensemble des classes que nous réutilisons à partir de l'ontologie de protocoles d'interactions ONTOPRO.

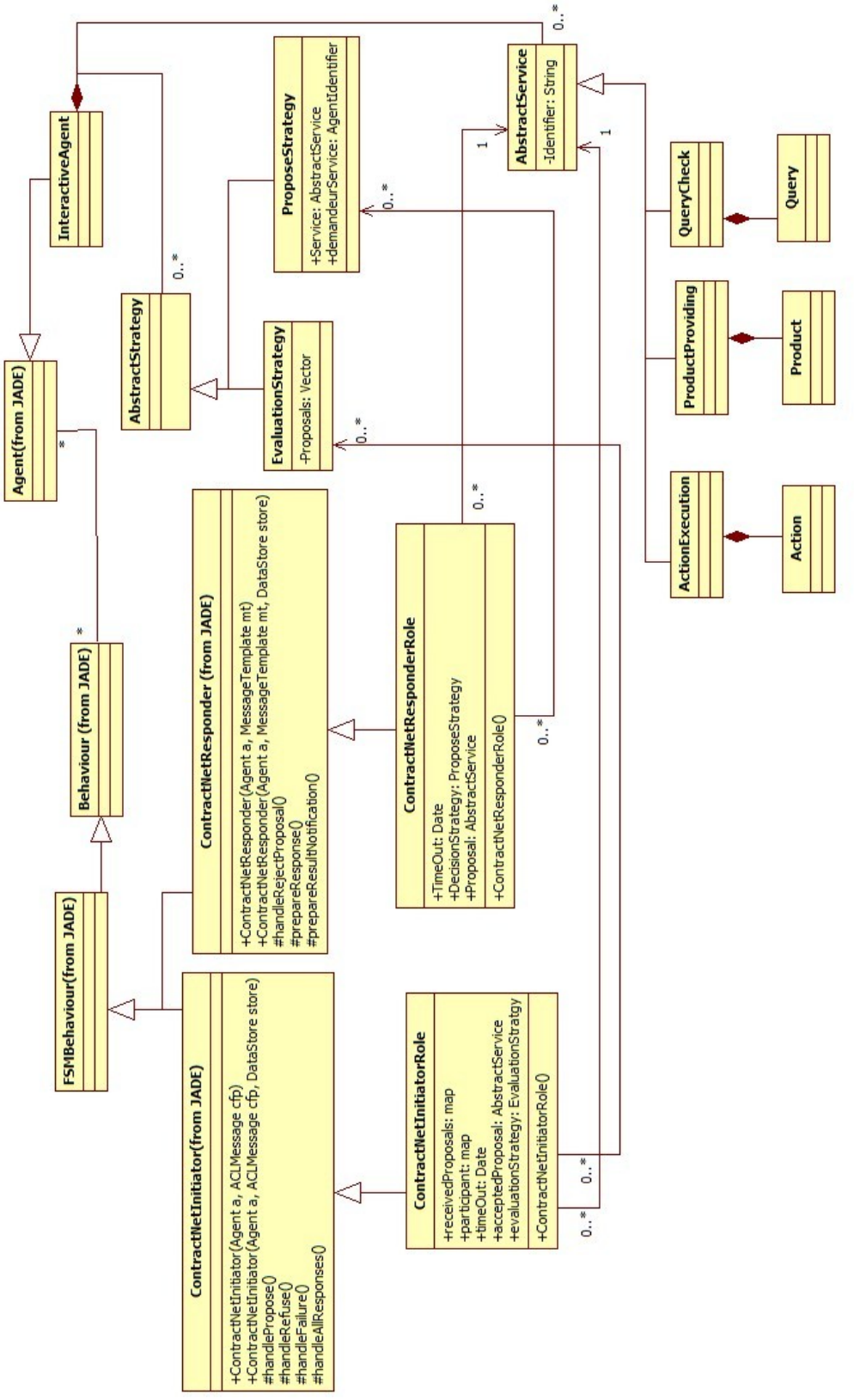


FIGURE 4.8 – Extrait du diagramme de classes relatif aux interactions directes d'agents

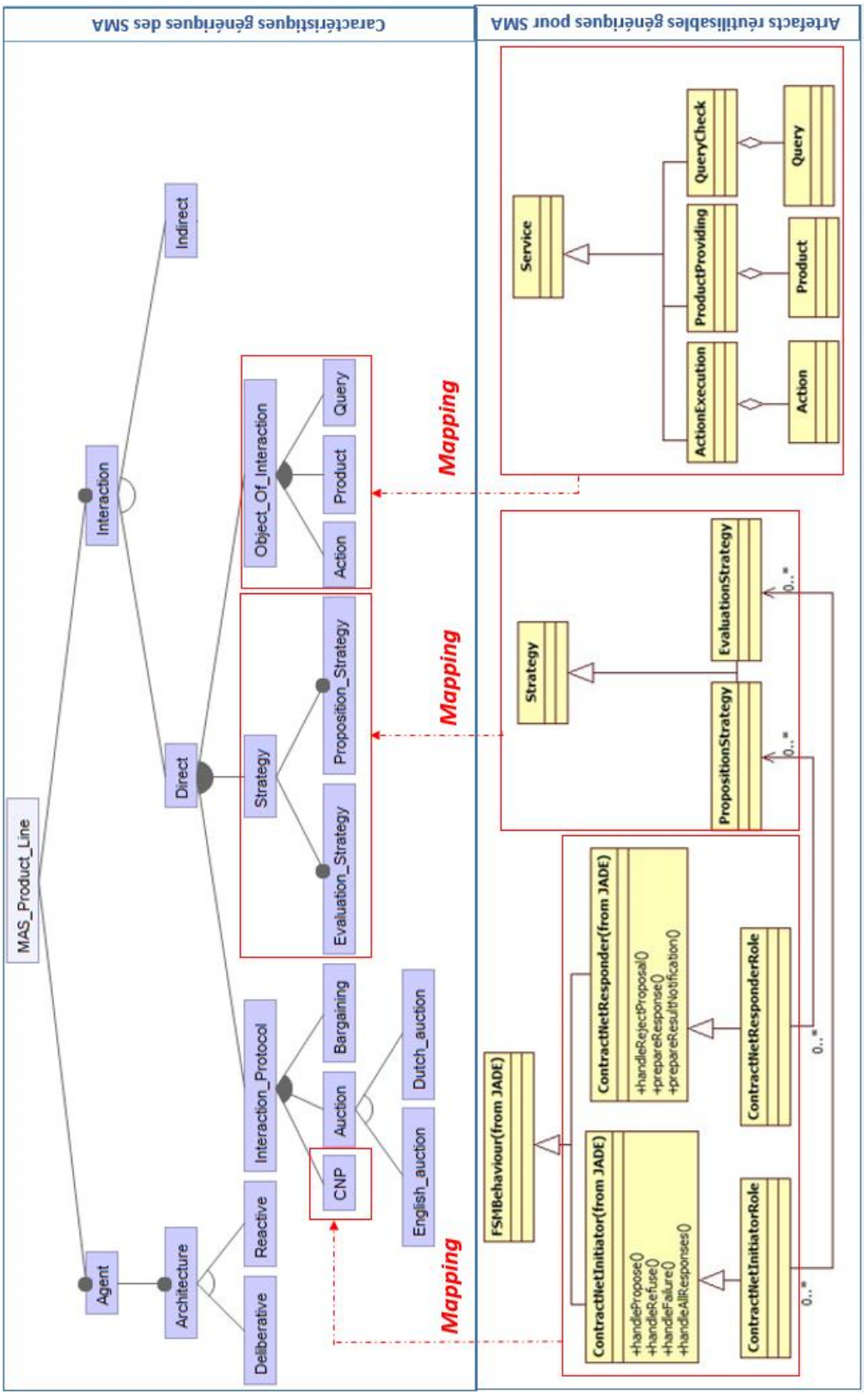


FIGURE 4.9 – un exemple de mapping entre des caractéristiques d'interactions multi-agents génériques et les artefacts génériques les implémentant

La figure 4.9 représente l'association (mapping) entre l'ensemble des classes présentées ci-dessus avec l'ensemble des features génériques associées.

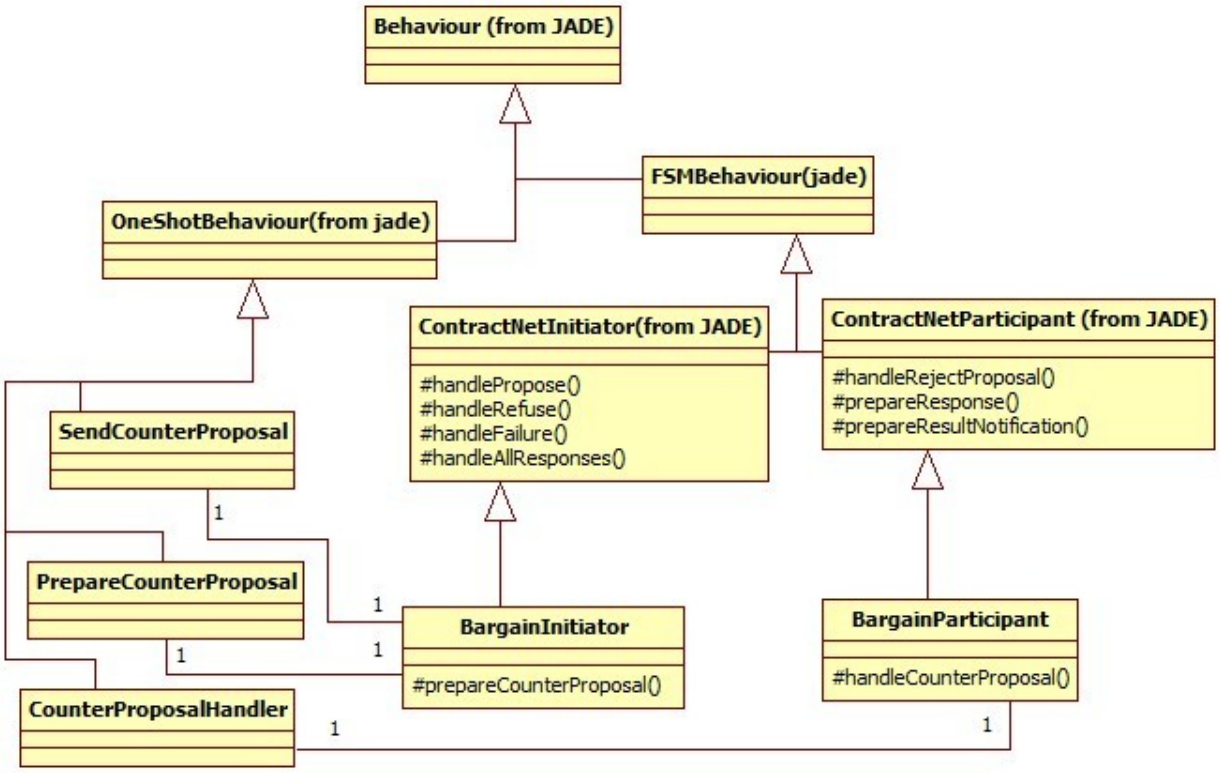


FIGURE 4.10 – Extrait du diagramme de classes relatif aux interactions directes d'agents pour réaliser un Bargaining

Un autre extrait du diagramme de classes représentant des artefacts d'interactions directes réutilisables est représentée dans la figure 4.10. L'ensemble de ces classes est relatif au protocole de *Bargaining*. Ceci est réalisé en raffinant le protocole d'interaction du contract net, de façon à ce que le comportement du rôle relatif au responder lui permette d'évaluer des nouvelles demandes CFP et que celui de l'initiateur ait la possibilité de négocier le prix proposé en faisant une autre demande représentant un prix inférieur à celui proposé par le responder. Ceci peut être réalisé par un Bargain protocol. Nous allons considérer la version la plus simple du bargaining. Comme notre approche est incrémentale, il est possible à tout instant de raffiner le feature model générique ainsi que l'ensemble des artefacts l'implémentant. Le processus de raffinement du modèle consiste à l'ajout de la classe BargainInitiator et BargainParticipant qui raffinent respectivement les classes ContractNetInitiator et ContractNetParticipant. Ces nouvelles classes sont décrites ci-dessous :

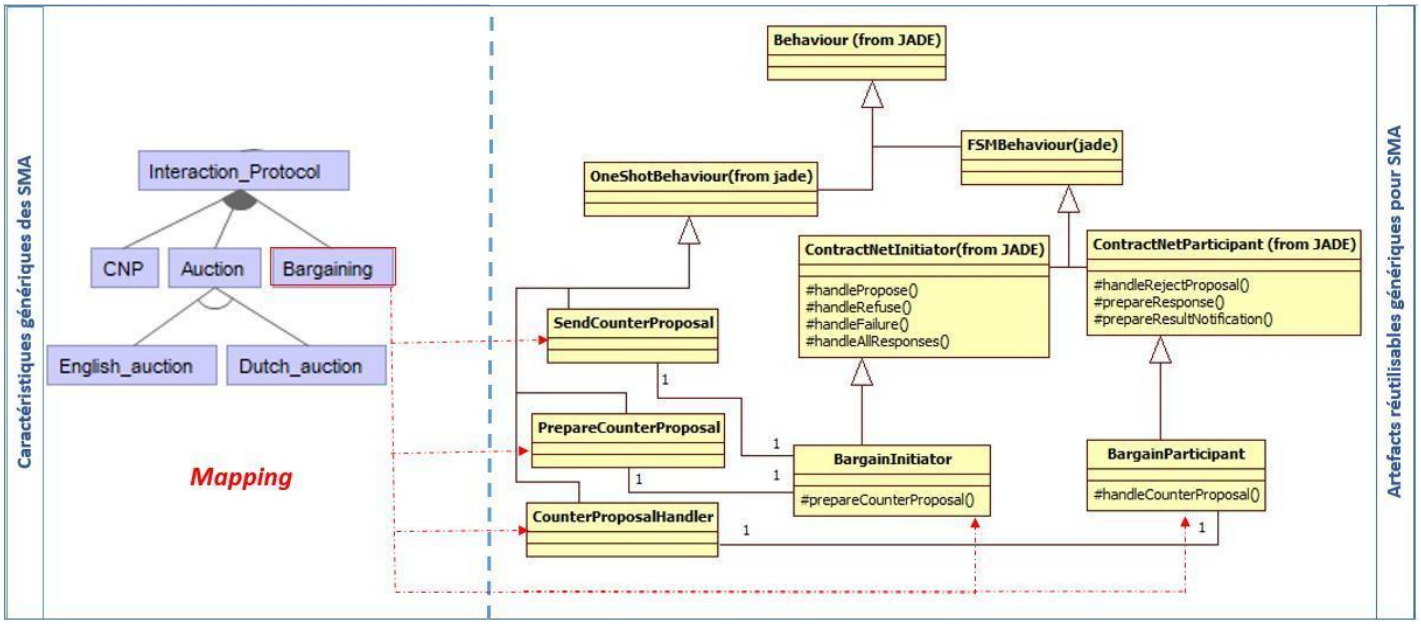


FIGURE 4.11 – un exemple de mapping entre des caractéristiques d'interactions multi-agents génériques et les artefacts génériques implémentant le Bargaining

1. *Classe BargainInitiator :*

   Cette classe étend la classe fournie par JADE implémentant un comportement optionnel permettant d'inclure des états et transitions supplémentaires lui fournissant la possibilité de réitérer le processus en émettant un nouvel CFP. FIPA a introduit l'acte counter-proposal qui peut être utilisé à cette fin comme représenté dans la figure. De ce fait la nouvelle transition permet d'envoyer un counter-proposal dans ce cas de figure. Cette nouvelle transition relative non pas au rejet de l'offre comme cela est fait dans le contract net, mais à la demande de négocier, est réalisée par les comportements PepareCounterProposal et SendCounterProposal représentant des comportements OneShotBehaviour (comportement générique fourni dans JADE). Ces comportements sont introduits par l'ajout de nouveaux états et transitions correspondants.

   En plus des deux états transitoires les méthodes permettant au programmeur d'effectuer des raffinements supplémentaires par le biais des comportements transitoires additionnels sont ajoutées au niveau de cette classe telle que la méthode prepareCounterProposal(). Celle-ci sera exécutée en même temps que le comportement transitoire y faisant référence et dans ce cas celui représenté par la classe PrepareCounterProposal.

2. *Classe PepareCounterProposal :*

Cette classe représente le comportement permettant de préparer les messages représentés par counter-proposal. Comme c'est un comportement qui ne s'exécute qu'une seule fois il hérite de la classe de comportements OneShotBehaviour fournie par JADE. Cette classe représente un état transitoire de la classe qui précède. Ce comportement n'est exécuté que si l'agent désire négocier l'offre.

3. *Classe SendCounterProposal :*

   Cette classe représente l'état transitoire qui suit le précédent. Il permet d'envoyer les messages avec performative counter-proposal. Ce comportement s'exécute également tel que spécifié par un comportement de type OneShotBehaviour.

4. *Classe BargainParticipant :*

   Cette classe étend la classe implémentant le ContractNetParticipant en ajoutant un nouvel état sous forme de comportement et transition supplémentaires permettant de traiter la réception de messages de type counter-proposal. Ceci étant réalisé par le comportement générique CounterProposalHandler de type OneShotBehaviour.

5. *Classe CounterProposalHandler :*

   Comme son nom l'indique, cette classe représente le comportement transitoire qui permet la gestion des messages counter-proposal reçus de la part d'agents désirant négocier la proposition faite par un agent vendeur. Il est associé à la méthode HandleCounterProposal() existante au niveau de la classe générique BargainParticipant. Des stratégies associées permettent de décider d'accepter ou pas de faire une nouvelle offre.

La figure 4.11 représente l'association(mapping) réalisée entre les artefacts précédents ainsi que les features ou caractéristiques génériques les implémentant.
Le dernier exemple que nous présentons est celui des artefacts pouvant être réutilisés lors des enchères.
Pour cet exemple nous proposons de réutiliser un protocole d'enchères Anglaises. La spécification FIPA de ce protocole d'enchère est représentée dans la figure 2.10 connu sous le nom de English auction. Le protocole stipule que le commissaire-priseur cherche à trouver le prix du marché d'un bien en proposant initialement un prix inférieur à la valeur du marché supposée et ensuite augmenter progressivement le prix. Chaque fois que le prix est annoncé, le commissaire aux enchères attend pour voir si les acheteurs exprimeront leur volonté de payer le prix proposé.
Dès que l'un des acheteurs indique qu'il accepte le prix, le commissaire priseur émet un nouvel appel d'offres avec un prix incrémenté. La vente aux enchères continue jusqu'à ce que les acheteurs ne manifestent plus aucun intérêt pour le prix proposé. Si le prix du dernier accepté par un acheteur dépasse le prix de réserve du client (privé), le bien

est vendu à cet acheteur pour le prix convenu. Si le dernier prix accepté est inférieur au prix de la réservation, le bien n'est pas vendu.
Les appels du commissaire-priseur, exprimés par envoi de message cfpc, sont multicast pour tous les participants dans la vente aux enchères. Les propositions soumises par les enchérisseurs concernent principalement le processus d'appel d'offres. En réponse à un cfp à soumettre des offres pour acheter un bon X, une proposition serait quelque chose de l'ordre : "Je propose que le niveau d'enchère soit augmenté pour acheter au prix Z et j'affirme que je peux payer le prix Z pour X " Cela permet au commissaire-priseur d'être convaincu que l'enchérisseur peut payer le prix sans s'engager à le payer jusqu'à ce que le commissaire aux enchères le demande spécifiquement auprès de X (au prix Z) de l'enchérisseur gagnant. La transaction d'enchères est complétée par la suite avec un seul gagnant.
Ces deux comportements respectifs sont représentés par les deux rôles réutilisables EnglishAuctionInitiator et EnglishAuctionParticipant.
Le protocole d'enchères Anglaises présente plusieurs variabilités relatives aux stratégies possibles de l'enchérisseur (bidding strategies). Ces stratégies sont génériques et peuvent permettre dans certains cas à aider l'enchérisseur à gagner l'enchère, comme par exemple lui permettre de connaître la vraie valeur de l'objet relatif à l'enchère.
Leurs stratégies leur permettent plus ou moins d'avoir, dans ce sens, plus d'informations relatives aux enchères. Dans [139] par exemple, ils proposent de voir les effets des paramètres d'enchères sur les résultats d'enchères en ligne, en explorant les modèles de décision des enchérisseurs. Dans [140], les auteurs ont proposé une comparaison de différentes stratégies spécifiques aux protocole d'enchères anglaises et proposent de soutenir le modèle de décision d'agents relatifs aux stratégies par des prédictions concernant le prix de fin d'enchère. L'agent ADBA (Automated Dynamic Bidding Agent) qu'ils proposent permet de faire des prédictions grâce à la conception des différentes stratégies d'enchères anglaises basées sur le comportement des enchérisseurs. Nous proposons une réutilisation générique des différentes stratégies d'enchérisseurs. Il existe en effet différentes catégories de comportements. Dans la littérature il peut exister différentes catégories d'enchérisseurs : (i) Evaluators, (ii) Participators, (iii) Opportunists, (iv) Snipers, (v) Unmaskers, et (vi) Shill [141] [142] [140].
Les *Evaluators* par exemple ont une idée claire de l'évaluation de la valeur de l'article, et de ce fait ils émettent une seule et unique enchère dès le début de cette dernière. Quant aux *unmaskers*, ils vont émettre plusieurs enchères de façon continue sur une courte période sans enchères intermédiaires lors de la progression de l'enchère en cours.
Dans [141] par exemple les auteurs proposent une taxonomie des enchérisseurs en présentant les aspects importants pour la conception des stratégies d'enchères, tel que l'instant

de la première et dernière enchère émise. Ces informations nous permettent de raffiner d'une part le feature model générique et d'autre part les artefacts correspondants. D'autres travaux comme [142] ont proposé le premier algorithme de détection d'enchérisseurs de type Shill. Ces derniers ont regroupé les aspects spécifiques aux comportements de ce type d'enchérisseurs.En se basant sur les travaux ci-dessus nous proposons de regrouper la variabilité des artefacts des stratégies en fonction des paramètres suivants :

1. *Le nombre d'enchères émises :*

   une seule ou plusieurs enchères peuvent être émises. Certaines stratégies proposent de placer une seule enchère. Dans ce cas l'unique enchère proposée se fera soit à la fin ou au début. Par exemple des enchérisseurs peuvent placer leur seule enchère à la fin soit les 5 dernières secondes(*Mystical*) ou les 5 dernières minutes (*Sturdy*). Ils considèrent les enchérisseurs qui placent plusieurs enchères comme *Strategic* car ils émettent une offre en fonction des propositions établies au préalable.

2. *L'instant où l'enchère est émise :*

   Certaines stratégies se caractérisent par l'émission d'enchères à la fin, au début ou tout au long de l'enchère.Ce type d'enchérisseurs ont plus ou moins d'informations concernant l'évaluation de l'article en question. En effet, tel que les enchérisseurs de type *Evaluateurs* qui ont une idée du prix final pouvant être atteint à la fin de l'enchère pourront émettre leur proposition dès le début de cette dernière.

3. *La nature de l'attitude de l'enchérisseur :*

   Les attitudes que peut avoir un enchérisseur sont de deux catégories il peut soit choisir de négocier progressivement le prix du bien, ou choisir de maximiser ses chances de gagner l'enchère en proposant un prix fort dès le départ. Dans [143] par exemple, les auteurs proposent de varier les protocoles d'interactions en incluant les enchères Anglaises, Allemandes et celles de Vickery. Ils définissent des contraintes d'enchérisseurs tels que le degré de désespoir d'un agent et son désir de faire une bonne affaire. Le niveau de désespoir d'un agent stipule que si l'agent est *Désespéré* d'obtenir l'objet, il se lance très rapidement dans des prix élevés pour s'assurer qu'il maximise ses chances de l'obtenir. Le contraire de ce comportement est celui d'un agent qui cherche à faire une bonne affaire à un prix avantageux, il commence donc à offrir à un prix très bas pour lancer finalement son évaluation privée quand il a très peu de temps restant. L'ensemble des considérations relatives au temps restant, aux autres enchères restantes, au désir d'obtenir une bonne affaire et le niveau de désespoir sont ici appelés les contraintes d'appel d'offres.

4. *Prédiction du prix final ou non :*

   Une stratégie peut ou pas inclure la possibilité de faire des prédictions sur le prix

final d'une enchère afin de l'aider dans le gain de l'enchère. Ce type de comportement se fait en analysant des enchères déjà réalisées dans le passé par exemple. Dans ce contexte, l'agent ADBA proposé dans [140] utilise des stratégies basées sur le calcul d'un montant d'enchère à un moment particulier concernant les agents enchérisseurs qui placent une ou plusieurs enchères vers la fin de l'enchère.

L'ensemble des classes servant d'artefacts réutilisables sont représentées dans la figure 4.12. Leur description est donnée ci-dessous :

1. *Classe EnglishAuctionInitiator :* Cette classe représente le comportement de l'initiateur d'enchères Anglaises. Le rôle hérite de la classe FSMBehavior de JADE permettant de définir un comportement par machine à états finis.
2. *Classe EnglishAuctionParticipant :* Cette classe hérite également de la classe FSMBehavior de JADE et permet de définir le comportement associé au rôle du participant à l'enchère Anglaise.
3. *Classe ProposeBidBehavior :* Cette classe représente un comportement de type OneShotBehavior de JADE permettant de représenter l'un des états du rôle du participant à l'enchère. Ce comportement lui permettra de prendre la décision relativement à sa stratégie d'enchère afin de faire une proposition.
4. *Classe BiddingStrategy :* Représente la stratégie de proposition d'enchères associée au rôle participant de l'enchère. En effet, la valeur de l'offre émise par les participants dépend de la stratégie adoptée. Dans ce contexte d'autres paramètres sont associés comme l'instant auquel la décision devra être prise.
5. *Classe TimingStrategy :* Cette stratégie utilisée par le participant aux enchères lui permet de faire une offre en fonction du temps.

   Par exemple dans certaines stratégies d'enchères la proposition peut se faire une seule fois à la fin, au début ou tout au long de l'enchère.
6. *Classe BidPlacementStrategy :* Représente les stratégies décisionnelles relativement au placement de l'offre. Par exemple le nombre d'enchères émises peut varier en fonction de la stratégie adoptée par l'enchérisseur. D'autres enchérisseurs peuvent décider de placer plusieurs enchères émises en fonction des propositions établies au préalable.
7. *Classe AuctionEvaluationStrategy :* Représente la stratégie d'évaluation d'enchères réalisée par l'initiateur de l'enchère soit le commissaire priseur.

Pour réaliser le raffinement du feature model générique en conséquence, il est nécessaire de raffiner les features *Interaction_ Protocol* avec une feature générique *Auction* qui comporte deux features génériques filles aternatives *English_ Auction* et *Dutch_ Auction* ; de

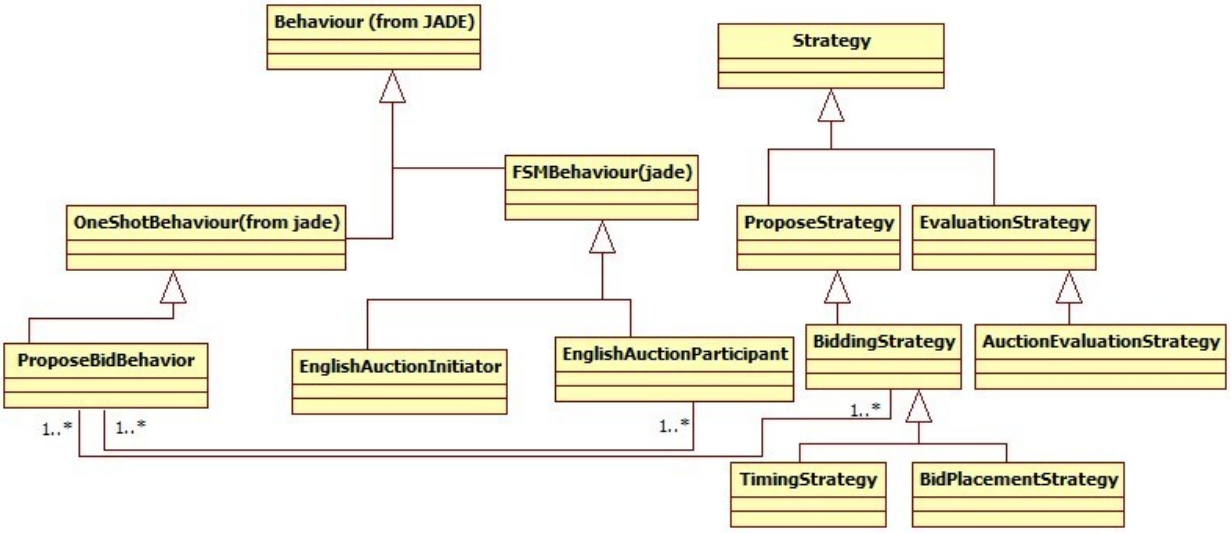


FIGURE 4.12 – Extrait du diagramme de classes relatif aux interactions directes d'agents pour réaliser une enchère Anglaise

raffiner également la feature *Evaluation_Strategy* avec *Auction_Evaluation_Strategy* et enfin la feature *Proposition_Strategy* avec *Bidding_Strategy*. Le mapping entre ces features et artefacts est représenté dans la figure 4.13. Les classes *BiddingStrategy*, *TimingStrategy* et *BidPlacementStrategy* sont associées à la caractéristique générique *Bidding_Strategy*. Certaines classes relatives aux comportements d'agents pour les enchères sont associées à la feature *English_Auction* comme la classe *EnglishAuctionParticipant*, tandis que d'autres sont associées à la caractéristique *Auction* car elles représentent des classes communes aux enchères, telle que la classe *ProposeBidBehaviour* qui représente un état possible du comportement du participant à l'enchère.

— **Artefacts d'organisation :**
Il existe à ce jour, plusieurs implémentations des modèles organisationnels. Ces implémentations vont d'une vue organisationnelle minimale (commune), à une vue plus étendue (variabilité). Par exemple, le modèle le plus simple que nous avons analysé (voir les chapitres 2 et 5), est celui de AGR [20]. Ce dernier, a été implémenté par Madkit (Multi-Agent Development Kit) [90], qui est une plate-forme générique de conception et d'exécution de systèmes multi-agents. Cette plate-forme, ne s'articule ni autour d'une architecture d'agents précise, ni d'un protocole d'interaction spécifique [144].
Les concepts de *rôles*, et de *groupes* servant de base à la modélisation, ainsi que la conception du système, ont servi également comme principe d'architecture de la plate-forme. Cependant, tous ces concepts que nous avons intégré au sein de notre modèle générique

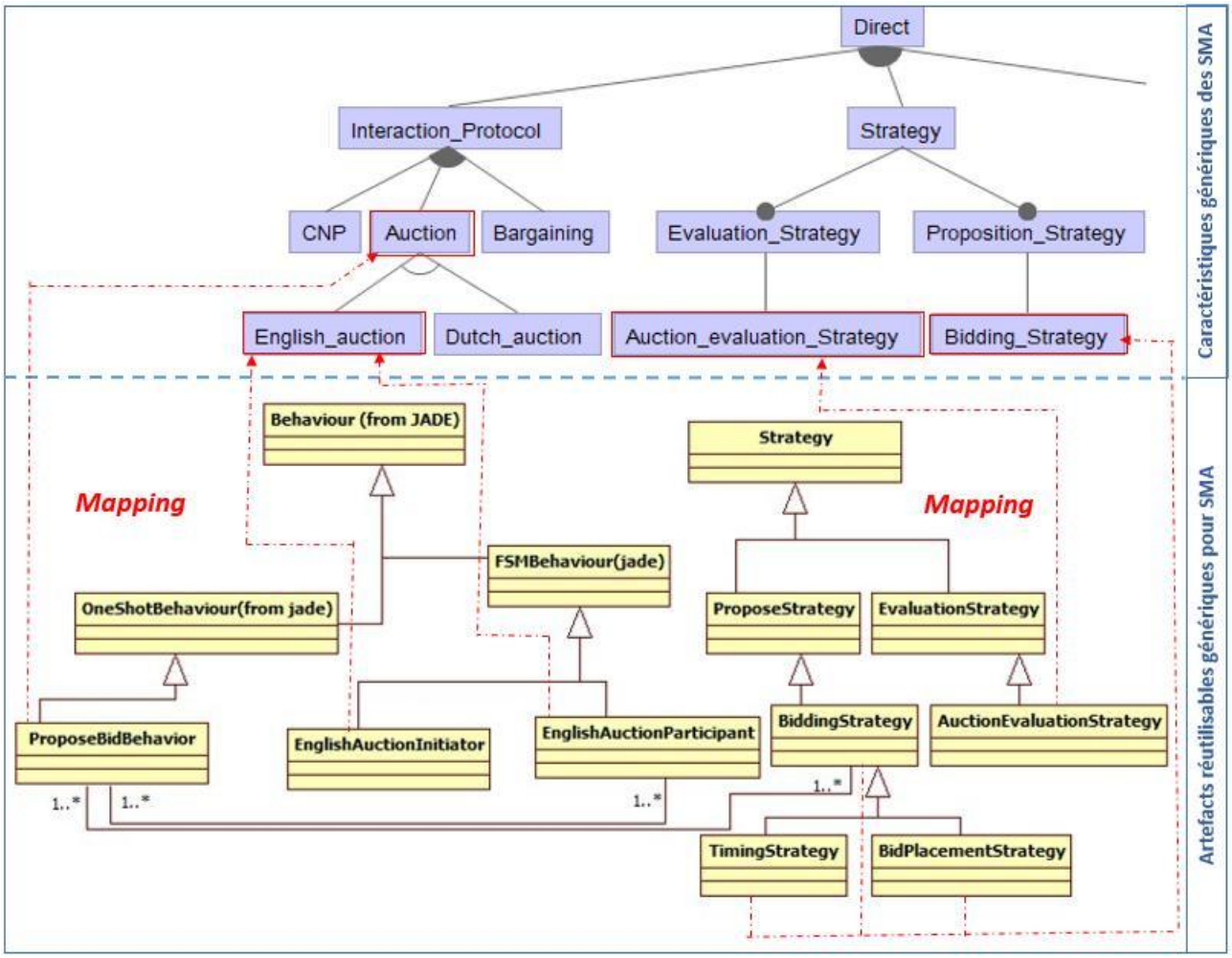


FIGURE 4.13 – un exemple de mapping entre des caractéristiques d'interactions multi-agents génériques et les artefacts génériques implémentant l'enchère Anglaise

(voir la figure 4.3), ne vont pas nécessairement figurer au niveau de l'implémentation. Par exemple, le concept de *Role*, va se traduire dans Madkit par une fonction, un service ou une identification d'un agent au sein d'un groupe particulier. Chaque agent peut jouer plusieurs rôles au sein de son organisation. De plus, ces rôles peuvent être menés au sein de plusieurs groupes. Un même rôle, peut être joué par un ou plusieurs agents.
Tandis qu'au niveau de la plate-forme JADE, les classes qui implémentent les *comportements* peuvent représenter des *rôles réutilisables*. Comme nos implémentations d'agents s'articulent principalement autour de la plate-forme JADE, nous avons donc intégré les principales implémentations relatives à l'organisation, en s'inspirant de Madkit afin de permettre aux agents de faire partie d'une organisation. Nous avons donc, raffiné l'ensemble des classes correspondantes. Par exemple, comme tout agent devrait disposer de primitives permettant d'observer son organisation, de façon à connaître les groupes

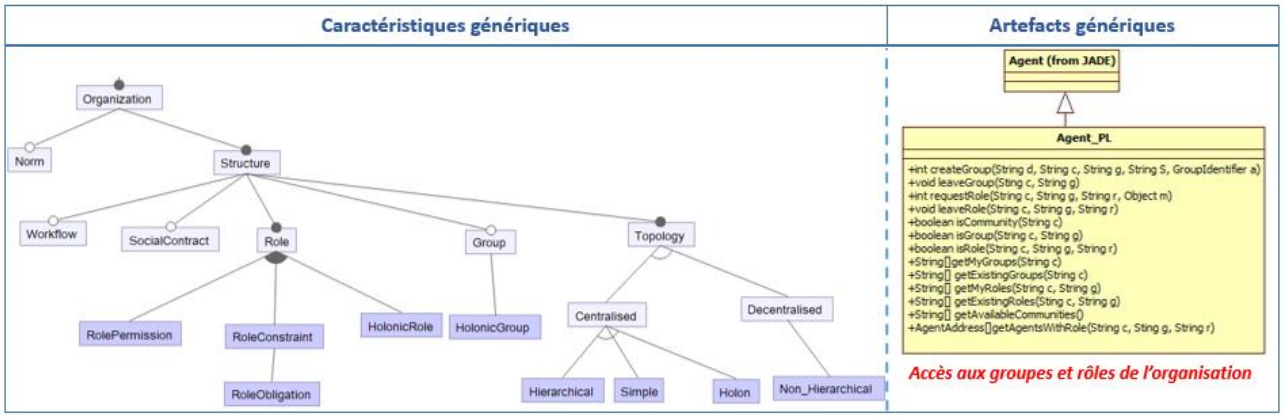


FIGURE 4.14 – un exemple d'association de caractéristiques d'organisation à un agent ayant accès aux groupes et rôles de l'organisation

et les rôles qu'il est entrain de jouer, mais aussi qu'il soit en mesure d'entrer ou de se retirer d'un groupe, l'implémentation générique de ces aspects est directement associée aux agents. La figure 4.14, illustre un exemple de réutilisation des structurations proposées par Madkit, afin de permettre aux agents s'exécutant au niveau de JADE, de tenir compte de leur rôles, ainsi que de leur appartenance aux groupes.

Certes, le paramétrage des comportements d'agents sous JADE est nécessaire afin de mener à bien les rôles. Ceci, sera réalisé durant l'activité d'implémentation spécifique au domaine d'application. Mais cela n'aura aucune influence sur l'accès à la vue organisationnelle. Dès lors que les caractéristiques d'organisation font partie d'une configuration, chaque agent du système aura accès aux organisations (groupes et rôles) ; quel que soit son modèle de comportement spécifique.

L'originalité de notre approche figure dans la possibilité de passer d'une organisation avec une configuration minimale pour un produit SMA, à une organisation à laquelle il est possible d'intégrer d'autres concepts issus d'autres modèles. Comme, la possibilité de tenir compte des normes de l'organisation, en sélectionnant les caractéristiques relatives aux contraintes *Role_ Constraint*, et permissions *Role_ Permission* liées aux rôles de l'organisation.

Ce concept de *Normes*, que nous avons extrait pendant notre analyse des concepts de l'organisation d'agents, est issu du modèle Moise+ [21], qui propose une vue dite *déontique* (normative), que nous avons présentée dans les chapitres 2 et 5. Ce dernier est associé à la plateforme Moise [136], qui a également été integrée au sein de la plate-forme jaCaMo [132].

Bien que nous n'ayons pas implémenté toute la variabilité relative aux organisations ; il est possible d'étendre notre implémentation de la même façon que nous venons de l'ex-

pliquer. Ce qui permettrait par exemple, de supporter d'autres implémentations telles que celles de Janus [92], qui est l'implémentation d'Aspecs [89], et qu'il serait intéressant d'intégrer pour supporter une ligne de produits de SMA organisés en holarchies. Ces fonctionnalités seraient directement associées au niveau de notre modèle, aux caractéristiques *Holon, HolonicGroup,* et *HolonicRole*.

4.3.3 Analyse spécifique au domaine d'application

Au cours de cette activité, nous analysons les points communs des SMA et les différences (variabilités) entre les membres d'une famille de SMA spécifique à un domaine d'application. Par exemple, la vente de livres, pour produire un *MAS spécifique FM*. Contrairement aux approches MAS-PL actuelles, ce modèle n'est pas construit de zéro, il est obtenu en raffinant le *Generic MAS FM*.

Le processus de raffinement, est réalisé en ajoutant des fonctionnalités spécifiques pour résoudre le problème. Ces caractéristiques spécifiques, sont placées hiérarchiquement sous les caractéristiques génériques qu'elles spécialisent. Plus précisément, ces caractéristiques présentent le deuxième niveau de notre modèle de caractéristiques.

Afin d'illustrer cette activité, nous allons analyser le domaine spécifique du multi-agent Contest. Nous ferons référence aux caractéristiques *MAS spécifiques* comme des caractéristiques *Contest MAS*.

Les équipes de jeu d'agents du concours (contest), se déplacent dans les rues d'une ville réaliste, dans le but d'être rémunérées en réalisant un certain nombre de jobs.

Les équipes devraient ensuite décider de la façon de parcourir la carte de la ville ; pour obtenir les ressources nécessaires à l'assemblage, à l'achat et à la livraison d'articles. Les équipes de jeu doivent tenir compte de destinations cibles comme les magasins, les entrepôts, les stations de recharge et les installations de stockage. Les points du tournoi, sont répartis selon le montant d'argent qu'une équipe possède à la fin de la simulation.

La figure 4.15 représente différents exemples de raffinement du modèle de caractéristiques génériques (premier niveau de variabilité). Une ligne sépare les deux catégories de caractéristiques appartenant à des niveaux de variabilité différents de la ligne de produits multi-agents. Par exemple, la caractéristique obligatoire (mandatory) du Contest *cityMap* (nœud fils de la caractéristique *Physical _ Environment*), raffine la caractéristique *Environnement* et correspond à l'environnement du Contest fourni. La caractéristique spécifique *BuyItem _ Goal* quant à elle, raffine la fonction *Mono _ Goal*. Cette dernière, permet de spécifier des agents de Concours spécifiques pour l'objectif d'acquisition d'articles.

La partie (a) de la figure est relative à une première version du *FM spécifique*, elle représente la plus simple des versions, et ne considère que les agents qui ont un seul but à atteindre. Mais,

comme notre approche est incrémentale, le modèle obtenu, peut également être raffiné à son tour. La figure 4.15 représente en (b) et (c) d'autres exemples de raffinement.

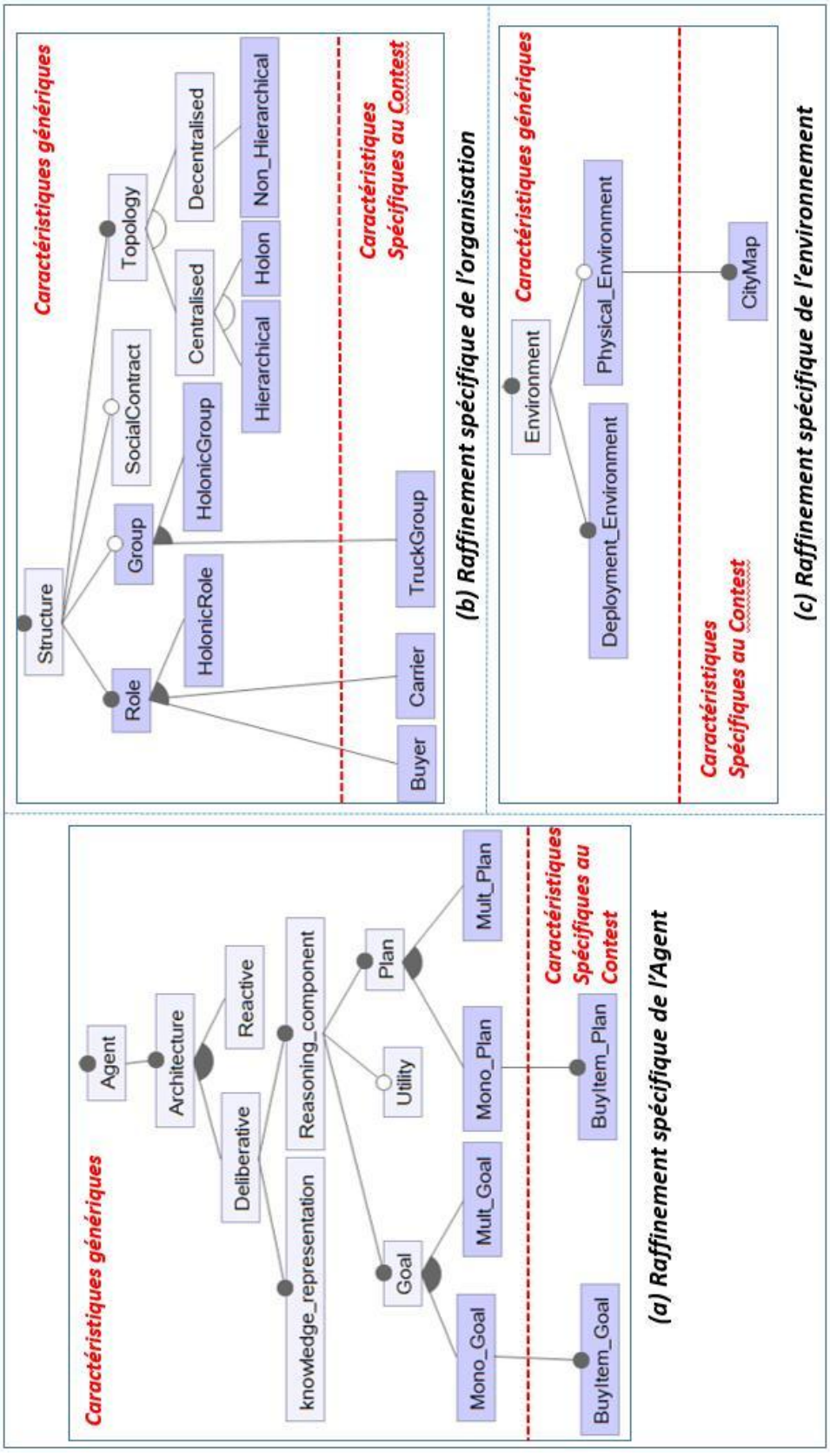


FIGURE 4.15 – Exemples de FM spécifiques à Contest obtenus par raffinement

Ce raffinement, comprend l'ajout de plus de caractéristiques pour supporter d'autres variantes multi-agents afin de répondre aux besoins (approche réactive).

Pour réaliser un tel raffinement, il suffit d'associer les besoins de la ligne de produits spécifiques, aux éléments conceptuels correspondants. Par exemple, lorsque nous parlons de la carte de la ville au niveau de laquelle les véhicules se déplacent. Celle-ci nécessite de prévoir un environnement où les agents se déplacent. Par conséquent, il s'agit d'un environnement physique. C'est la raison pour laquelle, cette caractéristique est raffinée, et non une autre. Plus de détails et d'exemples seront présentés dans la suite du présent manuscrit au niveau du chapitre 6, afin d'évaluer la faisabilité de notre approche et la possibilité de l'appliquer à différents cas d'application.

4.3.4 Implémentation spécifique au domaine d'application

Au cours de la dernière activité de l'ingénierie du domaine, les artefacts réutilisables de SMA spécifiques vont servir à implémenter les caractéristiques spécifiques des SMA, en réutilisant les artefacts génériques implémentés lors de la seconde activité. En effet, les approches MAS-PL existantes proposent d'implémenter la ligne de produits multi-agents de zéro. A l'inverse, notre approche offre au développeur, la possibilité de réutiliser les artefacts génériques que nous avons présentés plus haut.

La figure 6.23 donne un exemple d'artefacts spécifiques, qui implémente une portion d'une caractéristique spécifique au Contest. Il est important de souligner que, ce ne sont pas toutes les caractéristiques spécifiques à un domaine d'application donné, qui sont implémentées par la réutilisation d'artefacts génériques. Seules les implémentaions nécessaires au développement des caractéristiques relatives aux besoins des utilisateurs, ne seront pas implémentés de zéro.

Par exemple, la figure 4.16 représente un récapitulatif des différentes activités présentées dans cette section. Cette figure illustre un artefact réutilisable qui se trouve sur le côté droit de la figure. Cet artefact implémente la fonctionnalité de la caractéristique *BuyItem _ Goal*, en réutilisant la fonction PerformActionGoal() qui est associée (mapping) à la fonctionnalité *Mono _ Goal*.

Cet exemple, montre bien que pour satisfaire le besoin fonctionnel d'achat d'items par les véhicules, il faut une implémentation de but (Goal), qui doit être exécuté par un agent délibératif. Ceci pouvant être implémenté grâce au premier algorithme de wooldridge [58], ce dernier peut être réutilisé, et servir comme base d'implémentation du comportement des agents véhicules devant exécuter le job en question. Nous allons présenter d'autres exemples d'implémentation spécifiques avec détail au niveau du chapitre 6.

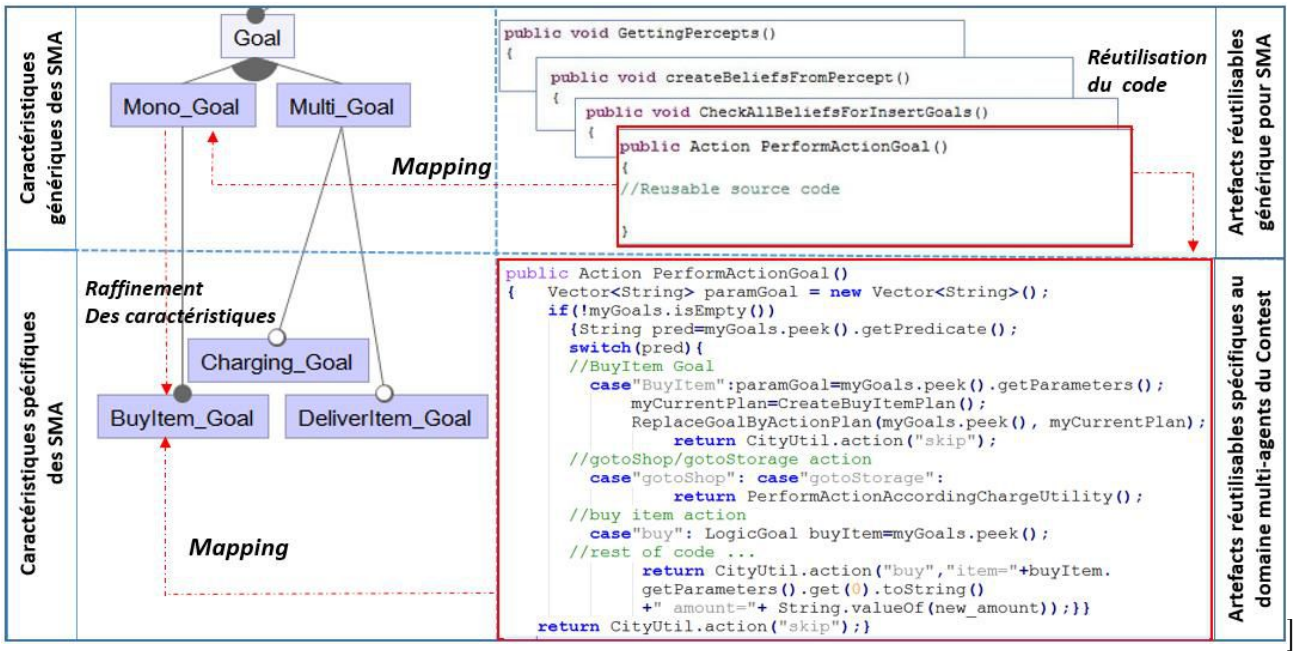


FIGURE 4.16 – Récapitulatif des activités de la phase d'ingénierie du domaine

4.4 Ingénierie d'application

Cette section regroupe les différentes activités de la phase d'ingénierie d'application. Chacune des activités est illustrée par de petits exemples.

Il est important de souligner que contrairement à la première phase d'ingénierie du domaine, nous n'avons pas ajouté de nouvelles activités au niveau de la phase d'ingénierie d'application.

Cette dernière phase, permet de sélectionner une configuration et de dériver des variantes multi-agents personnalisées.

L'originalité de cette phase, réside dans la possibilité de sélectionner des caractéristiques multi-agents provenant des deux niveaux distincts de variabilité (indépendante et spécifique au domaine d'application). Dès lors que notre approche s'appuie sur un modèle de caractéristiques multi-niveaux ; il est en effet possible d'obtenir des instances intermédiaires de la ligne de produits multi-agents. Nous présentons dans cette section des exemples de configurations possibles issues des deux niveaux du modèle de caractéristiques.

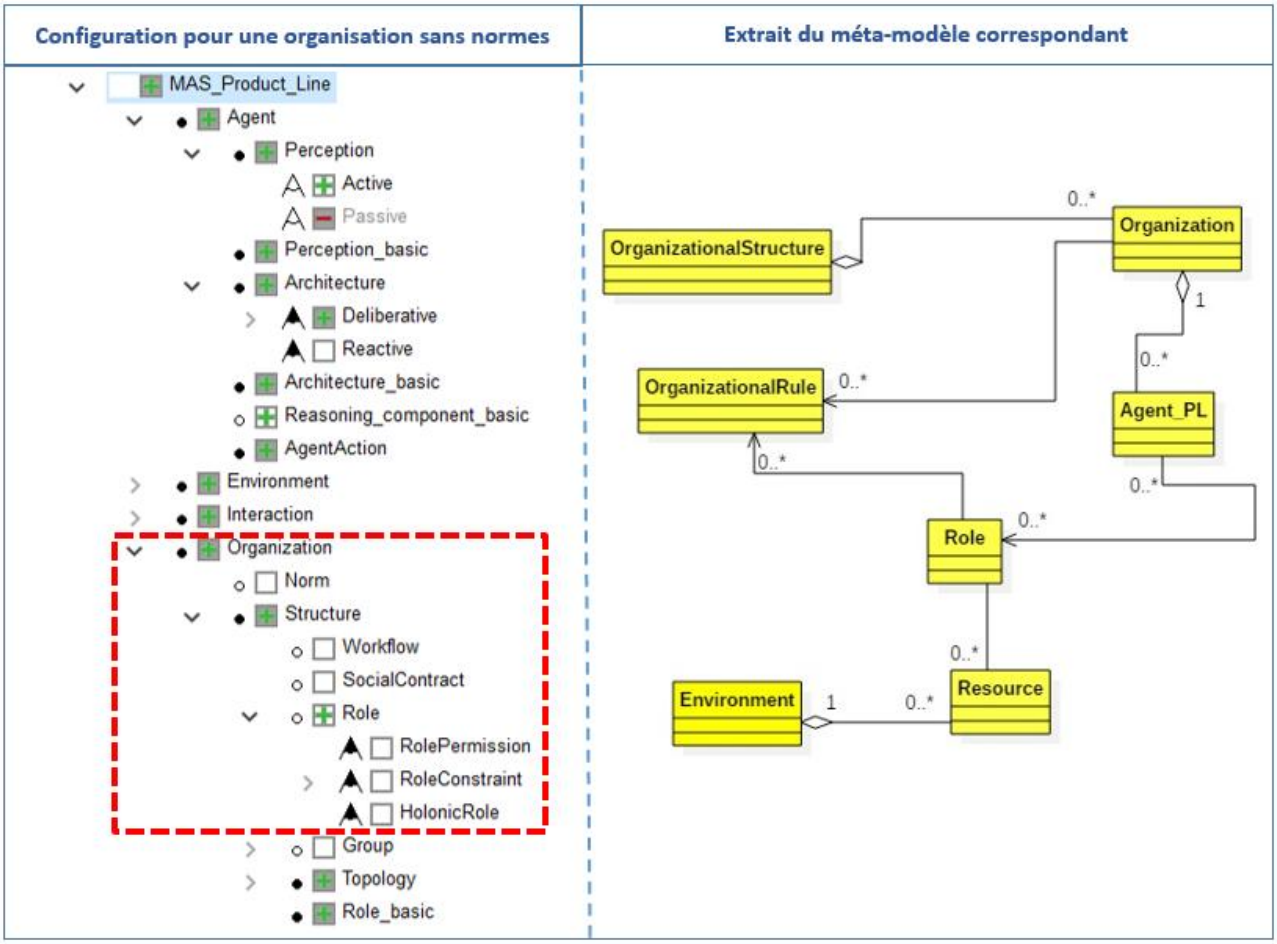


FIGURE 4.17 – Un exemple de configuration relative au besoin SMA1, et accompagnée d'un extrait du méta-modèle correspondant

4.4.1 Choix d'une configuration

Une configuration représente l'ensemble de caractéristiques qui doivent être sélectionnées, afin de satisfaire les fonctionnalités souhaitées dans le système multi-agent à dériver. Ce choix, devra être fait sur la base d'une sélection d'un ensemble valide de caractéristiques. Par exemple, il n'est pas possible de sélectionner la caractéristique *Evaluation_Strategy*, si le protocole d'interaction au sein duquel la stratégie sera implémentée, n'est pas sélectionné.

Ceci est bien entendu réalisable, grâce à l'écriture de contraintes permettant d'éliminer certaines configurations invalides. L'ensemble des contraintes indépendantes du domaine d'application ont été intégrées au modèle, cependant le concepteur de la ligne de produits multi-agents, devra ajouter l'ensemble des contraintes spécifiques au domaine d'application.

Avant de pouvoir sélectionner les caractéristiques devant faire partie d'une configuration,

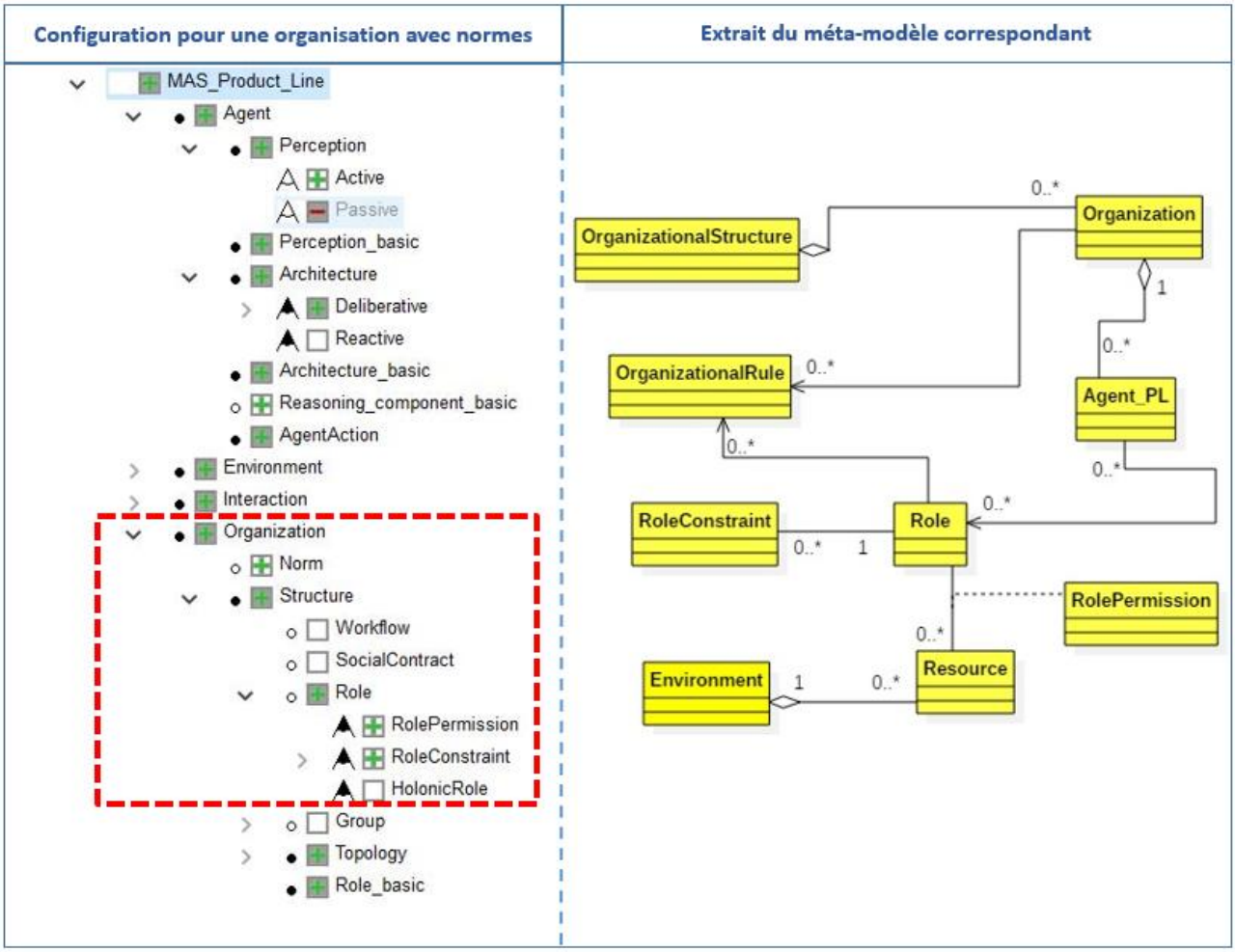


FIGURE 4.18 – Un exemple de configuration relative au besoin SMA2, et accompagnée d'un extrait du méta-modèle correspondant

il est important de représenter l'ensemble des besoins pour chacune des variantes. Il est également important de souligner, qu'il est possible de procéder à deux types de dérivations. Un premier type basé sur une configuration tenant compte uniquement des caractéristiques du premier niveau du modèle de caractéristique (FM générique) , et un deuxième type basé sur une configuration tenant compte des caractéristiques provenant des deux niveaux (FM spécifique).

4.4.1.1 Configuration à partir du modèle de caractéristiques indépendant du domaine d'application

Comme expliqué plus haut, le premier niveau du modèle de caractéristiques (générique), décrit la variabilité du domaine multi-agents, qui est indépendant d'un domaine d'application spécifique. Il est possible de procéder à une sélection des caractéristiques (configuration) du pre-

Variantes SMA	Besoins
SMA1	Un SMA constitué d'agents dotés de perception active, et d'une architecture délibérative. Une organisation ne tenant pas compte des normes. Un environnement d'agents où l'accès aux ressources n'est pas restreint.
SMA2	Un SMA constitué d'agents dotés de perception active, et d'une architecture délibérative. Une organisation tenant compte des normes, tenant ainsi compte des permissions liées au rôles ; ainsi qu'aux contraintes relatives aux rôles en question. Un environnement d'agents où l'accès aux ressources est restreint.
SMA3	Un SMA constitué d'agents réactifs et délibératifs. Les interactions entre agents se font de façon directe. L'organisation du système se fait selon une topologie hiérarchique.
SMA4	Un SMA constitué d'agents délibératifs basés sur le second algorithme de wooldridge. L'environnement d'agents est constitué de zones au niveau des quelles les agents se positionnent (physical environment). Les interactions peuvent se faire de manière directe et/ou indirectes. L'organisation du système est plate.
SMA5	Un SMA constitué d'agents délibératifs, dotés d'une perception active et capables d'interagir en utilisant différents protocoles d'interaction. L'organisation du système nécessite la définition de normes.

TABLE 4.4 – Exemples de besoins (requirements) multi-agents

mier niveau du modèle de caractéristiques, et ce avant de procéder au raffinement de ce dernier de façon spécifique. Le produit dérivé, n'étant pas associé à un domaine d'application spécifique, représentera une variante possible du méta-modèle regroupant l'ensemble des concepts qui sont associés aux caractéristiques choisies.

Le tableau 4.4, donne des exemples de besoins auxquels les caractéristiques du premier niveau répondent, et qui par conséquent feront également partie de la configuration réalisée à partir du modèle de caractéristiques spécifique au domaine d'application.

L'ensemble de ces besoins donnent une indication sur l'ensemble des caractéristiques qui doivent faire partie de la configuration, et ce afin de satisfaire ce dernier.

Le premier besoin par exemple (SMA1), permet de déterminer les caractéristiques qui doivent être sélectionnées pour chacune des dimensions de l'approche voyelles [28, 29]. La partie gauche de la figure 4.17 représente la configuration qui permet de satisfaire ce besoin, car ce dernier stipule entre autres que l'organisation du SMA n'est pas normée, et par conséquent, que la caractéristique *Norm*, ne doit pas être sélectionnée. La partie droite de la figure, correspond à l'extrait du méta-modèle correspondant à ce besoin. Ce dernier, permet de supporter l'ensemble des concepts relatifs à une organisation multi-agents sans normes.

Le deuxième exemple présenté dans la figure 4.18 représente la configuration relative au deuxième besoin (SMA2). A l'inverse du premier besoin, L'organisation du système doit intégrer des normes. Par conséquent, la caractéristique *Norm*, est sélectionnée. L'ensemble des permissions liées aux rôles, afin de restreindre entre autre l'accès aux ressources de l'environ-

nement, font également partie de la configuration du système. Par conséquent, la sélection de la caractéristique *Role_ Permission*, est également nécessaire afin de satisfaire le besoin en question.

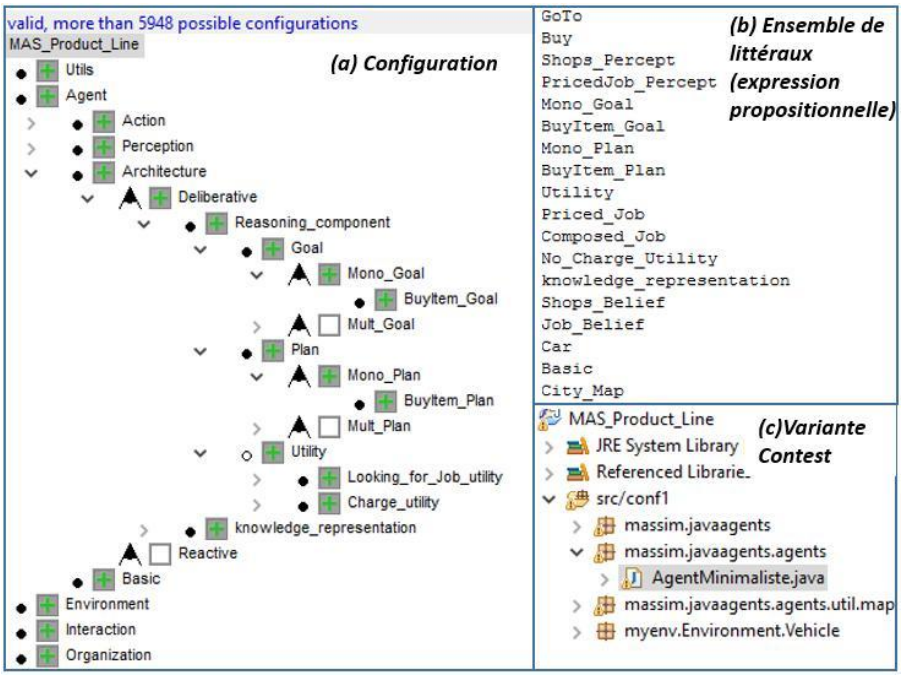


FIGURE 4.19 – Un exemple en (a) d'une configuration d'Agent de Contest et en (b) son expression propositionnelle et en (c) la dérivation de la variante

4.4.1.2 Configuration à partir du modèle de caractéristiques spécifique au domaine d'application

Le tableau 4.5 donne quelques exemples de besoins spécifiques à un domaine d'application spécifique, qui est celui du multi-agent Contest.

Le premier besoin correspond à la variante CAV1 (Contest Agent Variant) et est satisfait par la configuration représentée en (a) dans la figure 4.19. Toutes les caractéristiques à sélectionner afin de satisfaire ce besoin sont représentées en (b) par le biais de l'ensemble des littéraux. Par conséquent, toutes les autres caractéristiques n'étant pas sélectionnées sont exclues de la configuration.

Tout comme pour la sélection des caractéristiques du premier niveau du modèle (générique), le besoin d'utilisateurs, donne une indication sur l'ensemble des caractéristiques qui doivent faire partie de la configuration, et qui sont issues des deux niveaux du modèle.

Par exemple, dès lors que le besoin CAV1 indique que variante souhaitée ne comporte que

Variantes Contest	Besoins
CAV1	Un variante d'agent de contest de type véhicule (Car Contest Agent Variant) qui a un but à réaliser en exécutant un job d'acquisition (achat d'items dans un shop). L'agent n'a pas de fonction d'utilité de recharge de sa batterie (utility). Il possède un seul plan à exécuter (trouver un shop, s'y rendre, et acheter les items)
CTV1	Une variante d'équipe de Contest ou Contest Team Variant (CTV) qui exécute seulement des priced jobs (job à prix fixe). L'équipe est organisée selon une topologie hiérarchique sans inclure des structures de groupes.
CTV2	Une variante d'équipe de Contest qui exécute seulement des priced jobs. L'équipe est organisée selon une topologie non hiérarchique, avec la possibilité de structurer cette organisation en groupe de voitures et de camions. Ces groupes sont supervisés par des agents supervisors.
CTV3	Une variante d'équipe qui exécute des priced jobs ainsi que des auctioned jobs (jobs à enchères) qui représentent des jobs à prix négociable. L'équipe ne supporte pas de structuration en groupes.
CTV4	Une variante d'équipe qui exécute des priced jobs ainsi que des auctioned jobs (jobs à enchères) qui représentent des jobs à prix négociable. L'équipe est organisée en topologie hiérarchique structurée en groupes de voitures et de camions.

TABLE 4.5 – Exemples de besoins (requirements) multi-agents du Contest

des agents ayant pour but d'exécuter un job d'acquisition ; il est nécessaire que ces agents en question soient capables de percevoir l'ensemble des magasins disponibles. Ceci signifie, que la caractéristique *Shops_ Percept* doit faire partie de la configuration, sinon l'agent ne pourra exécuter son but. Ce dernier, afin d'être réalisé, nécessite par conséquent la sélection de la caractéristique *BuyItem_ Goal*, afin de garantir que le besoin soit satisfait. De plus, dès lors que le besoin indique que l'agent n'a pas de fonction d'utilité de recharge de sa batterie(utility), la caractéristique *No_ Charge_ Utility* doit également faire partie de la configuration du système. De plus, afin que l'agent soit capable d'exécuter le plan associé au job d'acquisition, il est primordial que la caractéristique *BuyItem_ Plan* soit également sélectionnée.

Les autres besoins représentés dans le tableau correspondent à des variantes d'équipes de Contest ou CTV (Contest Team Variant), et considèrent certains aspects organisationnels. Par exemple, contrairement à CTV3, afin de satisfaire les besoins de CTV1, il est obligatoire d'inclure des caractéristiques telle que *Non_ Hierarchical*, et d'exclure par conséquent les autres telle que la caractéristique *Hierarchical*.

4.4.2 Dérivation d'une variante multi-agents

La dérivation est réalisée sur la base d'une configuration valide. Le code dérivé satisfaisant le besoin exprimé par la configuration représente une variante multi-agents pouvant être déployée. Cette activité est automatique et est réalisée par le biais d'un composer tel que FeatureHouse.

La figure 4.19 représente en (c) un exemple de code dérivé relativement à la configuration représentée en (a).

4.5 Conclusion

Dans ce chapitre nous avons présenté le cœur de notre travail, qui s'articule autour de notre approche. Cette dernière est basée sur le framework SPLE que nous réutilisons. Notre approche propose de découper la phase d'ingénierie du domaine en deux, tout en introduisant de nouvelles activités.

Ces activités visent à faciliter le développement de lignes de produits mutli-agents, en repoussant les limites des approches MAS-PL existantes. Ces dernières, ne proposent pas de points de départ pour les activités de spécification et d'implémentation de la ligne de produits multi-agents, car ces derniers sont spécifiques au domaine d'application. C'est pour cela, que nous avons proposé de spécifier et d'implémenter la communalité ainsi que la variabilité des SMA indépendamment d'un domaine d'application spécifique.

En effet, nous avons proposé une ligne de produits multiples proposant deux niveaux de variabilité.

La premier niveau du modèle, représente le modèle de caractéristiques (feature model) génériques, ainsi que les artefacts qui l'implémentent. Nous avons montré que notre approche MAS-PL présente l'approche Voyelles [28, 29] sous forme de ligne de produits multi-agents, et par conséquent, elle se positionne parmi les extensions d'approches. Notre extension, est la première à se manifester au niveau du modèle de caractéristiques qui organise l'ensemble des fonctionalités d'un SMA en termes de points communs et variables selon les dimensions de l'approche voyelles. Nous avons montré, que ce modèle sert de point de départ aux concepteurs, et aux développeurs des lignes de produits multi-agents.

Nous avons également, présenté l'ensemble des artefacts réutilisables que nous avons extrait des implémentations réutilisables telles que celles provenant de JADE. Ensuite, nous avons montré l'association entre ces artefacts réutilisables et les caractéristiques génériques qu'elles implémentent.

Le second niveau du modèle, est associé à une variabilité spécifique d'un domaine d'application. Nous avons présenté à travers des exemples spécifiques, comment réutiliser le modèle de caractéristiques génériques, afin d'obtenir un modèle spécifique (raffinement du modèle). Nous avons fait de même pour les artefacts réutilisables afin d'aboutir à des implémentations spécifiques.

Pour finir nous avons donné des exemples de besoins relatifs à la ligne de systèmes multi-agents à satisfaire par le biais de configurations, sur les quelles se base la dérivation de variantes exécutables au sein de leur environnement.

Dans la suite du présent manuscrit, nous donnerons plus de détails sur l'application de notre approche à travers trois différents cas d'étude. Mais tout d'abord, nous allons présenter dans le prochain chapitre, en détail les étapes qui nous ont permis de construire le modèle de caractéristiques générique. Ce dernier, ayant été bâti sur les similitudes et la variabilité des connaissances des SMA, et plus précisément sur des concepts provenant des modèles et méthodologies du domaine.

Chapitre 5

Caractéristiques génériques des SMA

5.1 Introduction

L'introduction des approches d'ingénierie de lignes de produits multi-agents, a permis de faciliter le développement des familles d'applications multi-agents. Le principe de base utilisé au sein de ces approches, reste identique, car il permet une dérivation automatique de produits multi-agents. Cependant, la démarche adoptée par ces approches demeure différente. En effet, nous avons vu que certaines approches proposaient des compositions, d'autres des extensions ou encore des approches adhoc (construites de zéro). Notre approche présentée dans le chapitre précédent, s'inscrit dans le cadre d'extensions du modèle de base de l'approche VOYELLES, par des principes de communalité et de variabilité.

Le présent chapitre, présente en détail les étapes qui nous ont permis de construire le modèle de caractéristiques indépendant du domaine d'application, et issu de la première activité de notre approche.

Pour chacune des voyelles A.E.I.O, qui représentent respectivement les dimensions d'agent, d'environnement, d'interaction et d'organisation, nous allons donner les modèles de caractéristiques intermédiaires obtenus, avant leur consolidation au sein du modèle final que nous avons présenté dans le chapitre précédent (voir la figure 4.3).

Nous allons présenter la délimitation du domaine sur laquelle s'est basée notre analyse des concepts. Cette délimitation a été faite en sélectionnant un sous-ensemble de méta-modèles permettant de couvrir les principaux concepts de chaque dimension du système. Bien que notre délimitation, n'inclut pas tous les méta-modèles du domaine, elle sert de point de départ pour aller vers un modèle générique. De plus, il est possible d'inclure d'avantages de méta-modèles, et par conséquent d'avantages de concepts afin d'étendre le modèle actuel.

Nous allons également décrire chacune des caractéristiques obtenues, suite à notre analyse des points communs et variables, basée d'une part sur l'analyse sémantique des concepts, et

d'autres part sur les éléments présentés dans le chapitre 2.

Nous finirons par les contraintes que nous avons introduites au niveau de notre modèle, afin d'éliminer des configurations non valides. En effet, comme les éléments que nous avons groupés proviennent de modèles différents, il est important d'avoir de telles contraintes car les éléments de notre modèle ne sont pas tous composables.

5.2 Vers un modèle générique de caractéristiques d'agents (voyelle A)

5.2.1 Vue globale du modèle

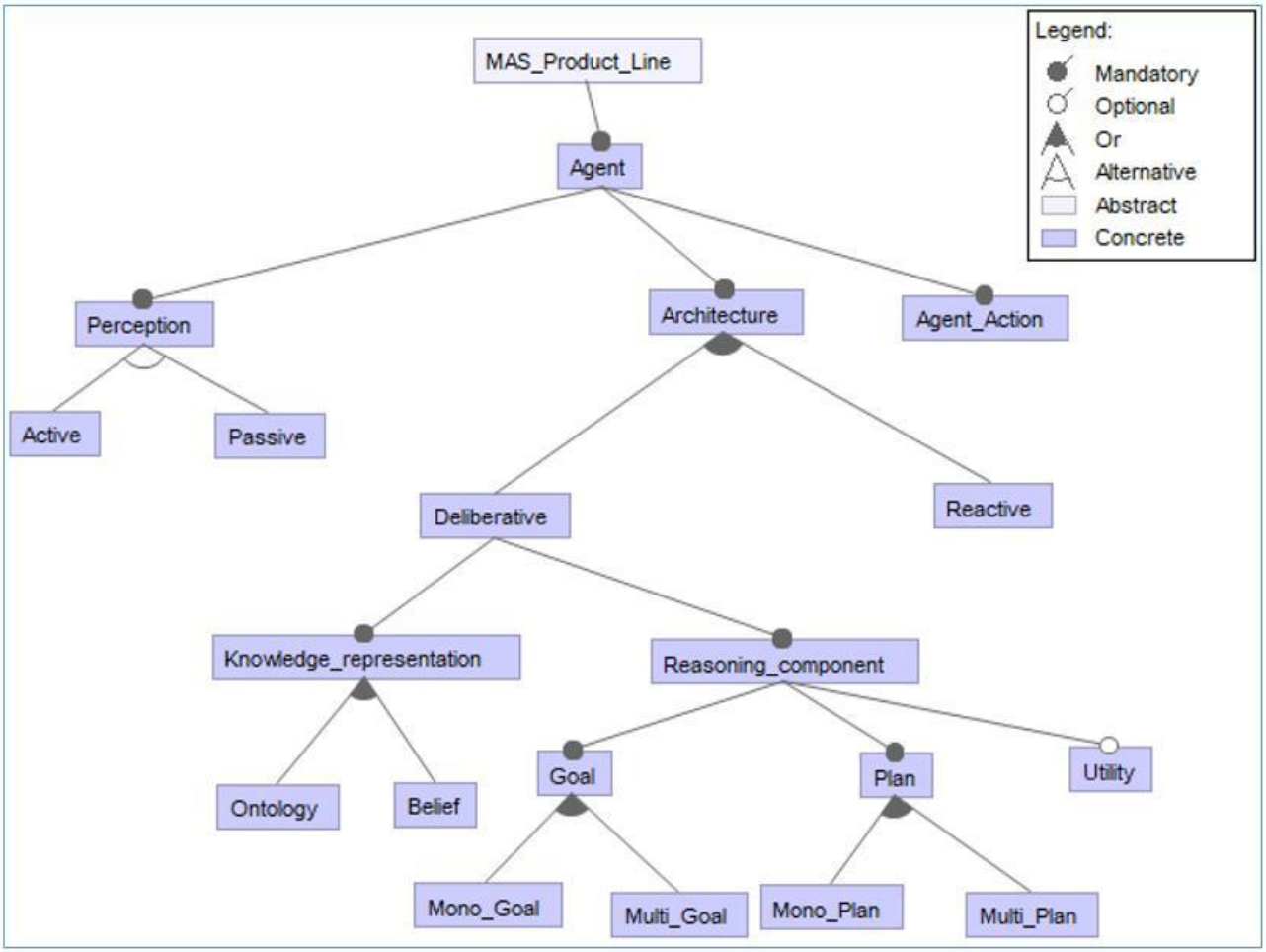


FIGURE 5.1 – Le modèle de caractéristiques d'agents (Agent_FM)

Le premier modèle construit, correspond à la voyelle A de l'approche VOYELLES [28, 29],

soit aux caractéristiques communes et variables de l'agent. Le modèle obtenu (Agent_FM) est représenté dans la figure 5.1.

Dans ce qui suit, nous expliquons comment sommes-nous arriver à un tel résultat.

5.2.2 Les concepts d'agents couverts par la délimitation du domaine

La délimitation du domaine, a été faite de façon à tenir compte des principaux concepts relatifs aux agents.

Par conséquent, nous avons choisi d'inclure des méta-modèles qui proposent de définir des concepts directement liés aux architectures internes de l'agent, à ses aptitudes (capabilities), à ses plans, ou encore à ses croyances (belief) entre autres.

Le résultat relatif à la délimitation de l'étendue de l'analyse du domaine (domain scoping) des concepts d'agents couverts par des méta-modèles multi-agents, est représenté dans le tableau 5.2. Les détails relatifs à l'analyse de ces concepts, ainsi que leurs sémantiques sont détaillés dans ce qui suit.

Concepts	**Aspecs**	**Ingenias**	**OperA**	**MaSE**	**Moise+**	**Gaia**	**Prometheus**
Goal	x	x		x	x		x
Plan	x				x		x
Belief		x					
Action	x				x	x	x
Ontology	x		x	x			
Task		x			x		
Mission					x		
Permission						x	
Percept							x
Capability				x			x
Capacity	x						
Skill	x						
Agent	x	x	x	x	x	x	x

TABLE 5.2 – Délimitation de l'étendue du domaine d'analyse des principaux concepts d'agents à partir des méta-modèles

5.2.3 Analyse sémantique des concepts d'agents pour le choix des caractéristiques

Notre analyse des caractéristiques d'agents, a été réalisée d'une part sur la base de l'ensemble des éléments présentés au niveau de la section 2.3 du chapitre 2, et d'autres part sur l'ensemble des concepts représentés dans le tableau 5.2 ci-dessus.

A partir de ces éléments, il ressort que les principales caractéristiques communes à tous les agents sont leur *perception*, *actions*, et *architecture*.

5.2.3.1 Analyse de l'architecture d'agent

1. **Les caractéristiques *Architecture*, *Deliberative*, et *Reactive* :**

   Les architectures d'agents fournissent des solutions aux agents, et définissent leurs fonctionnalités afin de leur permettre d'agir et d'interagir dans un environnement dynamique. Certaines architectures permettent à l'agent de suivre un cycle de comportement de type *Perception-Stimulus- "Re"action* . Nous parlons alors d'agents réactifs.

   Nous avons constaté, que ce comportement présente une variabilité qui est directement associée au module de délibération de l'agent, et qui lui permet de suivre un tout autre cycle de comportement de type *Perception - Délibération - Action* (voir 2.3.1). C'est pourquoi, nous avons décidé, d'introduire au niveau de notre modèle les caractéristiques d'architecture *délibérative* et *réactive*. Tel que la sélection de ces deux caractéristiques au sein d'une configuration, permettrait de supporter une architecture dite hybride.

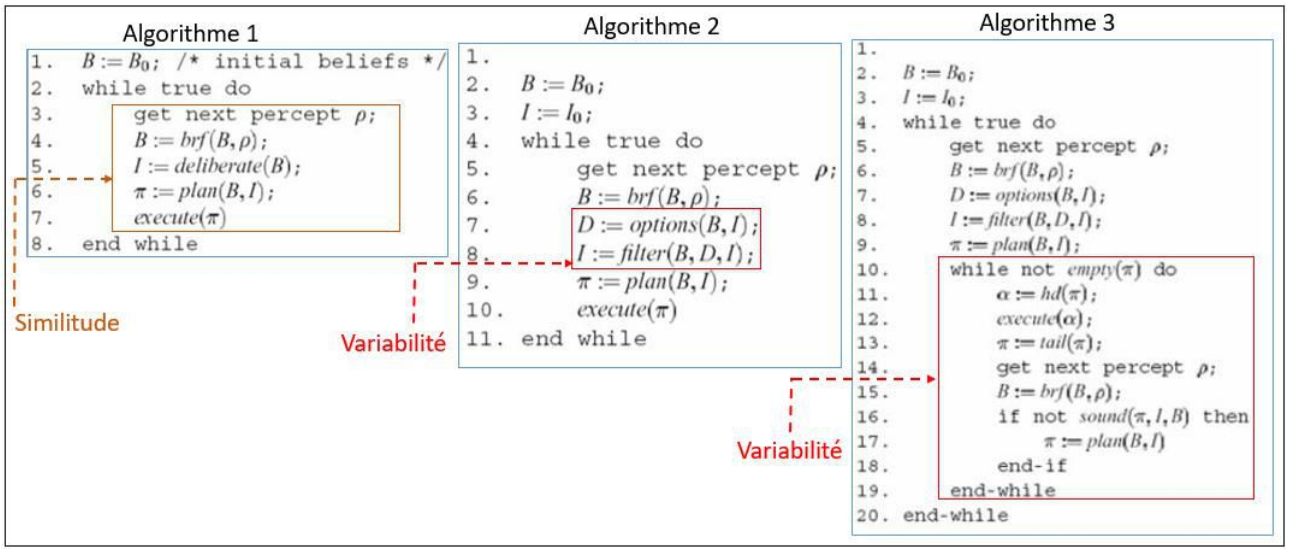


FIGURE 5.2 – Un exemple de variabilité algorithmique du modèle d'architecture BDI

Certaines méthodologies faisant partie de notre délimitation du domaine (voir le tableau 5.2), prennent justement en ligne de compte ces aspects de capacités mentales de l'agent comme la méthodologie Ingenias [57], qui reprend entre autre le modèle d'architecture délibérative la plus utilisée qui est celle du BDI (Belief Desire Intention) (voir 2.3.1.1). Ce modèle d'architecture a été adoptée par d'autres méthodologies, telles que Prometheus [60], qui a ensuite évolué en permettant des descriptions conceptuelles indépendantes de l'architecture interne de l'agent.

2. **Les Caractéristiques *Knowledge_ representation*, *Belief*, et *Ontology* :**
   En analysant de près l'architecture interne de l'agent, nous avons déduit que les agents délibératifs présentent comme point commun le fait d'avoir une représentation de leurs connaissances.

   Bien qu'aucun concept à part celui d'*agent*, ne soit commun à l'ensemble des méta-modèles, chacun d'entre eux met en avant les principaux éléments ou modules composant les agents. Par exemple le méta-modèle d'*Aspecs* [89], montre que l'agent est composé de buts (concept *Goal*) individuels qu'il devra atteindre par l'exécution de plans (concept *plan*) correspondants en fonction du rôle que l'agent jouera au sein de son organisation. Ce méta-modèle propose une représentation des connaissances de l'agent sous forme d'ontologie (concept *Ontology*, tandis qu'au sein du méta-modèle d'*Ingenias* [57] par exemple seules les croyances de l'agent (concept *Belief*) sont utilisés.

   Ces pour ces raisons, que nous avons décidé de spécifier cette variabilité au sein du modèle, en introduisant la caractéristique relative à la représentation des connaissances de l'agent (*knowledge_ representation*), et qui permet au concepteur de choisir le type de représentation qu'il désire exploiter pour les agents de la ligne de produits.

   Dans l'architecture BDI par exemple, les croyances de l'agent représentent l'ensemble de ses connaissances du domaine.

   Les caractéristiques *Belief* et *Ontology*, sont spécifiées par un choix de type "Ou" à faire. Comme mentionné ci-dessu, Ingenias [57] supporte également le concept *belief*, et le considère comme étant une spécialisation du concept *MentalState*. En effet, les croyances de l'agent de l'agent vont représenter et décrire les informations sur l'environnement, l'état interne que peut garder l'agent, et les actions qu'il peut réaliser.

   Le concept *Ontology* quant à lui, est supporté au niveau d'operA [77] entre autres. L'ontologie est nécessaire lors d'échanges de messages entre agents, et par conséquent est directement liée à la structure communicative chargée de spécifier l'ontologie.

   Aspecs [145] comme nous avons pu le voir, supporte également un tel concept au niveau du méta-modèle, et le décrit comme étant une spécification explicite d'une conceptualisation de la connaissance liée à un domaine particulier. L'ontologie est ainsi composée

des éléments concept, prédicat et action. Nous n'avons associé aucune variabilité relative à la caractéristique d'ontologie car cette dernière est décrite de façon unifiée, et elle sera définie de façon spécifique au domaine d'application.

3. **Les caractéristiques *Reasoning_ component*,*Goal*, et *Plan* :**
   Durant notre analyse, nous avons constaté qu'il était primordial d'intégrer le composant qui permet de distinguer les agents capables de raisonner de ceux qui sont réactifs. C'est pourquoi nous avons décidé d'intégrer la caractéristique *Reasoning_ component*, qui est un module obligatoire (mandatory) pour les agents délibératifs.
   Nous avons constaté également, que l'exécution de base des principes du modèle délibératif BDI pouvait changer, et par conséquent présenter une variabilité. Par exemple, dans le cas où l'agent n'a qu'un seul but (Goal) à atteindre, il n'aura pas à exécuter les fonctions *OGF* et *filter*. Cette exécution est implémentée par le premier algorithme de Wooldridge [58] représenté dans la Figure 5.2. C'est pourquoi, nous avons intégrer les caractéristiques relatives à la variabilité liée aux buts de l'agent *Goal*.
   Le concept de *Goal*, reste un élément primordial et déterminant pour le comportement des agents. Ce dernier est présent dans la plupart des méta-modèles méta-modèles faisant partie de notre délimitation du domaine. Selon Ingenias [57], le concept goal est en effet une entité mentale principale, qui peut être affectée de différentes façons. Cependant, ce concept n'est pas forcément lié uniquement aux agents de façon individuelles. En effet, dans Aspecs [89] par exemple, ce concept est relié directement aux rôles de l'organisation. Les rôles peuvent ainsi participer à l'accomplissement des buts de l'organisation par le biais de leur capacités.
   Comme il existe d'autres algorithmes relatifs au modèle BDI, il est ainsi possible d'analyser les points communs et la variabilité afin d'implémenter une architecture BDI. La Figure 5.2 représente un exemple des points communs et variables entre l'ensemble des algorithmes proposés par Wooldridge. Nous avons présenté dans le chapitre précédent, les détails relatifs aux liens entre ces caractéristiques réutilisables que nous proposons, et leurs implémentation.
   L'architecture BDI que nous venons de voir ci-dessus représente l'architecture classique connue. Cependant, il peut exister des variations de cette dernière. Dans ce contexte, il est possible par exemple d'étendre cette architecture BDI classique par un moteur CBR (case-based reasoning) [146] [147] [148], ce dernier permet de faciliter l'apprentissage des agents en leur fournissant un plus grand degrés d'autonomie que les architectures BDI pures. Ainsi, il est possible d'obtenir des agents délibératifs qui peuvent établir des plans adaptés par exemple. Cette extension d'architecture peut entrer dans le cadre d'architecture hybride du moment qu'elle combine deux types d'architectures possibles. Il est également possible de considérer des variantes pour les agents CBR. Dans ce contexte

un agent délibératif CBP-BDI est une variation possible de l'agent CBR-BDI. Ce type d'agents est spécialisé dans la génération des plans tout en adoptant un mécanisme de raisonnement basé sur la planification. En d'autres termes, les agents CBR vont avoir comme objectif de résoudre de nouveaux problèmes en adoptant des solutions similaires rencontrés par le passé [149] [148], tandis que les agents CBP se basent sur les plans générés pour chacun des cas possibles. En d'autres termes, cette architecture s'appuie sur un modèle BDI, au niveau duquel un moteur CBP a été intégré tel que les plans utilisés ont déjà été expérimentés. Bien que ces dernières caractéristiques ne soient pas supportées par notre modèle, elles peuvent néanmoins être ajoutées afin d'étendre la version actuelle du modèle.

5.2.3.2 Analyse de la perception et actions de l'agent

1. **Les caractéristiques *Perception*, *Passive*, et *Active* :**

   Comme nous l'avons vu au niveau de 2.3.2, la perception représente l'un des modules importants de l'agent. Dans Prometheus [60] par exemple, le système peut être spécifié par des scénarios, qui sont similaires à des cas d'utilisation. Ces scénarios sont décrits par le biais d'exécutions particulières de séquences qui incluent la perception, les actions ou les buts (goals).

   Suite à notre analyse de la *perception* de l'agent (voir 2.3.2), nous avons décidé d'introduire au sein de notre modèle les caractéristiques de perception *active* et *passive*.

   Pour la perception passive, il existe différents modèles proposés avec des points communs et variables [11] [63] [61]. Nous proposons de réutiliser un modèle générique de ce type de perception qui est représenté dans la figure 2.4 (voir chapitre 2).

   Comme nous avons pu le remarquer, la fonction de filtrage ou *filtering* dépend des critères de sélection spécifiques au contexte. De ce fait l'ensemble de ces critères est exclu des caractéristiques et artefacts génériques d'agents.

   Nous avons réutilisé les principaux éléments de perception passive tel représenté dans la figure 2.5. En comparaison avec la perception active, la perception passive ne permet pas à l'agent d'appliquer des filtrer. Ce qui représente donc un point de variabilité spécifié par un choix alternatif au sein de notre modèle. Cependant les fonctions de capteurs et d'interprétations représentent des éléments communs entre les deux types de perception.

2. **La caractéristique *Agent_ action* :**

   Concernant le concept *action* de l'agent, il est associé à la caractéristique *Agent_ Action* de notre modèle. Celle-ci étant spécifiée comme étant une caractéristique obligatoire (mandatory) car elle représente un composant élémentaire pour l'agent, et qui permet entre autres de le différencier d'un objet. En effet, l'agent a un comportement proactif,

et présente une autonomie de décision par rapport à ce qu'il doit faire en fonction d'une situation spécifique. En d'autres termes, il est capable de sélectionner une *action* appropriée.

Les agents exécutent leurs actions au niveau d'un environnement particulier, que ce dernier soit virtuel ou physique. Par conséquent, ces actions peuvent apporter des modifications au sein de leur environnement. La variabilité liée aux actions est donc liée entre autre à la variabilité de l'environnement, ainsi que des interactions. C'est pourquoi, nous avons ajouté la caractéristique *action* au niveau du modèle de caractéristique d'interactions, que nous présenterons plus tard dans ce chapitre. Ce dernier fait référence à l'action comme pouvant être l'objet même de l'interaction.

Cependant, l'action n'est pas toujours reliée à une interaction particulière. Dans Gaia [67, 115] par exemple, l'activité de l'agent, correspond à une unité *d'action* que l'agent peut exécuter en dehors d'une interaction spécifique. Les auteurs précisent, que seuls les protocoles définis vont représenter des activités qui elles nécessitent des interactions entre agents.

Ainsi, l'agent peut exécuter des actions lui permettant d'accéder à différentes ressources de son environnement en dehors d'une interaction spécifique. Cette variabilité liée aux actions possibles reste néanmoins dépendante du domaine d'application, c'est pourquoi, elle n'est pas spécifiée au sein de notre modèle. En d'autres termes, ce concept d'action que nous retrouvons au niveau de Gaia [67,115], Prometheus [60], Aspecs [89], ou encore Moise+ [21], possède la même sémantique. Dans [21] par exemple, chaque action possède des arguments, des pré conditions ainsi que des effets. La seule différence, est que les actions en question seront associées à un niveau organisationnel, et par conséquent, ne présente aucune variabilité au niveau de l'agent lui-même. L'agent exécutera donc les actions de la même façon, afin qu'il puisse par exemple créer des groupes (*create_group*) au sein de son organisation, les supprimer (*remove_group*, adopter un rôle (*adopt_role*) ; ou encore s'engager dans une mission *commit_mission*. Ces éléments peuvent à priori nous laisser penser à une variabilité liée aux actions, cependant, elle est dépendante de l'organisation du système, et par conséquent cette variabilité est spécifiée au niveau du modèle d'organisation. Par exemple, l'ensemble des actions possibles de l'agent que nous avons présentée dans le chapitre précédent (voir la figure 4.14) ; présente bien une variabilité des actions possibles au sein de l'organisation de façon indépendante du domaine d'application.

5.2.3.3 Granularité des caractéristiques d'agents

Afin de proposer des caractéristiques pertinentes, il est certes important d'analyser l'ensemble des concepts, et de s'intéresser à leurs sémantiques comme nous l'avons fait ci-dessus, mais il est également important de s'intéresser à la granularité des caractéristiques.

Il est important de rappeler que cette dernière, permet de définir le degrés de modularité. Ce dernier, est un point important dans la réutilisation. Par conséquent, comme la variabilité reste une hypothèse sur la façon de distinguer les membres d'une ligne de produits logiciels (voir la définition dans le chapitre 1), il est possible d'opter pour des caractéristiques à granularité fine, grossière ou les deux.

Par exemple, le concept *capablity* couvert par MaSE [145] et Prometheus [60] ne figure pas directement au niveau de notre modèle de caractéristiques. Nous l'avons cependant associé à des caractéristiques à granularité plus fine. En effet, ce concept *capability* relatif aux aptitudes de l'agent est directement associé aux caractéristiques de *perception* (aptitude sensorielle), et *d'action* (aptitude effectrice) de l'agent. Nous avons d'ailleurs vu dans le chapitre précédent, que bien que le concept *capability* ne figurait pas au niveau des caractéristiques, il est primordial durant l'implémentation, et apparaît par conséquent au niveau du code source.

Un autre concept que nous avons choisi de ne pas lié à une seule et unique caractéristique, est celui du *comportement* d'agents. Dès lors que que ce dernier va dépendre directement des caractéristiques que nous avons déjà intégré au sein du modèle, il ne peut être dissocié de ces derniers. En effet, le comportement de l'agent va dépendre des différents choix réalisés lors d'une configuration comme nous avons pu le voir dans le chapitre précédent (voir la figure 4.19). Par exemple, l'architecture interne de l'agent va définir en partie son comportement. De plus, les rôles que devra jouer un agent au sein de son organisation vont également contribuer à la définition du comportement de ce dernier. Un autre exemple, est que pour les agents réactifs, il est possible de représenter leur comportement par une machine à état. C'est d'ailleurs, ce que nous proposons comme implémentation réutilisable pour les agents de ce type à partir du comportement FSMBehaviour de JADE.

5.2.4 Synthèse sur les caractéristiques du modèle générique d'agents

Le présent tableau 5.3 donne une synthèse sur la description de chacune des caractéristiques de notre modèle (voir la figure 5.1) ; que nous venons d'expliquer en détail au niveau de cette section.

Caractéristiques/features génériques	**Description**
Architecture	Représente une caractéristique relative au module architectural de l'agent. L'architecture représente un point de variabilité. Elle est obligatoire dès lors que tout agent en possède une qu'elle soit délibérative, réactive ou hybride.
Deliberative	Représente une caractéristique d'architecture de type délibérative permettant ainsi aux agents de raisonner et de prendre des décisions en conséquence.
Reactive	Représente une caractéristique d'architectures d'agents réactifs. Elle peut être sélectionnée seule ou avec la caractéristique délibérative afin de supporter une architecture hybride.
Knowledge_representation	Permet d'associer aux agents délibératifs une représentation des connaissances qui est un module obligatoire faisant partie d'une architecture délibérative.
Ontology	Représente un moyen de représentation de connaissances de l'agent en utilisant une ontologie. Cette dernière peut inclure l'ensemble des concepts. La présence d'une ontologie n'exclue pas la présence d'un autre moyen de représentation des connaissances de l'agent.
Belief	Représente une représentation de connaissances d'agents par le biais de ses croyances (belief). Cette caractéristique n'est pas obligatoire car n'étant pas présente pour chaque métamodèle, cependant elle le devient si le concepteur d'agents choisit une architecture de type BDI.
Reasoning_component	Représente le module obligatoire (mandatory) faisant partie d'une architecture délibérative d'agents afin de lui permettre de raisonner.
Goal	Représente l'objectif de l'agent dit Goal. Cet élément est obligatoire dès lors que le raisonnement de l'agent ainsi que son comportement se feront toujours de façon à ce qu'il atteigne son but.

Caractéristiques/features génériques	**Description**
Plan	Représente une caractéristique obligatoire permettant à l'agent d'associer les plans qu'il devra exécuter afin d'atteindre ses objectifs.
Utility	Représente une caractéristique optionnelle relative à l'ajout d'une fonction d'utilité permettant à l'agent délibératif d'avoir une mesure d'utilité et ce, pour évaluer le niveau de réussite lors de l'atteinte du but, pour obtenir des utilitaires_Based agents [10].
Mono_Goal	Représente une caractéristique relative à des agents n'ayant qu'un seul Goal à atteindre et qui n'auront donc pas à exécuter d'autres fonctions telles que OGF.
Multi_Goal	Représente des agents devant atteindre au moins deux Goal. Cette caractéristique est représentative du second algorithme de wooldridge [58].
Mono_Plan	Représente une caractéristique ne permettant à l'agent d'exécuter qu'un seul plan afin d'atteindre son but.
Multi_Plan	Représente une caractéristique permettant à l'agent une variabilité au sein des plans qu'il pourra sélectionner parmi sa base de plans et l'exécuter par la suite.
Perception	Représente le second module important du cycle générique comportemental de l'agent. Cette caractéristique permet à l'agent de percevoir son environnement ainsi que les principaux éléments qui le composent relativement à la variabilité environnementale.
Passive	Représente le composant de perception passive de l'agent. Cette caractéristique est choisie de façon alternative avec la caractéristique de perception active.
Active	Représente le module de perception active de l'agent lui permettant lors de l'exécution de capter, d'interpréter et de filtrer les perceptions qu'il aura sur ce qu'il entoure. Cette caractéristique est sélectionnée de façon alternative avec celle de perception passive.

Caractéristiques/features génériques	**Description**
Agent_Action	Représente le dernier élément obligatoire pour un agent dès lors que chaque agent effectue des actions bien que pour les agents réactifs celles-ci sont plus considérées comme des réactions aux stimuli de l'environnement. Elle représente dans tous les cas un acte émanant de l'agent qu'elle que soit la cause de son exécution. Cette caractéristique est précédée du mot Agent afin de ne pas la confondre avec la caractéristique *Action* représentée au sein du modèle de caractéristique générique d'interactions afin de représenter l'objet sur lequel porte l'interaction et non l'action en elle même exécutée à chaque pas du comportement d'agent.

TABLE 5.3 – Descriptions des caractéristiques ou features génériques d'agents

5.3 Vers un modèle générique de caractéristiques d'environnement d'agents (voyelle E)

5.3.1 Vue globale du modèle

Le modèle générique des caractéristiques d'environnement d'agents obtenu (Environnement_FM) est représenté dans la figure 5.3. Nous expliquons son contenu en détail dans ce qui suit.

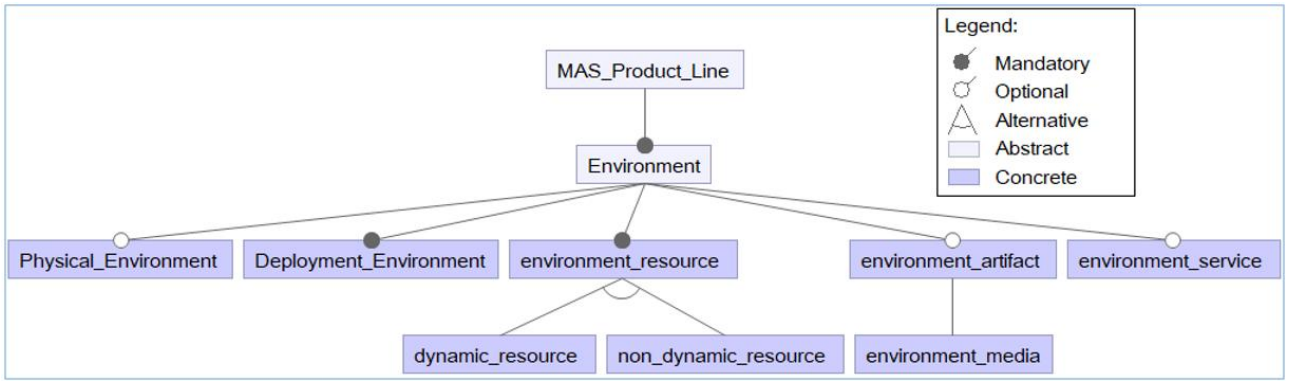


FIGURE 5.3 – Le modèle de caractéristiques d'environnement d'agents (Environnement_FM)

5.3.2 Les concepts d'environnement couverts par la délimitation du domaine

Avant de représenter les caractéristiques qui ressortent de notre analyse, nous allons présenter l'ensemble des concepts d'environnement couverts par les méta-modèles multi-agents existants en fonction de notre délimitation du domaine.

Cette dernière, est basée sur l'ensemble des méta-modèles qui prennent en compte l'environnement d'agents. Bien que certaines méthodologies populaires telles que *Moise+* [21], *Adelfe* [150], ou encore *Tropos* [151] considèrent quelques concepts de base de l'environnement, ils ne considèrent pas ce dernier comme une abstraction de premier ordre. C'est pourquoi notre domaine d'analyse comprend d'avantages de méthodologies et de modèles tels que les méthodologies *SODA* [152], et *GAIA v.2* [115] ou encore les modèles *A & A* [55] et AGRE [130], qui sont explicitement adaptés à l'environnement. L'ensemble de ces concepts est représenté dans le tableau 5.5.

Concepts	**Gaia**	**Moise+**	**Aspecs**	**A& A**	**SODA**	**AGRE**
Resource	x	x	x	x	x	
Service	x		x	x	x	
Artifact				x	x	
Physical World						x
Area						x
Space						x
Environmental Property				x	x	
Working specification				x	x	

TABLE 5.5 – Délimitation du domaine d'analyse (scoping) des concepts d'environnement à partir des méta-modèles

5.3.3 Analyse sémantique des concepts d'environnement pour le choix des caractéristiques

Dans ce qui suit nous analysons l'environnement de l'agent selon les deux points de vue présentés dans 2.4 : 1) le rôle qu'il joue au sein du système ; et 2) les aspects communs et va-

riables qu'il peut présenter. L'ensemble des éléments sur lesquels se base notre analyse regroupe également, les concepts couverts par la délimitation du domaine présentée ci-dessus.

5.3.3.1 Analyse des ressources et des services de l'environnement

1. **La caractéristique *Environment_ resource* :**

   Selon Zamboneli et al. "modelling the environment involves determining all the entities and resources that the multiagent system can exploit, control or consume when it is working towards the achievement of the organizational goal" [67]. En effet, les auteurs de l'extension de la méthodologie GAIA [115], soulignent que l'identification et la modélisation de l'environnement implique la détermination de toutes les entités et ressources que le SMA exploite, contrôle ou consume tout en faisant la distinction entre les deux catégories d'environnement physique et artificiel.

   Lors de la modélisation, il est donc nécessaire de 1) déterminer les *ressources* que les agents peuvent capter ; 2) représenter l'environnement pour déterminer comment l'agent devra percevoir cet environnement ; et 3) dans le cas de ressources dynamiques, décider pour la conception si les ressources doivent être modélisées entant qu'agents ou bien entant que ressource dynamiques de l'environnement.

   En analysant de près le concept *resource* de l'environnement, nous réalisons que ce dernier est couvert par la plupart des méta-modèles. C'est pourquoi, nous avons décidé d'intégrer au sein de notre modèle la caractéristique *Environment_ resource*. Dès lors que cette dernière présente une variabilité, nous avons également ajouté au sein du modèle les carcatéristiques *dynamic_ resource* et *non_ dynamic_ resource*.

   En nous intéressant de plus près à la sémantique associée à la définition des concepts relatifs aux ressources de l'environnement, nous remarquons que Gaia [115] considère que l'environnement peut être défini par une agrégation de ressources. Bien qu'Ingenias [57], ne fasse pas partie de notre délimitation du domaine d'environnement, ce dernier couvre le concept ressource, qu'il considère comme appartenant à un groupe. Les ressources, présentent donc une restriction d'accès. Nous retrouvons cette même restriction d'accès au niveau d'Aspecs [89], où le concept de ressource (*resource*) est directement associé au rôle de l'agent.

   Bien que le méta-modèle A& A [55] n'inclut pas le concept *resource* directement, celui-ci est coché au niveau du tableau 5.5 que nous avons construit car d'un point de vue sémantique le concept *artifact* qui représente les composants passifs du système multi-agents, peut comprendre des ressources ou encore des supports qui sont intentionnellement construits.

   En plus des ressources, il est également possible de considérer l'ontologie comme faisant

partie du modèle de l'environnement, cependant dans notre cas nous avons choisi de représenter l'ontologie au sein des caractéristiques d'agents, dès lors qu'elle peut faire partie également de la structure interne de l'agent et à partir de laquelle l'agent pourra prendre des décisions en se basant sur cette représentation des données.

2. **La caractéristique *Environment_ service* :**

   Le concept de service est généralement associé à la dimension d'agents. En effet, un agent peut être vu comme une entité capable de fournir des services à d'autres agents.

   Il est important de souligner que ce concept de service auquel nous faisons référence dans notre modèle n'a pas la même sémantique que celui-ci. Dans notre modèle, nous avons décidé d'intégrer la caractéristique *Environment_ service*, afin de faire référence à une caractéristique de l'environnement d'agent, car les artefacts décrits ci-dessus et également dans 2.4.2.1 peuvent eux même fournir des services, et par conséquent ces derniers peuvent également être vus comme faisant partie de l'environnement.

   En effet, dans le modèle A&A [55] et SODA [152] qui couvrent les mêmes concepts, les artefacts représentent des entités passives et réactives, qui sont cependant e, charge des différents services et fonctions qui permettent aux agents pris individuellement, de travailler ensemble au sein du système, mais aussi de façonner leur environnement en fonction des besoins du système ; dès lors qu'ils représentent les entités proactives du système.

5.3.3.2 Analyse des artefacts et des caractéristiques spatiales de l'environnement

1. **Les caractéristiques *Environment_ artifact* et *Environment_ media* :**

   Le concept d'artefact comme nous l'avons mentionné ci-dessus, provient du méta-modèle environnementale d'agents et d'artefacts *A & A* [55], et permet entre autres aux agents d'interagir de de façon indirecte, en s'inspirant d'objets concrets provenant du monde réel. Ce dernier est associé dans notre modèle, à la caractéristique Environment_artifact. Il est spécifié comme étant une caractéristique optionnelle car le concept d'artefact (*artifact*) n'est pas couvert par tous les modèles d'environnement d'agents.

   Le méta-modèle *A & A* [55] sera évoqué lplus tard lors de notre analyse des interactions indirectes, car il joue un rôle très important dans ce cas de figure. Les agents sont considérés comme des entités proactives, tandis que les artefacts comme des entités réactives capables d'avoir une dynamique ; tout en fournissant des services et fonctions permettant aux agents de collaborer.

   Dans *SODA* [152], des abstractions et des procédures spécifiques pour la conception d'infrastructures d'agents sont fournies. Cette méthodologie considère que l'environnement est l'espace dans lequel les agents opèrent et interagissent. SODA [152] fournit un

modèle de ressources qui modélise l'environnement d'application en termes de services disponibles, associés à des ressources abstraites. Le modèle environnemental associe les ressources aux classes de l'infrastructure. Une classe d'infrastructure est caractérisée par les services, les modes d'accès, les permissions accordées aux rôles et groupes, et les protocoles d'interaction associés à ses ressources. Les classes d'infrastructure peuvent être d'avantage caractérisées en termes d'autres caractéristiques telles que leur cardinalité (le nombre de composants d'infrastructure appartenant à cette classe), leur emplacement (par rapport aux abstractions de topologies), et leur propriétaire (qui peut être ou non le même que celui du SMA, étant donné l'hypothèse d'un contrôle décentralisé).

2. **La caractéristique *Physical_ environment* :**
   Il est possible, de considérer une métaphore de l'environnement des agents en prenant en ligne de compte les caractéristiques spatiales de l'environnement. La structure spatiale de ce dernier, peut jouer un rôle central dans la détermination des perceptions d'agents, et de leur possibilité d'interaction. Nous aborderons les mécanismes d'interactions indirectes qui lui sont associés dans la prochaine section d'analyse d'interactions (voir figure 2.14).
   Afin de donner une autre dimension à l'environnement, AGRE [130] propose d'étendre le modèle AGR [20] afin d'inclure une dimension *physique*, ou tout simplement géométrique pour l'environnement. Cette extension est basée essentiellement autour du concept *space* qui peut être vu comme une zone, mais aussi comme étant un groupe social. La caractéristique *Physical_ environment* a été ajoutée au sein de notre modèle, et a été spécifiée comme étant optionnelle car elle ne va pas faire partie de toutes les configurations possibles d'un SMA.

5.3.3.3 Granularité des caractéristiques d'environnement

De ce qui précède nous pouvons considérer comme commun à tous les environnements d'agents le fait qu'ils représentent un moyen implicite pour la communication d'agents et peuvent ainsi être considérés comme des conteneurs (container). L'environnement peut inclure des services ou des ressources en gérant l'accès à ces derniers. Il peut ainsi servir de régulateur d'accès.

L'environnement peut également fournir des *perceptions* aux agents et ces derniers peuvent agir sur leur environnement. Les environnements d'agents sont considérés comme partie intégral d'un SMA. Il est possible de distinguer deux grandes catégories d'environnement : *Artificielles* et *Physiques* (réels) [10]. Les agents étant des entités logicielles, existent au sein d'environnement logiciels artificiels ou physiques (tels que les robots). Lorsque nous considérons des environnements physiques, les objets appartenant à ce type d'environnement sont situés à une position spatiale donnée parmi l'ensemble des endroits possibles (locations) de l'environnement.

Dans le cas d'une variante minimaliste, l'environnement ne comportera qu'un agent et un ensemble d'objets. L'environnement peut de plus passé par différents états dès lors qu'il peut être modifié avec le temps. Ceci n'est autre que la réaction de l'environnement aux actions d'agents.

Dans certains cas l'environnement peut être vu comme un ensemble d'*objets* pouvant être perçus, crées, modifiés ou encore supprimés par les agents. Mais comme cela n'est pas commun à tous les environnement d'agents, il représente un point possible de variabilité. Dans ces variantes possibles, l'environnement peut faire d'une part référence à l'entité où les agents ainsi que les objets/ressources sont intégrés. Ou encore il peut représenter l'infrastructure logicielle où les agents sont déployés afin de s'exécuter. C'est pourquoi, nous avons considéré un niveau de granularité fine associée à la caractéristique *Deployment_ Environment*. Cette caractéristique n'étant associée à aucun concept issu de notre délimitation du domaine, mais que nous avons jugée être utile dès lors que notre approche peut par la suite être étendue afin de permettre de déployer les agents au sein d'environnement différents.

Il serait également possible d'intégrer une granularité encore plus fine relative aux environnements d'agents, et plus précisément aux éléments principaux des modèles d'environnement existants. Dans ce contexte, l'ensemble des propriétés clés de l'environnement proposés par Russel et al. [10] sont les suivants :

1. Accessible vs inaccessible : permet de déterminer si les agents ont un accès complet à l'état de l'environnement ou non.
2. Déterministe vs non déterministe : indique si le changement de l'état de l'environnement peut être déterminé uniquement par l'état qu'il a à un instant T, et les actions de l'agent ou pas.
3. Statique vs dynamique : détermine si l'environnement peut changer ou pas lorsqu'un agent délibère sur cet environnement.
4. Discret vs continue : détermine si le nombre de perceptions, et d'actions sont limités ou pas.

Mise à part les aspects cités ci-dessus, l'importance de l'environnement va dépendre également du type de l'interaction des agents. Dans des interactions *indirectes*, l'environnement servira de support entant que média d'interaction. Lorsque les agents coordonnent leurs activités par le biais d'interactions indirectes, il est important de pouvoir compter par exemple sur les traces laissées au sein de l'environnement. Ces aspects ont été évoqués dans ce chapitre lors de notre analyse d'interactions indirectes.

Pour finir, que la granularité de la caractéristique *Physical_ environment*, permet de regrouper plusieurs concepts tels que *Physical world, Area*, ou encore *Space*. En effet, tous ces concepts sont nécessaires et associés à la même caractéristique car ils vont permettre de supporter un environnement physique au sein de notre modèle. Rappelons, que AGRE [130] considère que

l'environnement peut être représenté sous forme de monde physique constitué de zones (*Area*) au niveau des quelles l'agent se positionne physiquement. Nous avons décidé de représenter l'environnement physique avec une granularité plus grossière que les autres certes, cependant ce choix est lié au fait que la variabilité la plus importante sera dépendante du domaine d'application, et celle-ci nécessitera d'intégrer une granularité plus fine afin de faciliter la réutilisation.

5.3.4 Synthèse sur les caractéristiques du modèle générique d'environnement

L'environnement étant la dimension la plus spécifique au domaine d'application, notre modèle générique présenté plus haut dans la figure 5.3, ne comprend donc pas un grand nombre de caractéristiques. Celles-ci que nous avons présentées plus haut sont décrites de manière synthétique au niveau du tableau 5.6.

Caractéristiques/features génériques	**Description**
Physical_Environment	Cette caractéristique représente les environnement physiques tels que nous les avons décrit plus haut. Elle est optionnelle car le monde réel n'est pas toujours représenté au sein d'un SMA. Des caractéristiques spatiales peuvent être intégrées afin de raffiner le modèle.
Deployment_Environment	Cette caractéristique représente l'environnement où les agents vont s'exécuter. Comme il est présent dans tout système dès lors que les agents déployés devront s'exécuter à ce niveau, cette caractéristique est obligatoire.
environment_resource	Cette caractéristique représente les ressources présentes au sein de l'environnement. Comme nous l'avons vu lors de notre analyse du domaine (scoping) ce concept est présent dans l'ensemble des méta-models supportant l'environnement, la caractéristique est donc obligatoire.
dynamic_resource	Cette caractéristique est sélectionnée lorsque l'ensemble des ressources de l'environnement est dynamique.
non_dynamic_resource	Cette caractéristique est sélectionnée lorsque l'ensemble des ressources de l'environnement n'est pas dynamique.
environment_artifact	Cette caractéristique représente l'artifact tel qu'il est définit dans le méta-modèle A& A et qui peut être utilisé comme moyen de coordination au sein du système. L'artifact permet de représenter les composants passifs du système (ressources, média etc) intentionnellement construits, partagés, manipulés et utilisés par les agents pour supporter leur activités coopératives ou compétitives. Les artifacts sont optionnels dès lors qu'ils ne sont pas supportés par l'ensemble des environnements d'agents.
environment_media	Cette caractéristique est sélectionnée lors de l'utilisation de média d'interaction. Ce média est une spécialisation ou raffinement du concept d'artifact.

Caractéristiques/features génériques	**Description**
environment_service	Cette caractéristique fait référence aux services qui peuvent faire partie de l'environnement dès lors qu'ils peuvent être fournis par les artifacts décrits ci-dessus. Il est important de souligner que ce concept de service n'est pas le même que celui relatif aux agents qui peuvent également fournir différents services au sein du système.

TABLE 5.6 – Descriptions des features génériques de l'environnement

5.4 Vers un modèle générique de caractéristiques d'interactions d'agents (voyelle I)

5.4.1 Vue globale du modèle

Le modèle des caractéristiques d'interaction que nous avons obtenu (Interaction_FM) suite à notre analyse, est représenté dans la figure 5.4. Dans ce qui suit, nous allons expliqué comment l'ensemble des éléments présentés au niveau de 2.5, ainsi que les différents concepts d'interactions, nous ont conduit à choisir de telles caractéristiques.

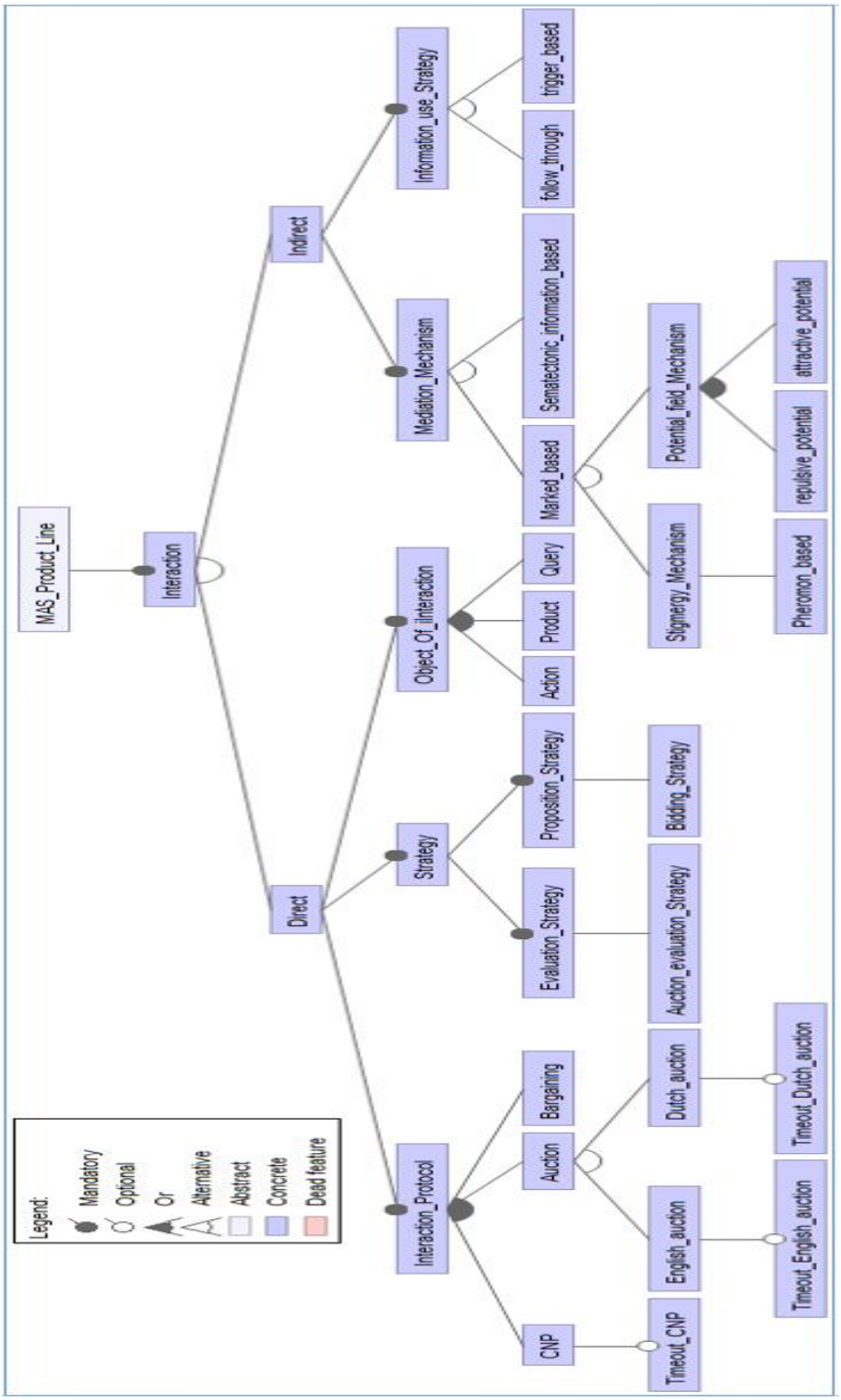


FIGURE 5.4 – Le modèle des caractéristiques d'interactions d'agents (Interaction_FM)

5.4.2 Les concepts d'interaction couverts par la délimitation du domaine

Avant d'analyser et de déduire les principales caractéristiques d'interactions d'agents, il est nécessaire de réaliser une délimitation de l'étendue du domaine (scoping), couvrant les concepts d'interactions les plus importants. Nous avons entre autres choisi différents méta-modèles existants qui couvrent des interactions directes tels que AGR [20], l'extension de GAIA [115], ou encore OperA [77]. Ainsi que l'ontologie de protocoles d'interactions *Ontopro* [84] présentée dans 2.5.1.2, afin d'étendre notre analyse du domaine. Les principaux concepts que nous retrouvons au sein de cette ontologie, sont ceux que nous retrouvons également dans les travaux sur la négociation proposés par Jenning et al. [153].

Le tableau 5.8 regroupe les principaux concepts sur les quels se base notre analyse.

5.4.3 Analyse sémantique des concepts d'interaction pour le choix des caractéristiques

Certes, il existe plusieurs dimensions et aspects de modèles d'interactions d'agents, pouvant servir à définir une taxonomie possible. Cependant, nous nous sommes basés sur la taxonomie conceptuelle d'agents réalisée à partir des différents modèles d'interaction existants (voir Figure 2.7). Celle-ci étant la plus courante, nous avons de ce fait analysé les caractéristiques respectives des interactions *directes* et *indirectes* (voir 2.5).

La principale différence entre les deux types d'interactions, réside dans le fait que la *médiation d'interactions* entre les agents se fait par *Messages* lorsqu'il s'agit d'interactions *directes*, et par le biais de *l'environnement* lorsqu'il s'agit d'interactions *indirectes*. Les interactions peuvent également présenter d'autres aspects de variation. Par exemple, le fait que les agents soient plus compétitifs, ou plutôt collaboratifs, va influer sur la nature des interactions.

5.4.3.1 Analyse des interactions directes

1. **Les caractéristiques *Direct* et *Interaction_ Protocol* :**

   Par soucis d'organisation et aussi afin de représenter les points communs et variables des interactions directes, nous avons intégré au sein de notre modèle la caractéristique *Direct*.

   Du tableau 5.8, il ressort que les protocoles d'interactions représentent la caractéristique centrale autour de laquelle s'articule l'interaction directe d'agents.

   En effet, le concept *Interaction Protocol* que nous pouvons retrouver aussi sous le nom de *AIP (Agent Interaction Protocol)*, est présent au niveau de beaucoup de méta-modèles

Concepts	Aspecs	Ingenias	OperA	AGR	GAIA	PASSI	OntoPro
Communication	x				x	x	
Message	x		x			x	
Interaction	x	x					x
Service	x				x	x	x
Interaction protocol	x		x	x	x	x	x
Communication act			x				
Strategy							x
Evaluation Strategy							x
Propose Strategy							x
Interaction Role							x
Bidding Strategy							x
Auction Evaluation Strategy							x

TABLE 5.8 – Délimitation de l'étendue du domaine d'analyse des principaux concepts d'interactions à partir des méta-modèles

multi-agents tels que AGR [20], l'extension de GAIA [115], OperA [77], ou encore PASSI [76].

Dans le méta-modèle d'AGR [20] par exemple (voir la figure 2.17), bien que celui-ci soit dédié à l'organisation d'agents, le concept de protocole d'interactions est couvert. Ce concept, est associé à un rôle d'initiateur et de participant à l'interaction.

OperA [77] considère le protocole d'interaction comme une agrégation de messages. Il en est de même pour PASSI [76], qui associe le protocole d'interaction au concept de communication qu'il considère comme étant composé de messages à base de différentes performatives. En effet, un protocole d'interactions va définir un ensemble de messages, envoyés dans un ordre précis en utilisant différentes performatives sur lesquelles se basent les agents pendant l'interaction.

2. **Les caractéristiques *CNP* et *Auction* :**

Parmi les protocoles existants, nous retrouvons le CNP (Contract Net Protocol) présenté en 1. Nous avons choisi de l'introduire au sein de notre modèle car il est très utilisé ; mais aussi car il a été standardisé par l'organisation FIPA (Foundation for Intelligent Physical Agents). Afin de permettre sa réutilisation, nous avons introduit la caractéristique *CNP* qui regroupe les différents rôles d'initiateur et de participant à l'interaction. Ces derniers représentent des rôles génériques et par conséquent réutilisables indépendamment du domaine d'application. La sélection d'une telle caractéristique, nécessite également de sélectionner les stratégies à utiliser par chacun des deux rôles. C'est pourquoi, nous avons réutilisé les concepts de l'ontologie OntoPro [15], et avons intégré au sein de notre modèle entre autres les caractéristiques *Strategy*, *Evaluation_ Strategy* ; et *Proposition_ Strategy.*

Notre analyse, s'est focalisée également sur le déroulement de chacun des protocoles, leurs paramètres d'entrée et de sortie, ainsi que les caractéristiques des différents rôles d'interaction.

De la description du Contract-Net, il ressort que la cardinalité des participants peut influer sur son exécution. Par exemple, dans le cas où la cardinalité est de 1 :1, le rôle initiateur du contract-Net peut ne pas utiliser sa stratégie d'évaluation. De plus, le service sur lequel porte l'interaction, et qui comme expliqué représente l'objet de l'interaction peut varier.

Les paramètres tels que le *Timeout* représentent des options au sein du protocole. Mise à part ces aspects, de notre analyse ressort que les stratégies de proposition et d'évaluation utilisées respectivement par les participants et l'initiateur présentent un point de variabilité au sein du protocole.

Mise à part le protocole CNP, nous avons également intégré d'autres protocoles d'interactions tels que celui de vente aux enchères présenté dans 2, qui est directement associé

à la caractéristique *Auction*. La variabilité liée à ce protocole d'interaction est spécifiée par un choix alternatif à réaliser entre un protocole d'enchères Anglaises et hollandaise. Les caractéristiques associées sont décrites au niveau du tableau 5.9 récapitulatif.

La variabilité liée au protocole d'enchères Anglaises a été détaillée dans le chapitre précédent, en présentant les implémentations réutilisables proposées (voir 4.3.2.3).

Quant aux paramètres de chacun des protocoles analysés, ils peuvent être obligatoires ou optionnels, comme le prix de réserve dans le cas de vente aux enchères.

Pour finir, les aspects négociés, concernent un ou plusieurs attributs de l'objet de la négociation comme le prix d'un produit sur lequel porte l'interaction.

3. **La caractéristique *Object_ Of_ interaction* :**

   Sur la base des concepts utilisés au sein de l'ontologie OntoPro [15], mise à part les caractéristiques déduites précédemment, nous avons également integré au sein du modèle la caractéristique *Object_ Of_ interaction*. Cette dernière est directement associé au concept de *service* utilisé dans le contexte de protocoles d'interactions. C'est-à-dire qu'il possède une sémantique différente du concept de service que nous avons associé à la caractéristique *environment_ service* durant notre analyse de l'environnement d'agent présentée ci-dessus.

   Le concept de *service* auquel fait référence l'ontologie est directement associé au concept *Interaction Role* présent au niveau du tableau 5.8. Un rôle d'interaction est donc capable de fournir un service ou encore de demander l'accès au service. Ce concept de service fait référence à l'objet de l'interaction qui peut par conséquent représenter un produit, une action ou encore une requête. C'est pour ses raisons, que nous avons spécifié cette variabilité au sein de notre modèle en intégrant les caractéristiques *Product, Action*, et *Query*. Celles-ci sont spécifiées par une variabilité de type choix "Ou", car elles peuvent toutes faire partie d'une même configuration.

   Ce même concept de *Service*, est également couvert par PASSI [76], et Aspecs [89] qui l'associent également au concept *Role*. Gaia [115], est la seule méthodologie de notre délimitation du domaine, à associer directement ce concept de *service* non pas au rôle de l'agent directement, mais à l'agent (*Agent Type*). Par conséquent, il est possible d'associer ce concept à la dimension d'agent, ou encore à celle de l'interaction. Mais, afin de faciliter la réutilisation et le paramétrage des rôles d'agents durant l'interaction directe, nous avons choisi de l'intégrer au niveau de la dimension d'interaction.

5.4.3.2 Analyse de des interactions indirectes

Comme nous avons pu le voir pendant l'analyse de l'environnement d'agents, dans tous les cas d'interactions indirectes, l'environnement jouera un rôle primordial. Cependant, nous

constatons qu'aucun concept relatif aux interactions indirectes n'est considéré dans l'ensemble des méta-modèles de notre délimitation du domaine. Les caractéristiques principales que nous avons déduites, et intégrées dans notre modèle, sont issues des éléments présentés dans 2.5.2.

1. **La caractéristique *Mediation_ Mechanism* :**

   De notre analyse des principales caractéristiques des interactions indirectes, il ressort que ces dernières ont lieu grâce à un intermédiaire ou média, qui peut faire partie de l'environnement, ou être représenté par l'environnement lui-même. Les types de médias possibles ont été décrits lors de notre analyse des caractéristiques de l'environnement d'agents, tels que les artefacts. Ces derniers, représentent un composant primordial vue qu'ils médiatisent l'interaction.

   L'interaction indirecte présente ainsi une variabilité liée aux mécanismes d'interaction. C'est pourquoi, nous avons intégré au sein de notre modèle la caractéristique *Mediation_ Mechanism*. La variabilité liée aux mécanismes d'interactions est spécifiée par un choix alternatif entre les deux prochaines caractéristiques que nous présentons.

2. **Les caractéristiques *Marked_ based* et *Sematectonic_ information_ based* :**

   En analysant de plus près les mécanismes d'interactions indirectes, nous remarquons qu'il est possible de spécifier leur variabilité par deux choix alternatifs : les *marqueurs*, et les *informations sematectoniques* [154]. C'est pourquoi, nous avons ajouté au sein de notre modèle les caractéristiques respectives correspondantes *Marked_ based* et *Sematectonic_ information_ based*.

   Dans les interactions basées sur les *marqueurs* (*Marked_ based*), le mécanisme de *stigmergie* par exemple peut être utilisé. Nous avons associé ce concept à la caractéristique alternative *Stigmery_ Mechanism*. Ce mécanisme d'interaction, relève de l'intelligence en essaim, qui s'inspire principalement de la biologie pour les approches à base de phéromones (voir 2.5.2.2). Ce qui permet aux agents, de s'échanger des traces sous forme de substances chimiques telles que les *phéromones*.

   D'autres marqueurs peuvent être utilisés, tels que les signaux quand il s'agit d'approche des *champs potentiels*. Ces derniers, s'inspirent de la physique (voir 2.5.2.1). Nous avons associé ce mécanisme à la caractéristique alternative *Potential_ Field_ Mechanism*.

   Sur la base des éléments présentés en 2.5.2.1, nous avons ajouté deux caractéristiques spécifiées par un choix à faire parmi les caractéristiques *repulsive_ potential* et *attractive_ potential*. La première, permet d'utiliser un mécanisme avec une force de répulsion. Tandis que la deuxième, permet d'utiliser un mécanisme avec une force d'attraction.

   La particularité des ces *marqueurs*, est qu'ils présentent une même sémantique pour chaque récepteur. Tandis qu'avec les *informations sematectoniques*, les agents tentent de fournir des informations implicites sans qu'il y est pour autant une sémantique partagée

spécifique pour les agents. Un exemple, serait qu'un agent empile des objets à un endroit spécifique. Ce qui pourrait, influencer les autres agents à poursuivre ce travail. Un autre facteur de variabilité, serait lié à l'exploitation de l'information. En effet, les agents peuvent interpréter des signaux et agir en conséquence. Par exemple, les agents peuvent être avertis d'un danger par des signaux. Alternativement, le comportement des agents peut être influencé par les marques laissées, en les incitant à réaliser une suite d'actions. Par exemple, les agents pourraient suivre les traces laissées par un autre agent.

5.4.3.3 Granularité des caractéristiques d'interactions

Pour les interactions directes, nous avons montré que le protocole d'interaction jouait un rôle centrale. La variabilité, est représentée par les différentes stratégies de décision que l'agent interactif peut utiliser. Chacune des caractéristiques choisies à ce niveau, est associée à un seul et unique concept, et présente une granularité fine et élémentaire comme les caractéristiques *Evaluation_ Strategy*, et *Proposition_ Strategy*.

Par conséquent, les stratégies de l'enchérisseur du protocole d'enchères Anglaises, sont également représentées de façon générique et avec une granularité fine pour les caractéristiques choisies.

Que cela concerne les interactions directes ou indirectes, la coordination reste un aspect important des interactions d'agents et aide à définir la granularité des caractéristiques choisies. Dans le cas de coordination d'interactions indirectes, il y-a différentes caractéristiques que nous pouvons distinguer au sein des *infrastructures de coordination* chargées de médiation par le biais de l'environnement. Par exemple, si les agents exploitent la *stigmergie* pour communiquer en déposant des phéromones dans un environnement. Ce mécanisme à base de phéromones, est associé dans notre modèle à la caractéristique *pheromon_ based*. Dans ce contexte, l'infrastructure de coordination sera responsable de l'évaporation du *phéromone* en question. Cependant, nous avons remarqué que le rôle le plus important dans ce type d'interactions médiatisées, reste celui joué par l'*environnement*. En effet, dans ce type d'interactions, l'environnement devient alors une entité active du système en régulant son activité. Ainsi, ce dernier pourra lui-même attribuer des activités à des ressources indépendantes des activités des agents. Dans le modèles à base de phéromones par exemple, l'environnement supporte différentes activités telles que *l'agrégation*, *la diffusion* et *l'évaporation* du phéromone. La granularité de la caractéristique *pheromon_ based* choisie, fait intervenir des aspects relatifs aux processus d'évaporation et de diffusion qui seront tous encapsulées au niveau de cette même caractéristique .

5.4.4 Synthèse sur les caractéristiques du modèle générique d'interactions

L'ensemble des caractéristiques que nous avons intégrées dans la partie du modèle de caractéristiques relative aux interactions d'agents sont représentées de façon synthétique au niveau du tableau 5.9.

Caractéristiques/features génériques	**Description**
Direct	Représente une caractéristique décrivant une interaction directe. Cette feature est sélectionnée de façon alternative avec celle définissant une interaction indirecte.
Indirect	Représente une caractéristique décrivant une interaction indirecte. Cette feature est sélectionnée de façon alternative avec celle définissant une interaction directe.
Interaction_Protocol	Représente la caractéristique permettant de définir un protocole d'interaction dans le cas d'interaction directes. Elle est obligatoire (mandatory) car elle est nécessaire dans la définition d'interactions directes. Dans le cas de la sélection de cette caractéristique, il faudra choisir au moins un protocole parmi ses caractéristiques filles.
CNP	Représente le protocole du contact-net, celle-ci pouvant être exclue ou incluse dans une suite de protocoles nécessaire au sein d'une variante donnée.
Timeout_CNP	Représente l'option du Timeout qu'il est possible d'intégrer au sein d'une variante multi-agents utilisant le protocole du contract-net.
Auction	Représente les protocoles d'enchères de façon générale, celle-ci pouvant être spécifiée en sélectionnant un des protocles d'enchères possibles tel que l'enchère Anglaise.
English_Auction	Représente le protocole du english-auction, celle-ci pouvant être exclue ou incluse dans une suite de protocoles nécessaire au sein d'une variante donnée.
Timeout_English_Auction	Représente l'option du Timeout qu'il est possible d'intégrer au sein d'une variante multi-agents utilisant le protocole d'enchères anglaises.

Caractéristiques/features génériques	**Description**
Ducth_Auction	Représente le protocole du dutch-auction, celle-ci pouvant être exclue ou incluse dans une suite de protocoles nécessaire au sein d'une variante donnée.
Timeout_Dutch_Auction	Représente l'option du Timeout qu'il est possible d'intégrer au sein d'une variante multi-agents utilisant le protocole d'enchères allemandes.
Bargaining	Représente le protocole de marchandage, celle-ci pouvant être exclue ou incluse dans une suite de protocoles nécéssaire au sein d'une variante donnée. Nous considérons la version la plus simple de ce protocole afin d'illustrer notre travail.
Strategy	Correspond aux stratégies décisionnelles d'agents telle qu'elle a été définie dans l'ontologie ONTOPRO que nous réutilisons. Elle est obligatoire au sein des variantes interagissant de façon directes.
Evaluation_Strategy	Correspond aux stratégies décisionnelles relatives à une évaluation donnée de la part d'agents telle qu'elle a été définie dans l'ontologie ONTOPRO que nous réutilisons. Elle est obligatoire au sein des variantes interagissant de façon directes dès lors que le protocole nécessitant son exécution est sélectionné.
Auction_Evaluation _Strategy	Représente la stratégie d'évaluation utilisée par l'agent participant à une vente aux enchères et devant évaluer une offre faite lors d'un protocole de vente aux enchères. Cette caractéristique doit être intégrée au sein de la variante multi-agents dans le cas de sélection de la caractéristique du protocole d'enchères.
Proposition_Strategy	Correspond aux stratégies décisionnelles relatives à une proposition faite de la part d'agents telle qu'elle a été définie dans l'ontologie ONTOPRO que nous réutilisons. Elle est obligatoire au sein des variantes interagissant de façon directes dès lors que le protocole nécessitant son exécution est sélectionné.

Caractéristiques/features génériques	**Description**
Bidding_Strategy	Représente la stratégie de proposition utilisée par l'agent participant à une vente aux enchères et devant proposer une offre lors d'un protocole de vente aux enchères. Cette caractéristique doit être intégrée au sein de la variante multi-agents dans le cas de sélection de la caractéristique du protocole d'enchères.
Objetc_Of_interaction	Correspond aux services relatifs aux interactions en représentant l'objet de l'interaction quelque soit son type. La caractéristique est obligatoire au sein des variantes interagissant de façon directes car il ne peut y-avoir d'interactions directes sans objet d'interaction. Elle est relative au concept inclue au sein de l'ontologie ONTOPRO que nous réutilisons.
Action	Cette caractéristique est sélectionnée lorsque l'objet d'interaction représente une action à exécuter.
Product	Cette caractéristique est sélectionnée lorsque l'objet d'interaction représente un produit donné.
Mediation_mechanism	Représente l'élément principal et obligatoire (mandatory) caractérisant une interaction indirecte entre agents. Il représente les différents mécanismes possibles pour la médiation.
Marked_based	Représente les mécanismes d'interactions indirectes à base de marqueurs tels qu'ils ont été décrits précédemment. Elle est sélectionnée de façon alternative avec la caractéristique Sematectonic_information_based.
Sematectonic_information_based	Représente les mécanismes d'interactions indirectes à base d'informations sematectoniques dont la sémantique est implicite. Elle est sélectionnée de façon alternative avec la caractéristique Marked_based.
Stigmergy_Mechanism	Représente l'interaction indirecte qui a lieu grâce au mécanisme de stigmergie décrit plus haut. Elle est sélectionnée alternativement aux mécanismes de champs potentiels.
Pheromon_based	Représente une spécialisation de la caractéristique de stigmergy. Elle permet aux agents d'interagir de façon indirectes en laissant des traces de phéromones qui s'évaporent avec le temps.

Caractéristiques/features génériques	**Description**
Potential_field_Mechanism	Représente l'interaction indirecte qui a lieu grâce aux mécanisme de champs potentiels décrit plus haut. Elle est sélectionnée alternativement aux mécanismes de stigmergie.
Repulsive_potential	Permet d'utiliser un mécanisme de champs potentiels avec une force de répulsion pour les agents en interaction.
Attractive_potential	Permet d'utiliser un mécanisme de champs potentiels avec une force d'attraction pour les agents en interaction.
information_use_strategy	Représente la stratégie utilisée lors de l'utilisation de l'information. Elle possède deux caractéristiques filles alternatives follow_through et trigger_based.
Follow_through	Représente une alternative d'utilisation de l'information par les agents qui induit les agents à exécuter un ensemble d'actions. Tel que des agents qui suiveraient les traces qu'un autre agent laisserait dernière lui sur son chemin.
Trigger_based	Représente une alternative d'utilisation de l'information par les agents qui est dirigée par des déclencheurs d'événements. Tel que les signaux audio perçus par des agents les pousseraient à fuir un danger.

TABLE 5.9 – Descriptions des caractéristiques génériques d'interactions

5.5 Vers un modèle générique de caractéristiques d'organisations d'agents (voyelle O)

5.5.1 Vue globale du modèle

La dernière voyelle mais pas la moindre, correspond à l'organisation du SMA. Le modèle générique des caractéristiques (features) d'organisation que nous avons obtenu (Organisation_FM), est représenté dans la figure 5.5. Dans ce qui suit, et comme nous l'avons fait pour les autres voyelles, nous expliquons la provenance de chacune des caractéristiques choisies au sein du modèle.

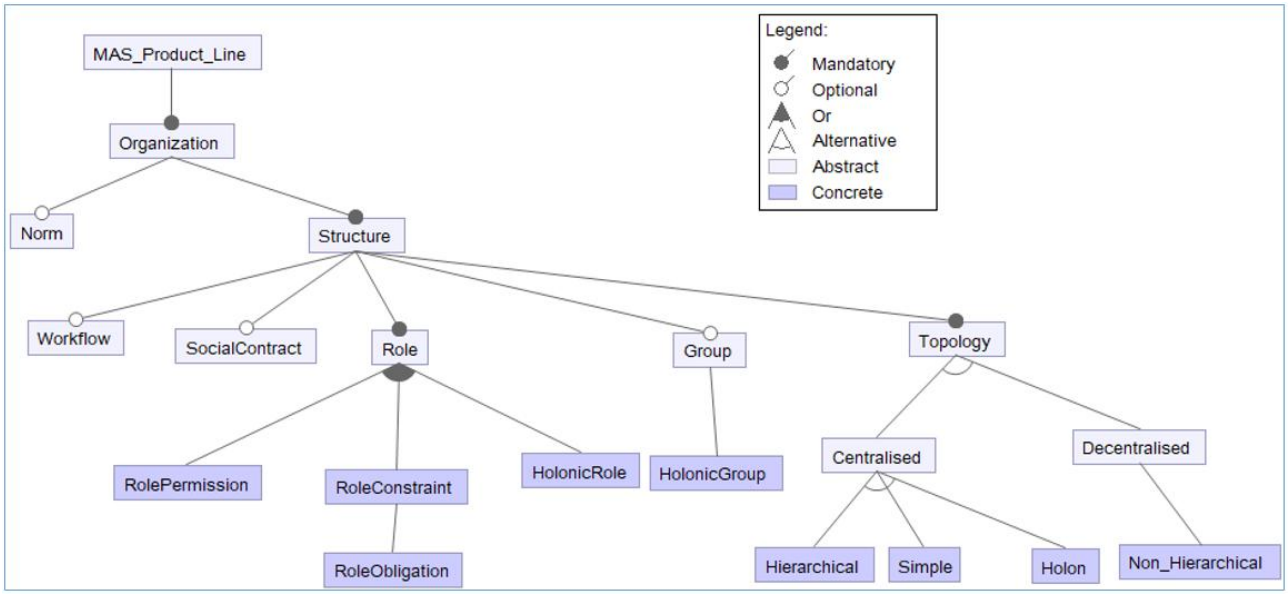


FIGURE 5.5 – Le modèle des caractéristiques d'organisation d'agents (Organisation_FM)

5.5.2 Les concepts d'organisation couverts par la délimitation du domaine

Notre délimitation du domaine a été réalisée de façon à inclure les principaux concepts de l'organisation ; mais aussi des modèles permettant d'offrir différentes vues de l'organisation, tel que Moise+ (voir 2.6.2).

Dès lors que le présent travail vise à faire un pas vers un modèle générique de caractéristiques d'organisation, nous avons limité notre étendue du domaine analysé. Il existe des propriétés, que nous n'avons pas traitées. Par exemple, il est possible de modéliser l'organisation, par le biais de différents mécanismes ; parmi les quels nous retrouvons des mécanismes d'auto-organisation inspirés de phénomènes biologiques naturels. Ces mécanismes d'auto-organisations, sont utilisés dans différents domaines principalement pour les applications industrielles [155]. Ainsi, que des travaux également dans le domaine du Grid computing [156]. L'auto-organisation, est entre autre efficace pour traiter les besoins dynamiques des systèmes distribués. Mais, les aspects de l'auto-organisation du système ne font pas partie de notre travail.

Les concepts de l'organisation couverts par notre délimitation du domaine sont présentés au niveau du tableau 5.11. L'aspect sémantique lié à ces concepts, est présenté dans ce qui suit.

5.5.3 Analyse sémantique des concepts d'organisations pour le choix des caractéristiques

Notre analyse de l'organisation est basée d'une part sur la sémantique des concepts présentés dans le tableau 5.11 ; et d'autres part sur l'ensemble des éléments présentés dans 2.6.

Nous nous sommes basés entre autres sur la vision de l'organisation proposée par Moise+ [21], qui est présentée dans 2.6.2 du chapitre 2.

La principale raison de ce choix, est que ce modèle englobe trois différentes vues de l'organisation (*structurelle*, *fonctionnelle* et *normative*) ; et couvre ainsi différents aspects.

Concepts	**Ingenias**	**Aspecs**	**AGR**	**Moise+**	**Gaia**	**MaSE**	**OperA**
Role	x	x	x	x	x	x	x
Role Constraint		x	x				
Responsability					x		
Organisational Rule		x			x		
Permission					x		
Role Capabilities		x				x	
Policy						x	
Social Contract							x
Norm							x
Workflow	x						
Group	x	x	x	x			
Holonic Role		x					
Holonic Group		x					
Holon		x					
Role Plan		x		x			
Role Task		x		x			

TABLE 5.11 – Délimitation de l'étendue du domaine d'analyse des principaux concepts (scoping) des concepts d'organisation à partir des méta-modèles

Sur cette base de la vue générique de l'organisation des aspects clés proposés par Moise+, nous avons analysé les modules suivants : 1) la structure de l'organisation avec les différentes

topologies qu'elle peut supporter, 2) l'aspect fonctionnel de l'organisation ; et 3) les règles liées à l'organisation.

5.5.3.1 Analyse de la structure et de l'aspect fonctionnel de l'organisation

1. **Les caractéristiques *Role* et *Group* :**

   Dès lors que la vue organisationnelle, permet entre autres d'associer un ou plusieurs rôles à un ou plusieurs agents ; nous avons décidé d'introduire des caractéristiques relatives aux rôles d'agents. L'ensemble des informations liées aux rôles de l'organisation, est associé à la caractéristique *Role* de notre modèle.

   Le rôle étant un composant obligatoire et primordial au sein d'une organisation, nous avons spécifié la caractéristique liée aux rôles comme étant obligatoire. En effet, le concept de rôle est présent dans tous les méta-modèles analysés (voir le tableau 5.11.

   La variabilité liée aux rôles est abordée plus bas, et concerne par exemple les caractéristiques relatives aux permissions qui s'appliquent aux rôles.

   Il est important de noter qu'un rôle décrit entre autres différentes contraintes, telles que les besoins, les compétances (skills), ou encore les obligations qu'un agent devrait présenter afin de satisfaire l'obtention du rôle [20]. Nous avons associé le concept *Role Constraint* proposé par AGR (voir la figure 2.17) à la caractéristique correspondante *Role_ Contraint*.

   Différentes façons peuvent servir à structurer l'organisation. Il est possible par exemple de regrouper les rôles en groupes. Les méta-modèles supportés par AGR [20], Moise+ [157] [21], Ingenias [57] et Aspecs [89] structurent l'organisation en utilisant ce concept de groupe. Ce dernier, a été associé dans notre modèle à la caractéristique *Group*. Cette dernière étant spécifiée comme étant optionnelle, car elle n'est pas présente au sein de tous les méta-modèles analysés.

2. **Les caractéristiques *Workflow* et *Social_ contract* :**

   Un autre concept intéressant, est celui de *workflow* qui est supporté par Ingenias [57]. L'organisation est vue comme étant composée de workflow, composés à leur tour de tâches (*Task*). En effet, Ingenias [57] décompose l'organisation en sous groupes d'agents et de workflow. Cette décomposition, pourrait convenir aussi bien pour des organisations *Hierarchiques*, que des organisations en *Equipes* (*Team*) ; ou encore des *Coalitions*. L'exécution du *workflow*, permettra de guider le comportement d'agents devant accomplir des tâches au sein de l'organisation. Ainsi, lensemble des activités et informations échangées entre les différentes parties du système mutli-agents, seront définies. D'un point de vue sémantique, le workflow va représenter des éléments tels que les *plans* devant être exécutés au sein de l'organisation. Nous avons associé cette option permettant d'utiliser des

workflow, à la caractéristique *Workflow*.

D'autres approches comme OperA [77] proposent de considérer l'organisation à travers la mise en place de plusieurs contrats sociaux. Le concept *Social Contract* correspondant et présent au niveau du tableau 5.11, est une agrégation de différentes clauses. Ce même contrat social, permet d'établir un accord entre l'agent et son modèle organisationnel, tout en définissant la façon dont l'agent va accomplir son rôle. Nous avons associé ce concept à la caractéristique *Social_ Contract*, qui est spécifiée comme étant optionnelle, étant donné qu'elle n'est sélectionnée que si l'organisation détermine un contrat social.

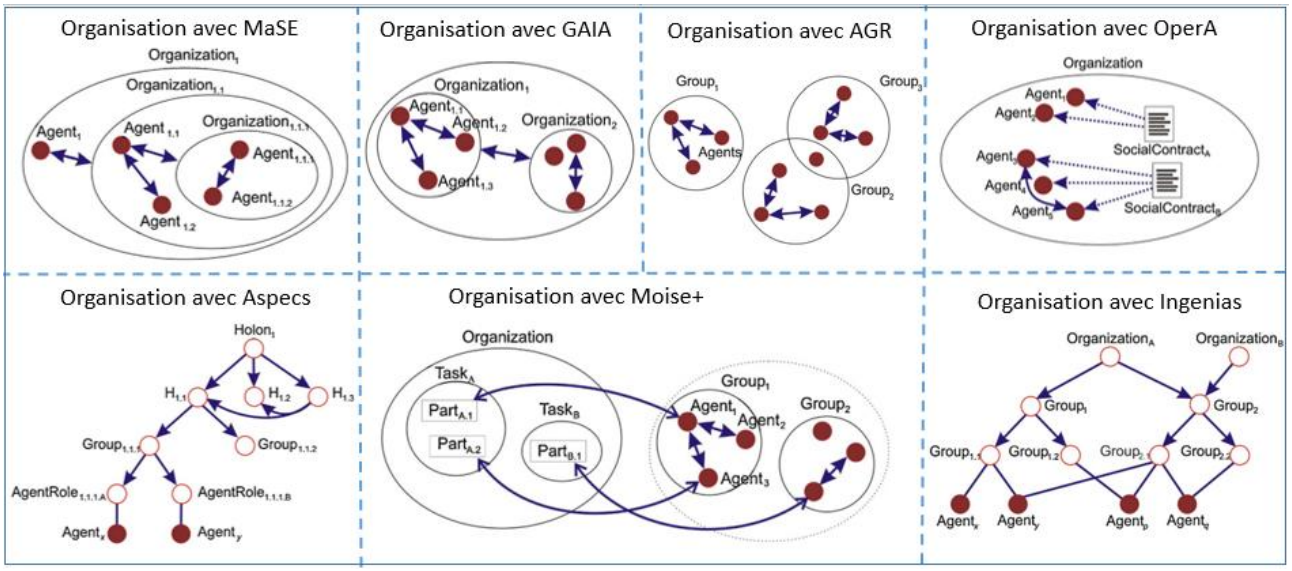


FIGURE 5.6 – Les topologies d'organisation supportées par les méta-modèles du domaine d'analyse

3. **Les caractéristiques *Topology, Centralised*, et *Dentralised* :**

   L'ensemble des méta-modèles existants permet de supporter différentes topologies au sein de l'organisation. Des exemples de topologies relatives aux méta-modèles issus de notre délimitation du domaine d'analyse de l'organisation, sont présentés dans la figure 5.6.

   Nous avons donc intégré au sein du modèle, la caractéristique *Topology* ; qui est spécifiée par une variabilité liée aux caractéristiques *Centralised* et *Decentralised*.

   Ces deux caractéristiques définissent les deux grandes catégories de topologies : (i) *Centralisées* qui comportent les *Hiérarchies*, et les *Holarchies* ; et (ii) *Décentralisées*, qui sont des organisations plates (non-hiérarchique), et qui sont utilisées pour des organisations co-opératives telles que les *Coalitions*, *Equipes* et *Congrégations*.

   La méthodologie ASPECS [89] par exemple, est basée sur le méta-modèle d'organisation

CRIO, qui étend le méta-modèle RIO [158]. Elle permet de supporter des organisations holoniques. De plus, nous considérons des variantes de topologies *Holarchiques* comportant les topologies *Markets* pour des organisations compétitives, ainsi que les topologies *Fédérations* pour les organisations pas très compétitives.

(ii) *Décentralisées* qui sont utilisées pour des organisations coopératives telles que les *Coalitions*, *Equipes* et *Congrégations*. Une autre topologie qui s'inscrit également dans cette seconde catégorie considère l'organisation sous forme de société ou *Society* nécessitant l'utilisation de *SocialContract* afin d'être supportée par le SMA. De ce fait, le rôle est étendu par le biais de *RoleEnactment* représenté par un role-enactment agent (REA) comme représenté au sein du méta-modèle OperA [77].

5.5.3.2 Analyse des règles de l'organisation

1. **La caractéristiques *Norm* :**

   L'aspect structurel de l'organisation ne suffit pas toujours. L'organisation, doit intégrer dans certains cas des règles organisationnelles relatives au concept *Organizational Rule* supporté par certains méta-modèles comme celui de GAIA [115], ou encore celui d'Aspecs [89].

   Bien n'étant pas supporté par certains méta-modèles tel que AGR [20] ou encore INGENIAS [57], il reste indispensable au sein de modèles d'organisations tel que Moise+ [21], OperA [77] ou encore MaSE [145]. Cet aspect figure cependant sous différentes formes et présente ainsi une variabilité.

   L'aspect normatif incluant des normes instaure au sein de l'organisation une nouvelle dimension permettant de définir des règles sociales. C'est pourquoi, nous avons placé au sein de notre modèle la caractéristique *Norm*.

   Les normes dans OperA [77] définissent les droits et obligations des agents, et qui sont directement liés aux rôles que les agents doivent jouer. Les normes des rôles spécifient donc les règles de comportement respectives aux scènes (scene) d'interactions.

   Un autre aspect de règles régissant l'organisation représente les politiques d'affectation d'agents aux rôles. Dans MaSE [145] par exemple, ces politiques associées au concept *Policy*, limitent l'affectation des agents aux rôles (potentiel) contrôlant ainsi les états admissibles ou non de l'organisation. Ces politiques posent ainsi des contraintes au niveau des rôles.

2. **Les caractéristiques *Role_ Obligation* et *Role_ Permission* :**

   Pour tenir compte de la variabilité des rôles de l'organisation indépendamment du domaine d'application, nous avons introduit entre autres au sein de notre modèle les permissions et contraintes liées aux rôles.

GAIA [115] par exemple, supporte le concept *Permission* afin de restreindre l'accès des rôles aux ressources sur lesquelles ils agissent ou avec lesquelles ils interagissent. Un rôle ne peut donc accéder à une ressource que s'il en a la permission. La caractéristique *Role_ Permission* que nous avons introduite permet de supporter une telle variabilité. Dans MaSE [145], le rôle requiert des aptitudes (capabilities) pour mener à bien un tel rôle.

Parmi les contraintes liées aux rôles, ce dernier peut par exemple avoir des obligations. Les obligations entrent dans le cadre d'une logique déontique de l'organisation. En effet, les normes intégrées au sein de l'organisation, indiquent que le comportement attendu par un rôle donné utilise des normes qui sont directement traduites en obligations, prohibitions et permissions. C'est pourquoi, nous avons spécifié cette variabilité en intégrant au sein du modèle la caractéristique *Role_ Obligation*

Moise+ [21] définit aussi l'aspect déontique de l'organisation par le biais de *permissions* et *d'obligations*. Une permission, va indiquer qu'un agent qui joue un rôle donné est autorisé et a donc la permission de s'engager dans une mission donnée, de façon à ce que cette autorisation reste valable en fonction de la contrainte relative à cette permission. Une obligation quant à elle, va indiquer qu'un agent doit s'engager dans une mission donnée pendant des périodes données.

5.5.3.3 Granularité des caractéristiques d'organisations

De notre analyse ressort que la vue proposée par *Moise +* [21] regroupe les composants primordiaux de l'organisation d'agents s'articulant autour des trois points de vue : *structurel*, *fonctionnel* et *normatif*. La première caractéristique commune des organisations d'agents est relative aux structures.

Afin d'avoir une bonne modularité au sein du composant organisationnel du SMA, nous avons choisi principalement des caractéristiques issues de concepts à granularité fine telles que la caractéristique *Role*. L'organisation structurelle minimale de tout type d'organisation doit en effet définir les rôles au sein de cette organisation comme nous avons pu le voir ci-dessus.

De façon optionnelle, il est possible de structurer une organisation d'agents en intégrant des groupes de rôles au sein de l'organisation.

Une autre caractéristique commune aux structures d'organisations d'agents est que ces dernières peuvent se présenter sous formes de topologies variables. Une topologie pouvant être de façon alternative centralisée ou décentralisée. Les topologies centralisées peuvent être simples, hiérarchiques ou bien holoniques.

Concernant l'aspect déontique de l'organisation, nous avons également procédé à un découpage maximal, afin de déduire des caractéristiques à granularité fine telles que les permissions ou

encore les obligations liées aux rôles de l'organisation ; et que nous avons associé respectivement aux caractéristiques *Role_ permission* et *Role_ Obligation*.

5.5.4 Synthèse sur les caractéristiques du modèle générique d'organisation

L'ensemble des caractéristiques que nous avons présenté et proposé afin de déterminer les similitudes et la variabilité des organisations d'agents, est synthétisé dans le tableau 5.12.

Caractéristiques/features génériques	**Description**
Structure	Représente la structure de l'organisation permettant d'identifier entre autre sa topologie.
Role	Représente le role devant être joué par un ou plusieurs agents au sein de l'organisation.
Role Constraint	Représente l'ensemble des contraintes sur le rôle tels définis par AGR [20]. Cette caractéristique décrit les contraintes en terme de besoins et obligations que l'agent doit satisfaire afin d'obtenir le rôle.
Role Obligation	Représente un raffinement des contraintes de rôles. En effet les obligations du rôle représentent une des contraintes liées aux rôles d'organisation.
Role Permission	Représente la caractéristique relative aux permissions que possède un rôle.
Norm	Représente un module optionnel de l'organisation bien qu'il soit important. Ce concept fait référence en plus des normes, à l'ensemble des concepts instaurant des règles au niveau de l'organisation tels que le concept OrganizationalRule défini par GAIA [115]. De plus que les rôles possèdent des obligations relatives aux règles à respecter au sein de l'organisation.
Holonic Role	Représente le rôle d'holon pouvant être joué par un agent. Les agents et les groupes définissent les holons.
Holonic Group	Représente la possibilité d'intégrer au sein de l'organisation des catégories de groupes holoniques. Ce type de groupes est composé de rôles holoniques.

Caractéristiques/features génériques	**Description**
Group	Représente la possibilité de représenter les groupes (option) au sein de l'organisation.
Social Contract	Représente une option qui permet au sein de l'organisation d'établir un accord entre l'agent et son modèle organisationnel en définissant la façon dont l'agent va accomplir son rôle.
Workflow	Représente une option permettant de décomposer l'organisation en sous groupes d'agents et de workflows relatifs au flux de travail.
Topology	Représente un module obligatoire qui permet de structurer l'organisation de façon centralisée ou décentralisée.
Centralised	Représente un choix alternatif afin de supporter une topologie d'organisation centralisée.
Simple	Représente un choix alternatif pour des organisations centralisées simples, qui ne présentent donc aucune hiérarchie ou structure sous forme holonique au sein de l'organisation.
Hierarchical	Représente un choix alternatif afin de supporter une topologie d'organisation centralisée de façon hiérarchique.
Holon	Représente un choix alternatif afin de supporter une topologie d'organisation holonique. Ainsi le système peut être découpé de façon récursive avant répartition aux holons. Un holon peut jouer plusieurs rôles au sein de différents groupes.
Decentralised	Représente un choix alternatif afin de supporter une topologie d'organisation décentralisée.
Non_Hierarchical	Représente l'ensemble des organisations décentralisées possibles comme par exemple une organisation en coalition qui est dynamique et donc capable de se former puis de se dissoudre selon le besoin.

TABLE 5.12 – Descriptions des caractéristiques génériques d'organisation

5.6 Contraintes du modèle de similitudes et de variabilité des SMA

Une fois l'ensemble des caractéristiques communes et variables des SMA identifiées pour chacune des dimensions de l'approche VOYELLES [28], celles-ci ont été regroupées au sein du modèle de variabilité des SMA, représentant ainsi la ligne de produits multi-agents indépendamment du domaine d'application.

Ce modèle de caractéristiques sert de base pour la construction d'un modèle de caractéristique spécifique au domaine d'application. Comme nous l'avons présenté dans le chapitre 4, ce modèle fait partie de l'approche MAS-PL que nous proposons.

Afin de garantir un modèle cohérent, à chacune des étapes de construction des modèles intermédiaires correspondant aux voyelles A (Agent), E (Environnement), I(Interaction) et O(Organisation) ; nous avions due intégré les contraintes permettant d'exclure des configurations non valides. Quant à l'intégration de tous les modèles intermédiaires au sein du modèle global (voir la figure 4.3) ; il a fallu également ajouter des contraintes afin de garantir une validité de la sélection des caractéristiques avant la dérivation de systèmes multi-agents personalisés.

La figure 5.7 présente quelques exemples des contraintes relatives à notre modèle de variabilité multi-agents. La première contrainte par exemple, indique que la présence de normes nécéssite la définition des obligations ainsi que des permissions liées aux rôles. Cette contrainte s'appuie sur la vue normative des méta-modèles tels que Moise+ [21], que nous avons analysé plus haut. Cette contrainte est dépendante de la dimension d'organisation du modèle.

La cinquième contrainte est aussi en relation avec cette même dimension. Elle indique que la caractéristique de groupe de holons ne peut être présente au sein d'une configuration que si les caractéristiques holon et rôle holonique sont présentes également. En effet, un groupe d'holons ne peut exister sans holons.

La sixième contrainte quant à elle, a été introduite en analysant la validité des dimensions d'environnement et d'interactions. Elle indique que pour la mise en place de mécanismes d'interactions indirectes au sein de l'environnement, il faut obligatoirement sélectionner la caractéristique *environment_ media* d'environnement.

De façon générale, une configuration ne peut être acceptée par notre modèle que si elle satisfait l'ensemble de toutes les contraintes qui mettent en jeu les caractéristiques sélectionnés pour la satisfaction d'une configuration relative au besoin d'utilisateur.

$$Norm \Rightarrow RoleObligation \vee RolePermission$$
$$English\_auction \vee Dutch\_auction \Rightarrow Auction\_evaluation\_Strategy \wedge Bidding\_Strategy$$
$$Mono\_Plan \vee Mono\_Goal \vee Multi\_Plan \vee Multi\_Goal \Rightarrow Belief$$
$$Reasoning\_component\_basic \Rightarrow Deliberative$$
$$HolonicGroup \Rightarrow Holon \wedge HolonicRole$$
$$Mediation\_Mechanism \Rightarrow environment\_media$$
$$HolonicRole \Rightarrow Holon \wedge HolonicGroup$$
$$Holon \Rightarrow HolonicGroup \wedge HolonicRole$$
$$follow\_through \vee trigger\_based \Rightarrow Stigmergy\_Mechanism$$
$$Mediation\_Mechanism \Rightarrow Physical\_Environment$$

FIGURE 5.7 – Un extrait des contraintes du modèle de la ligne de produits multi-agents

5.7 Conclusion

L'analyse des différentes caractéristiques d'un SMA indépendamment d'un domaine d'application nous a permis de réifier les principaux concepts communs et variables des SMA. Ce qui nous a également permis de construire des modèles de caractéristiques pour chacune des dimensions de l'approche Voyelles [28, 29].

Nous avons en résumé 1) procédé à une délimitation du domaine, 2) analysé les points communs et variables des SMA à partir des différents concepts issus des méta-modèles existants, ontologies et connaissances multi-agents ; 3) proposé une représentation générique réutilisable des points communs et variables construits sur la base d'une analyse sémantique ; et 4) fait un pas vers un modèle générique pour chacune des caractéristiques voyelles Agent, Environnement, Interaction et Organisation.

Ainsi, chaque agent peut être décrit par les modules obligatoires qui ont été représentés par 1) son architecture (délibérative, réactive, hybride) ; 2) sa perception (active ou passive) ; et 3) les actions qu'il peut réaliser.

L'environnement quant à lui peut être décrit par son module de déploiement pour les environnements artificiels, ainsi que par la possibilité d'être physique (représentation du monde réel). En plus du fait d'inclure un ensemble de ressources accessibles et consommables par les agents. Comme option, l'environnement peut donc être vu entant qu'espace de travail (workspace) comportant des composants génériques appelés artefacts. Ces derniers jouent un rôle primordial lors d'interactions indirectes entre les agents, afin de leur servir entre autre de moyen

de coordination, ainsi que de média d'interaction.

Les interactions s'articulant principalement autour des rôles d'interactions pouvant être décrits par la typologie de paramètres d'entrée. Les principales caractéristiques sont organisées en fonction de la nature des interactions, qui peuvent être directes par envoi de messages, ou indirectes via l'environnement.

Pour finir, l'organisation du SMA peut être unifiée grâce aux aspects structurels, fonctionnels et normatifs. Le concept de *Role* est considéré comme un élément minimal et central de l'organisation. Les caractéristiques optionnelles de l'organisation comprennent les *Groupes* d'agents, ainsi que des aspects instaurant des règles comportementales. Nous avons également considéré des organisations centralisées et décentralisées, afin de supporter ce type de variabilité au sein de notre modèle de caractéristiques réutilisable et indépendant du domaine d'application.

L'ensemble des différents modèles des caractéristiques réalisés pour chacune des voyelles (A.I.E.O), représente une spécification de la ligne de produits SMA, indépendamment du domaine d'application. Ces caractéristiques génériques sont directement associées aux implémentations génériques réutilisables que nous avons présentées dans le chapitre précédent.

Dans le chapitre suivant, nous allons passer à l'évaluation de notre approche montrant entre autres comment le modèle de caractéristique que nous proposons et qui a été détaillé dans ce chapitre, ainsi que les implémentations génériques proposées par notre approche, peuvent être réutilisés par différents cas d'application, et par conséquent faciliter le développement d'applications multi-agents similaires.

Cinquième partie

Validation expérimentale

Chapitre 6

Application et évaluation de notre approche MAS-PL

6.1 Introduction

L'ensemble des étapes proposées dans notre approche MAS-PL a pour principal objectif de faciliter le développement de familles de systèmes multi-agents. L'originalité de notre approche est qu'elle se base sur un modèle de caractéristique (resp. implémentation) à deux niveaux de variabilité. Le premier niveau du modèle qui a été présenté en détail dans le chapitre précédent, est réutilisable indépendamment du domaine d'application. Il en est de même pour les implémentations réutilisables associées aux caractéristiques de ce même niveau.

Dans ce chapitre nous évaluons l'approche que nous proposons. Notre démarche d'évaluation s'articule autour de trois types d'évaluation.

En premier lieu, nous avons évalué la faisabilité de notre approche par le biais de trois différents cas d'application comprenant (1) La vente de livre (book trading), (2) La gestion d'un emploi du temps (Timetable) et (3) Le multi-agent contest. Cette évaluation, met en avant la possibilité de réutiliser le niveau générique du modèle de caractéristiques ainsi que son implémentation afin de faciliter la conception et la mise en œuvre de différentes variantes spécifiques aux trois domaines d'application.

Nous allons montrer pour chacun des cas d'applications, qu'il est possible d'obtenir la première version du modèle spécifique de caractéristiques. Celui-ci, sera relié au second niveau de variabilité proposé par notre approche. Nous, allons également montrer, qu'il est possible de raffiner le modèle générique de caractéristiques associé au premier niveau variabilité, pour enrichir la ligne de produits. Ce qui a pour but, de mettre en avant l'aspect incrémental de notre approche. Nous avons également évalué la faisabilité de notre approche relativement à la possibilité de réutiliser les artefacts d'implémentation durant le développement des artefacts

spécifiques au domaine d'application.

En second lieu, nous avons procédé à une évaluation prospective. Celle-ci visant à analyser et mettre en avant les différents résultats obtenus par des groupes d'étudiants utilisant une approche MAS-PL, dont les étapes ont été générées par l'approche Meduse [159], reprenant ainsi les principales phases de notre approche, sans utiliser comme point de départ le modèle générique de caractéristiques, ou encore les artefacts proposés par notre approche.

En troisième et dernier lieu, nous avons évalué la validité de notre approche par rapport à la dérivation des variantes, de leurs conformité à répondre aux besoins, et aussi de leurs respect de l'ensemble des contraintes indépendantes et dépendantes du domaine d'application. L'ensemble des variantes en question, a été dérivé sur la base des configurations possibles qui permettent de répondre aux besoins des utilisateurs. Les variantes ont été obtenues par dérivation avant d'être déployées.

Pour les variantes en question, nous avons évalué des aspects relatifs à la réutilisation. Nous présenterons l'ensemble des métriques que nous avons choisies, et qui permettent d'évaluer entre autres le pourcentage de code source réutilisé au niveau des variantes pour lesquelles des simulations ont été réalisées afin de garantir que ces variantes obtenues par dérivation, répondaient bien aux besoins.

Nous présentons aussi les avantages et inconvénients de notre approche.

6.2 Évaluation de la faisabilité de notre approche

Dans cette section, nous évaluons la faisabilité de notre approche par le biais de trois cas d'études, soit : *la gestion de l'emploi du temps*, *la vente de livres* et enfin *le multi-agent contest*. Pour chaque cas d'étude, nous proposons de démarrer du cas le plus simple du domaine d'application concerné. Par le cas le plus simple, nous faisons référence aux variantes multi-agent qui présentent les fonctionnalités minimales requises au niveau de la ligne de produits multi-agents. Sur la base de la première variante minimaliste, nous présenterons pour chacun des cas d'études, le diagramme de classe correspondant, ensuite nous illustrerons l'utilisation des caractéristiques (features) pour l'obtention de la première version du *FM spécifique*. Cette dernière étant obtenue par raffinement du *FM générique*. De plus, nous présenterons l'association (mapping) entre le modèle de caractéristiques spécifique obtenu avec les artefacts qui l'implémentent.

Nous donnerons aussi, des détails sur le processus incrémental de notre approche, grâce auquel il est possible de raffiner la première version du *FM spécifique*. Le but du raffinement, étant d'agrandir le nombre de variantes multi-agents supportées par le modèle, afin de répondre à de nouveaux besoins de la ligne de produits.

Nous donnerons également, des détails sur la manière de réutiliser les artefacts d'implémentation, relatifs à chaque caractéristique. Pour finir, nous allons montrer comment ajouter

les contraintes spécifiques au domaine d'application, et nécessaires afin d'être en mesure de dériver des variantes sur la base de configurations valides capables de répondre aux besoins d'utilisateurs, et capables d'être déployées.

6.2.1 Gestion de l'emploi du temps

Le benchmark de la gestion de l'emploi du temps (appelé TimeTable) a été proposé par Bernon et al. [160] afin d'étudier certains problèmes dans le domaine des SMA. Il représente un problème d'allocation de ressources complexe pour lequel la recherche de solutions doit être collective.

6.2.1.1 Cas simple : aucune contrainte

Afin d'appliquer notre approche, il faut toujours commencer par la variante la plus simple de la ligne de produits multi-agents. Pour cette première variante nous ne prenons en compte aucune contrainte et considérons uniquement que les enseignants et étudiants doivent se mettre d'accord pour la mise en place de l'emploi du temps. Nous donnons plus de détails dans ce qui suit.

— **Première version du diagramme de classes :**
Le diagramme de classes correspondant est représenté dans la la figure 6.1 qui résume l'implémentation de cette variante. Il comprend d'une part un ensemble de classes relatives à l'ontologie du domaine et d'autre part un ensemble de classes représentant les agents et leurs comportements.
Cette variante minimaliste ne considère aucune contrainte sur les ressources. Par exemple, elle considère que les enseignants et les groupes d'étudiants trouvent une solution pour les cours. Les classes sont ensuite affectées à ces cours avec tout le matériel nécessaire (Tableau noir, vidéo projecteur ...) et les contraintes (taille de la salle ...).
Dans notre modèle, chaque utilisateur est doté d'un agent assistant dont le comportement est guidé par le protocole contrat Net (CNP).

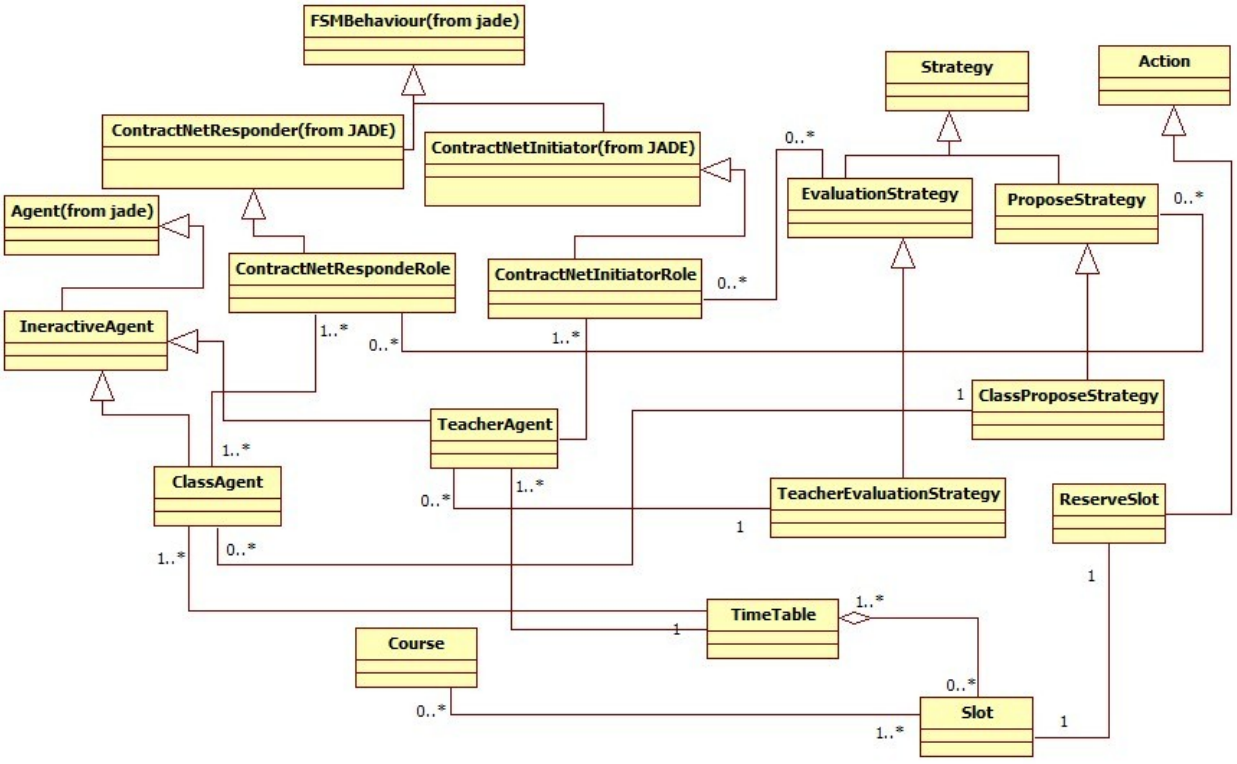


FIGURE 6.1 – Diagramme de classe de gestion de l'emploi du temps sans contraintes

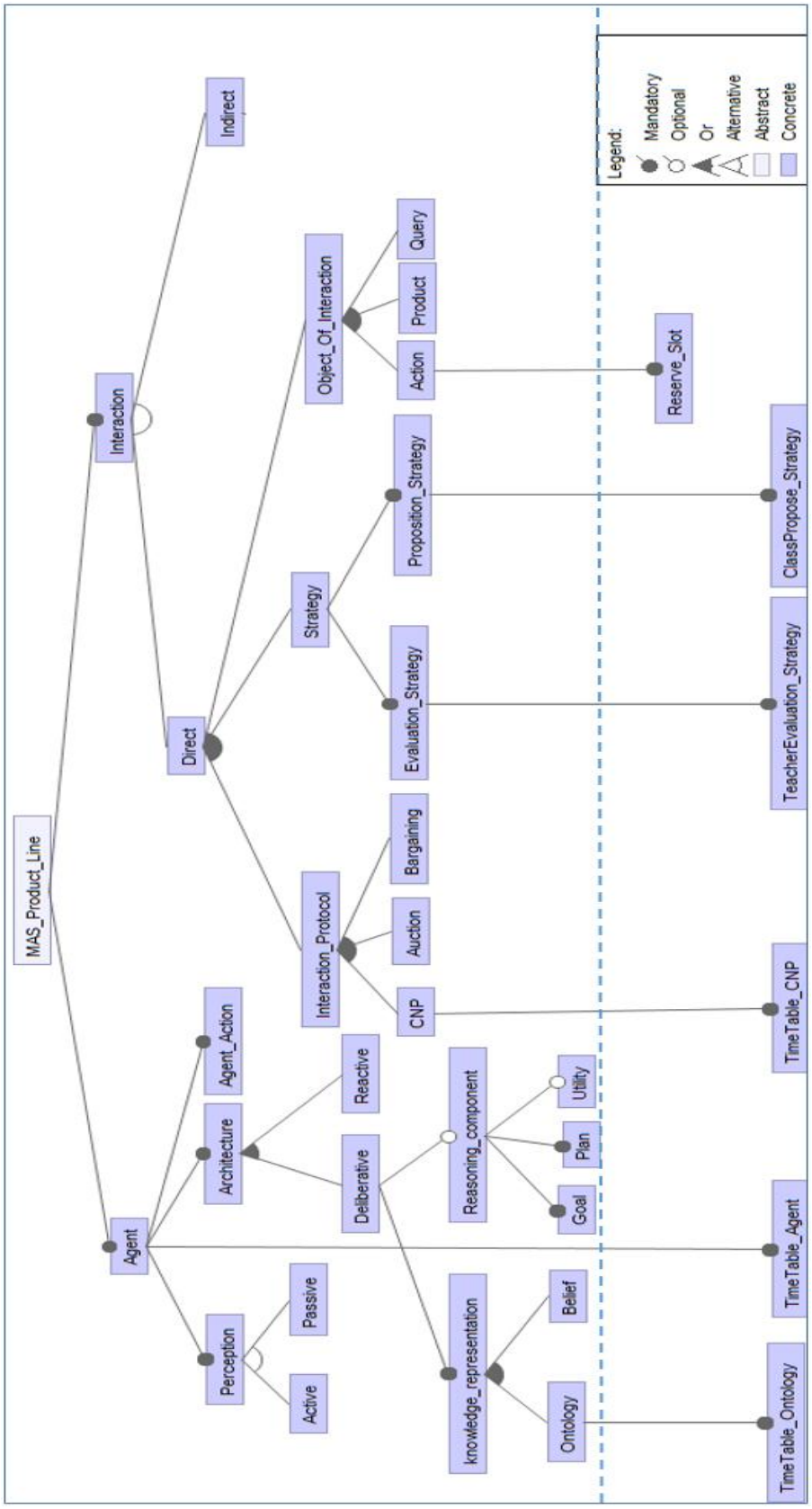


FIGURE 6.2 – Modèle de caractéristiques spécifique de gestion de l'emploi du temps sans contraintes

— **Première version du modèle de caractéristiques FM (Feature Model) spécifique à la gestion de l'emploi du temps :**
Une fois la première version du diagramme de classe obtenue, il est nécessaire de raffiner le modèle générique de caractéristiques afin qu'il devienne spécifique. Pour ce faire nous introduisons les caractéristiques correspondantes à l'implémentation. La première version du modèle spécifique de caractéristiques est représentée dans la figure 6.2. Les caractéristiques génériques et spécifiques qui raffinent le modèle sont séparées par une ligne horizontale. Les nouvelles caractéristiques introduites font référence d'une part à l'ensemble des agents utilisés dans cette variante, leurs stratégies, le protocole utilisé ainsi que l'ontologie du domaine d'application. L'ajout des caractéristiques a été réalisé sur la base de l'analyse du diagramme de classes en regroupant ces artefacts en catégories relatives aux fonctionnalités de la gestion de l'emploi du temps.

— **Association (mapping) entre la nouvelle version du modèle de caractéristiques FM et les artefacts implémentant l'emploi du temps sans contraintes :**
Comme cité ci-dessus, nous avons regroupé les classes qui implémentent cette première variante de l'emploi du temps en groupes. Par exemple les classes *Course*, *Slot* et *TimeTable* implémentent la caractéristique de l'ontologie du domaine *TimeTable_Ontology*, qui raffine la caractéristique générique *Ontology*. L'association entre la première version du modèle spécifique de l'emploi du temps et les artefacts qui l'implémentent est représentée dans la figure 6.3.

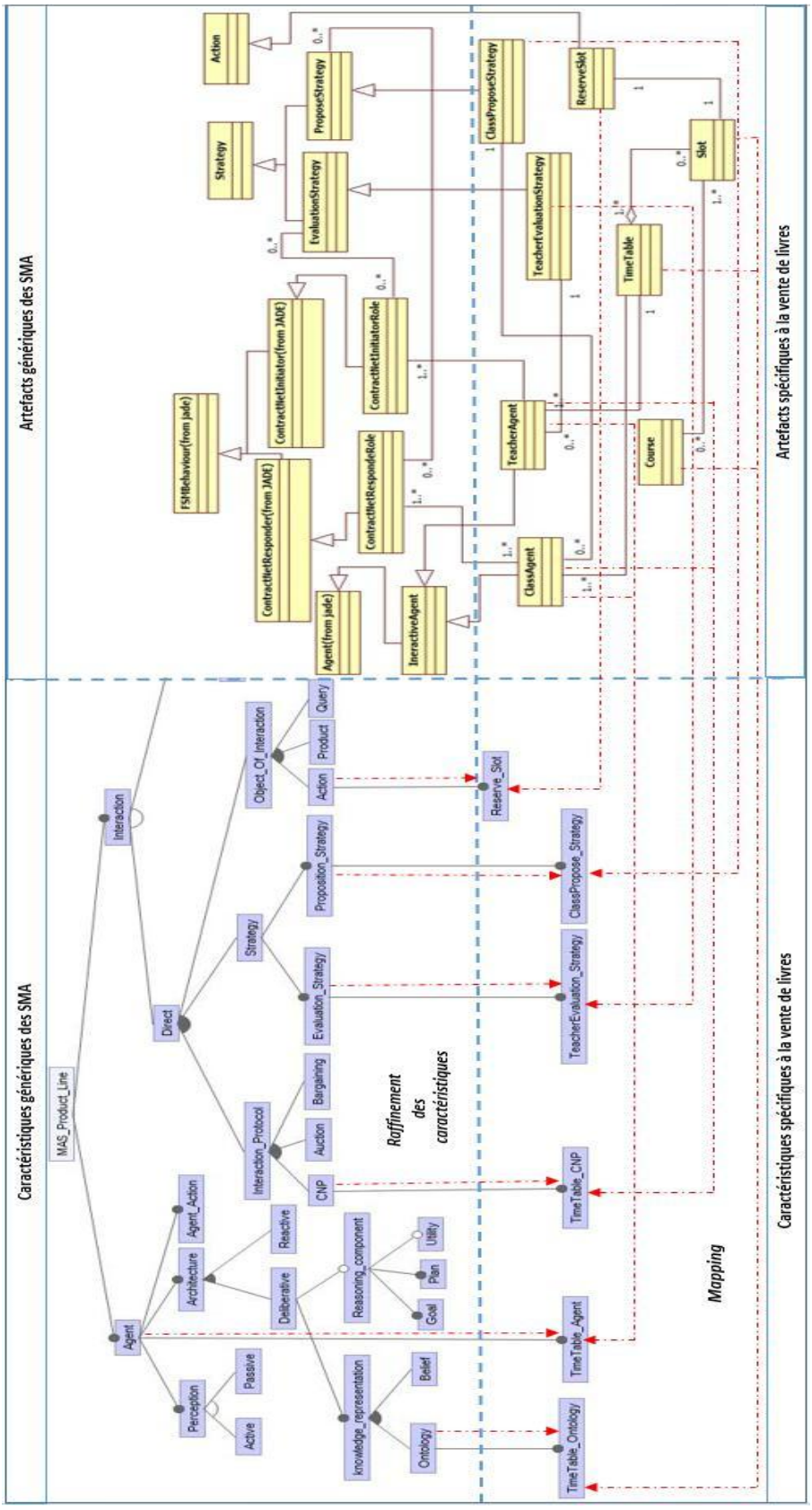


FIGURE 6.3 – Association (mapping) entre le FM spécifique de gestion de l'emploi du temps sans contraintes et ses artefacts d'implémentation

6.2.1.2 Raffinement avec passage aux contraintes sur les salles de classes

Le problème de l'emploi du temps peut avoir plusieurs variantes. Ceci est réalisé en augmentant la complexité du système grâce à l'inclusion de contraintes, ou bien en ajoutant d'autres fonctionnalités ou encore en jouant sur la cardinalité d'agents lors des interactions. Par exemple, il est possible de changer le nombre d'agents Enseignants dans notre système (Version 1 :1 Enseignant et n Classes ; Version 2 : m Enseignants et n Classes). Ainsi, les agents Classes peuvent recevoir plusieurs propositions pour le même cours. Ensuite, les agents Classes peuvent définir une stratégie pour sélectionner l'un d'entre eux à accepter.

Pour cet exemple, nous allons raffiner le modèle par l'inclusion de contraintes. En effet, si on considère le problème associé à ce système et qui consiste à trouver un emploi du temps cohérent en tenant compte des points suivants : (i) une liste de m enseignants, (ii) une liste de n classes (groupes d'élèves) et, si nécessaire, (iii) une liste des p salles. Compte tenu de ces nouvelles considérations, plusieurs variantes peuvent être déclinées. En effet, l'introduction de la variabilité concernant le nombre d'agents entraîne le changement du protocole d'interaction. En plus, la prise en charge des salles génère aussi d'autres variantes en fonction de la façon dont les ressources sont gérées. Afin de raffiner notre modèle pour supporter une nouvelle variante permettant de prendre en considération des contraintes sur les salles de classes, nous avons tout d'abord analysé le nouveau diagramme de classe obtenu.

— **Seconde version du diagramme de classes :**
La nouvelle version du diagramme de classes supportant l'ajout de contraintes sur les ressources telles que les salles est représentée dans la Figure 6.4. Pour ce faire, il faut entre autre ajouter un agent pour gérer toutes les salles ou associer à chaque salle un gestionnaire.

— **Seconde version du modèle de caractéristiques FM :**
La première version du FM spécifique est raffinée en fonction du dernier diagramme de classe obtenu. En effet, afin de supporter les contraintes sur les classes, il est nécessaire d'ajouter les caractéristiques : *Room_ Concept* et *Room_ Constraint_ Agent*. Ces deux nouvelles caractéristiques étant optionnelles et reliées par deux contraintes car la présence de l'une d'entre elles au sein d'une configuration de variantes implique la présence de la seconde. Le FM obtenu est représenté dans la figure 6.5.

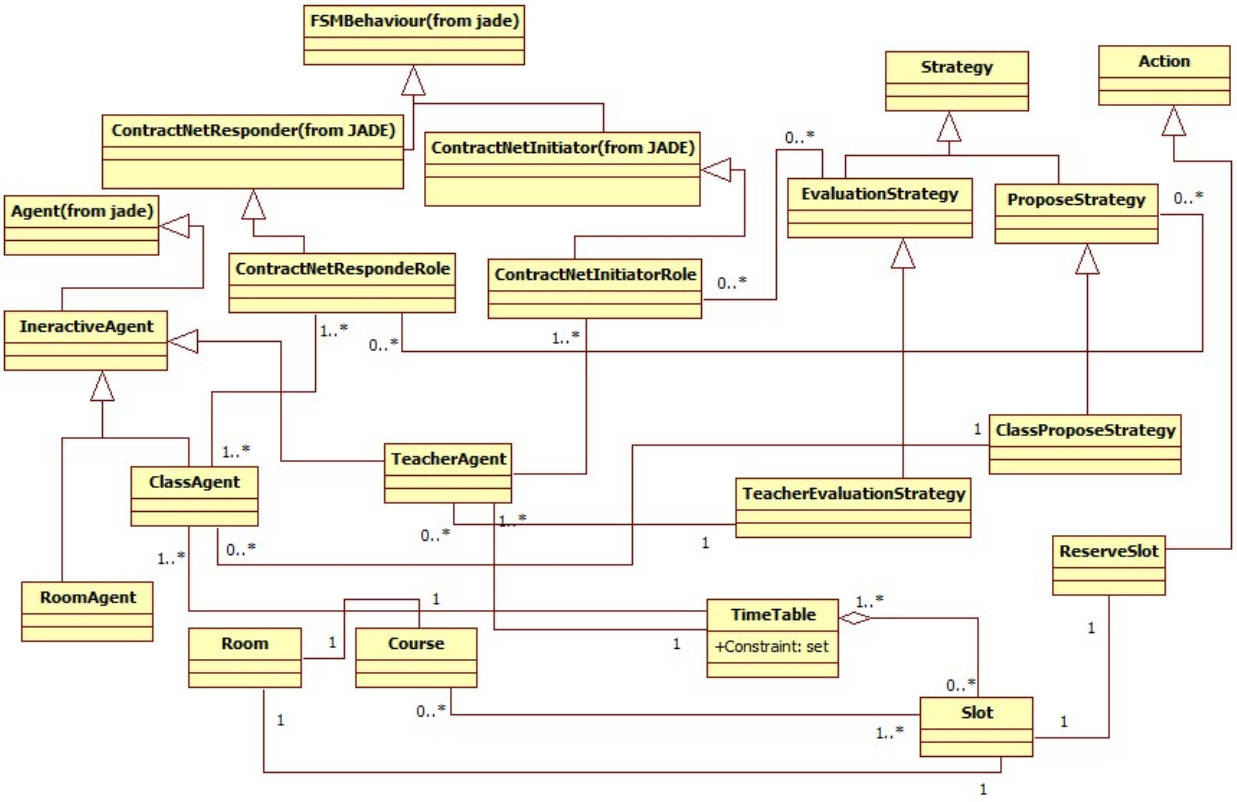


FIGURE 6.4 – Seconde version du diagramme de classe de l'emploi du temps avec contraintes sur les salles

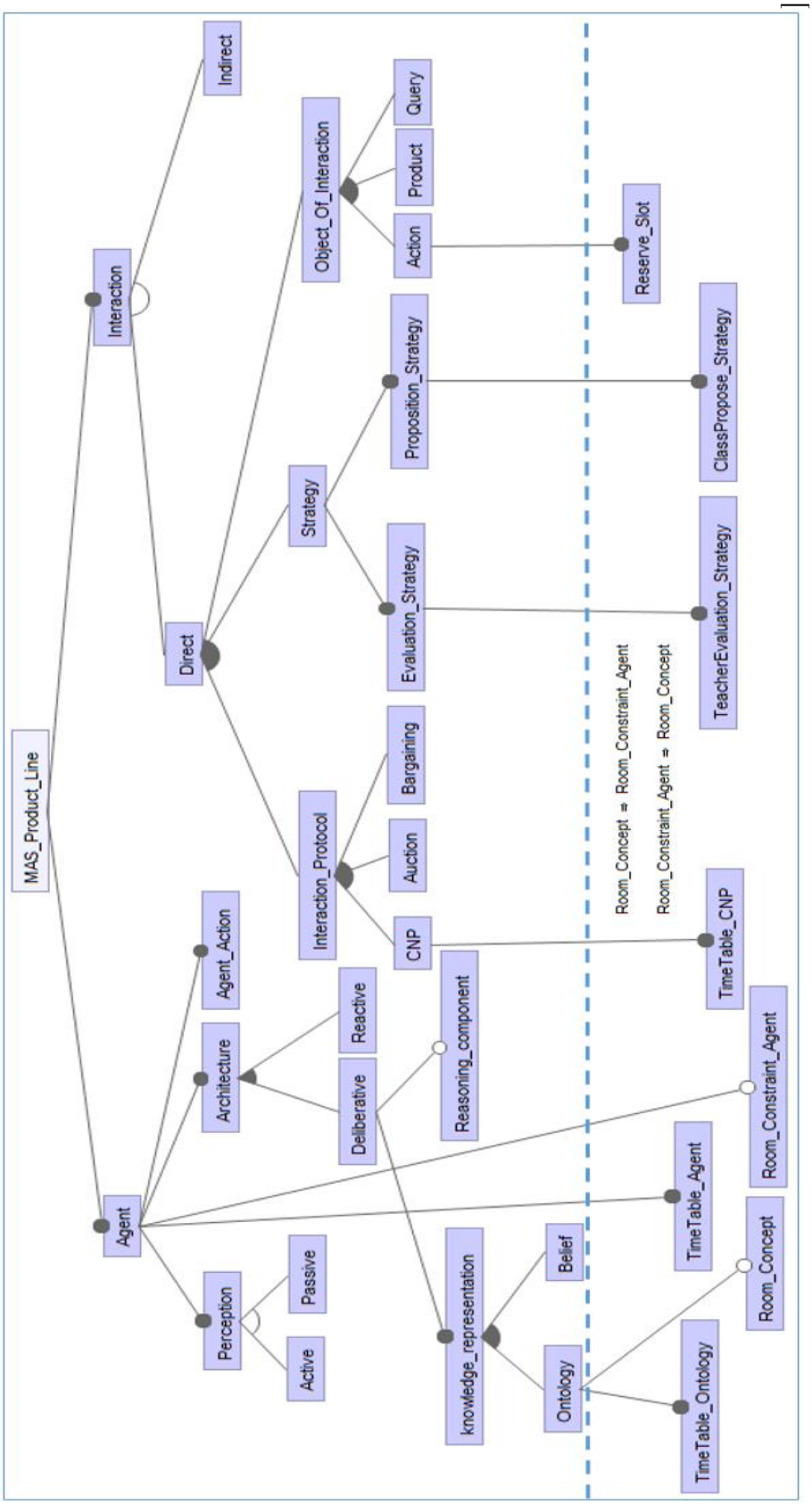


FIGURE 6.5 – Seconde version du feature modèle spécifique à de l'emploi du temps avec contraintes sur les salles

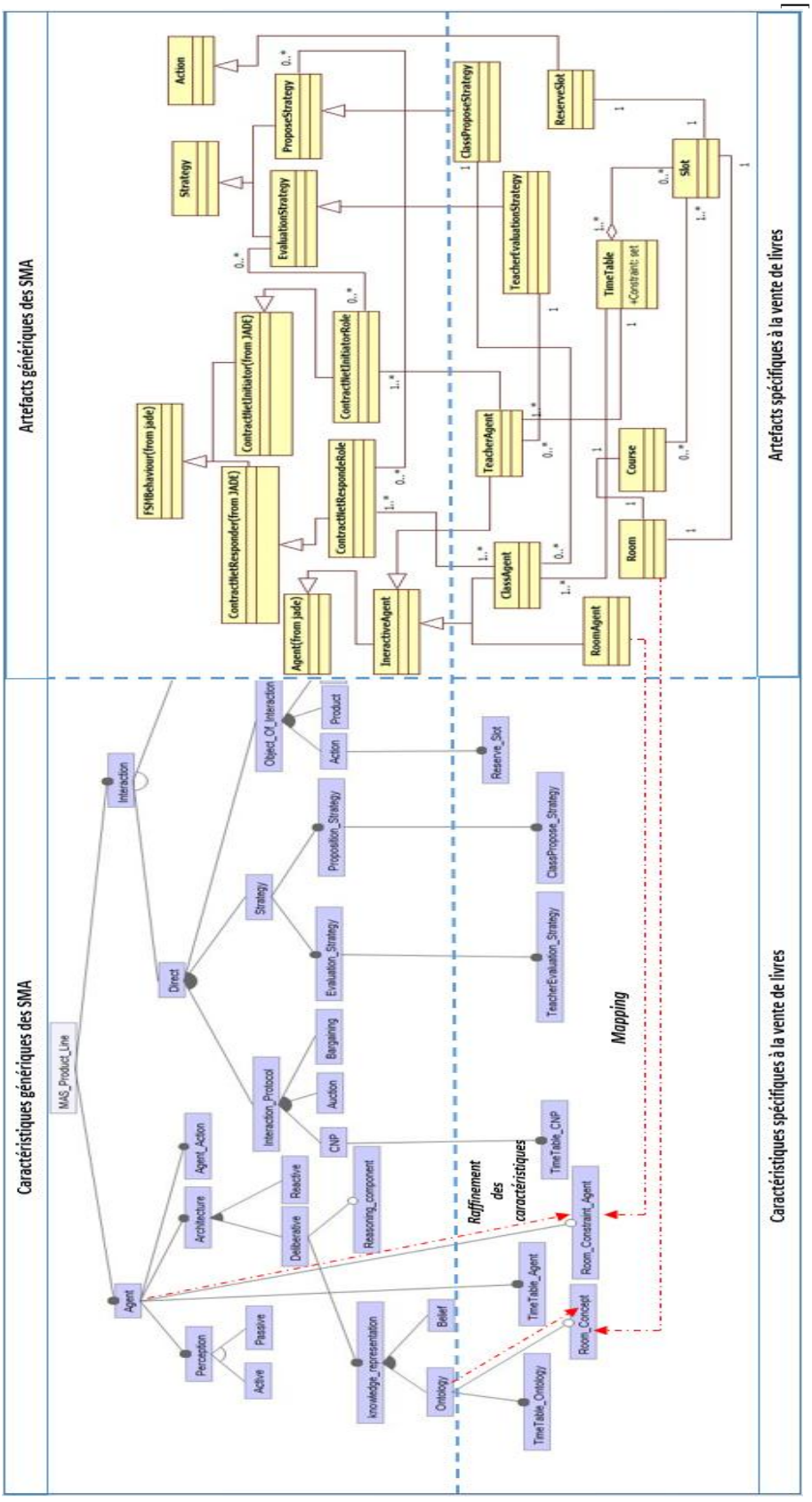


FIGURE 6.6 – Mapping du feature modèle spécifique et les artefacts qui implémentent l'emploi du temps avec contraintes

— **Association (mapping) entre la nouvelle version du modèle de caractéristiques FM et les artefacts implémentant l'emploi du temps avec contraintes :** Le mapping entre les nouveaux éléments introduits au sein du diagramme de classes (artefacts d'implémentation) et la nouvelle version du FM spécifique est représenté dans la figure 6.6.

6.2.1.3 Ajout des contraintes de validité des configurations et des variantes

Avant de dériver les variantes de la gestion de l'emploi du temps, il est nécessaire d'intégrer au sein du modèle les contraintes spécifiques au domaine d'application. Un exemple des contraintes est donné dans la figure 6.7.

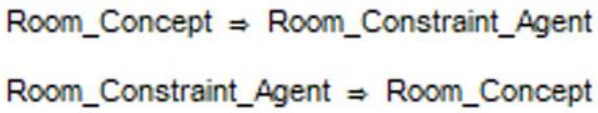


FIGURE 6.7 – Exemple de contraintes spécifiques à la gestion d'un emploi du temps

6.2.2 Vente de livres (Book trading)

Le benchmark suivant concerne la vente de livres (book trading). Il s'agit de simulations où les agents Acheteurs désirent acquérir un ou plusieurs livres auprès d'agents Vendeurs. Nos scénarios sont cependant différents pour chacune des variantes de la ligne de systèmes multi-agents que nous considérons. Cette variabilité prend ainsi en considération des interactions de cardinalité variante allant de un à plusieurs agents vendeurs/acheteurs.

Le cas d'interaction le plus simple pouvant avoir lieu entre un agent acheteur et un agent vendeur passe par la négociation. Celle-ci pouvant présenter une variabilité relative à la mise en place d'une stratégie à prix fixe ou à prix variable par exemple. Dans le premier type de variante l'agent acheteur n'aurait donc pas la possibilité de négocier le prix du livre. Ceci peut être spécifié et implémenté par le protocole FIPA *Contract Net* étant à la base de quasiment tous les travaux sur les protocoles de négociation. Le Contract Net Protocol permet une négociation qui consiste en un appel d'offres de la part d'un agent appelé initiateur ou manager. Différentes offres sont alors proposées à l'initiateur par des participants ou contractants, mais seulement l'une d'elles est choisie par l'initiateur sans autre formulation si aucune offre ne convient. Dans notre première variante, nous ne considérons qu'un seul participant. Par conséquent le comportement de l'agent acheteur serait celui définit par le rôle initiateur du CNP, tandis

que le comportement de l'agent vendeur serait celui définit par le rôle participant du CNP. Dans le second type de variante, l'agent acheteur aurait la possibilité de négocier le prix du livre. Cette variabilité apparaît également au niveau des variantes comportant une cardinalité supérieure d'agents acheteurs et vendeurs. Par conséquent ces caractéristiques peuvent être introduites indépendamment de la cardinalité sachant que l'on peut offrir à un Agent acheteur la possibilité de négocier le prix du livre qu'il soit ou pas le seul agent inclus dans le scénario. Plus encore, il est possible d'utiliser les différentes variantes du contract net, ce qui représente ainsi un autre niveau de variabilité.

Les interactions de cardinalité supérieure à 1 peuvent faire intervenir des négociations plus complexes. Celles-ci pouvant faire appel à des actions décisionnelles d'agents différentes. Ces actions décisionnelles vont être représentées par des comportements plus ou moins similaires faisant intervenir différentes stratégies tant pour les agents acheteurs que pour les agents vendeurs.

Par ailleurs, lors d'interactions à cardinalité multiple, l'autre niveau de variabilité observé est celui du protocole d'interaction réutilisé au sein de la variante. En effet, il est possible de considérer la négociation entre les agents acheteurs et vendeurs autrement. L'interaction commencerait ainsi à un niveau où plusieurs offres seront échangées, où des contre-propositions seraient formulées et des concessions seraient faites par les différentes parties impliquées. Une variante possible suivrait le protocole d'enchères. Les enchères font en effet partie de ce type de négociations sous sa forme la plus simple vu qu'aucune concession n'est faite par l'une des parties et aussi parce que le résultat n'est satisfaisant que pour l'agent vendeur et l'acheteur ayant gagné l'enchère. Dans notre cas, nous ne considérons pas des variantes plus complexes telles que celles supportant les négociations multi-niveaux et les négociations par argumentation.

Étant donné que pour cet exemple les agents sont destinés à s'exécuter sur la plateforme de JADE, les classes réutilisées et implémentées par JADE sont aussi présentées et réutilisées.

Bien que l'exemple du *BookTrading* proposé par JADE soit intéressant, il n'a pas été repris dans notre exemple de ligne de produits multi-agents, car étant implémenté par des classes de comportements non réutilisables indépendamment du domaine de vente de livres, il ne s'avère pas pertinent pour notre problématique de réutilisation. En effet, l'exemple proposé est utilisé en tant que tutoriel dont l'objectif est de manipuler les classes de comportements Behaviour dans laquelle le développeur implémente seul au niveau même de l'agent les comportements et conditions d'arrêt du comportement. Par exemple, bien que JADE fournisse une librairie implémentant certains protocoles d'interactions, contrairement aux variantes que nous proposons, ces dernières ne sont pas réutilisées dans l'exemple de vente de livre de JADE.

6.2.2.1 Cas simple (interactions 1 :1)

Les variantes les plus simples considérées dans notre ligne multi-agents sont représentées par des interactions à cardinalité 1 à 1 soit un agent Acheteur et un agent Vendeur. Ce type de variantes comme nous allons le montrer plus tard présenteront des configurations minimales. Ces variantes sont implémentées par la réutilisation des deux rôles d'initiateur et de participant du contract net. Ces variantes vont cependant différer par rapport à la *Stratégie de proposition* du rôle *ContractNetParticipant*. Nous allons montrer comment grâce à notre approche, il est plus simple d'effectuer un changement de stratégie et de générer automatiquement la variante multi-agent souhaitée pour la vente de livres avec une stratégie donnée ou encore une autre. Dans notre exemple, nous proposons deux stratégies différentes afin d'illustrer notre approche.

— **Stratégie de proposition à prix fixe :**
Les interactions réalisées entre les deux agents *Acheteur* et *Vendeur* vont suivre le protocole d'interaction du Contract-Net. On retrouve ainsi les deux rôles Initiator et Participant tels qu'ils ont étaient définis par FIPA.
L'agent Acheteur jouera le rôle de l'initiateur du contract net, tandis que l'agent Vendeur jouera le rôle du participant.
Les deux diagrammes d'états correspondants sont représentés plus haut dans les figures 6.8 et 6.9 (voir chapitre 5).
Cette première variante multi-agent de vente de livres permet de réaliser une vente avec une *Stratégie de proposition* à prix fixe pour le rôle tenu par l'agent *Vendeur*.

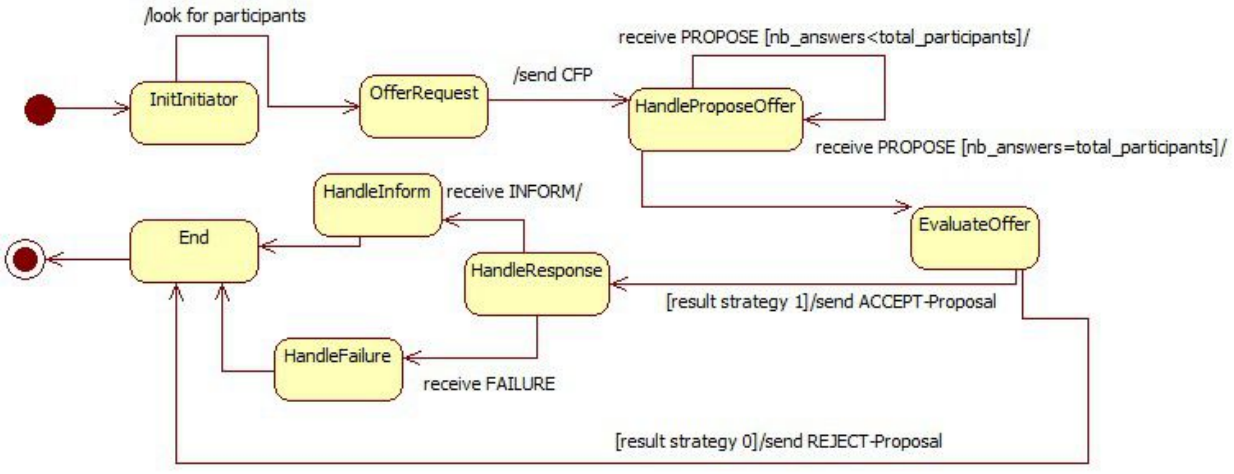
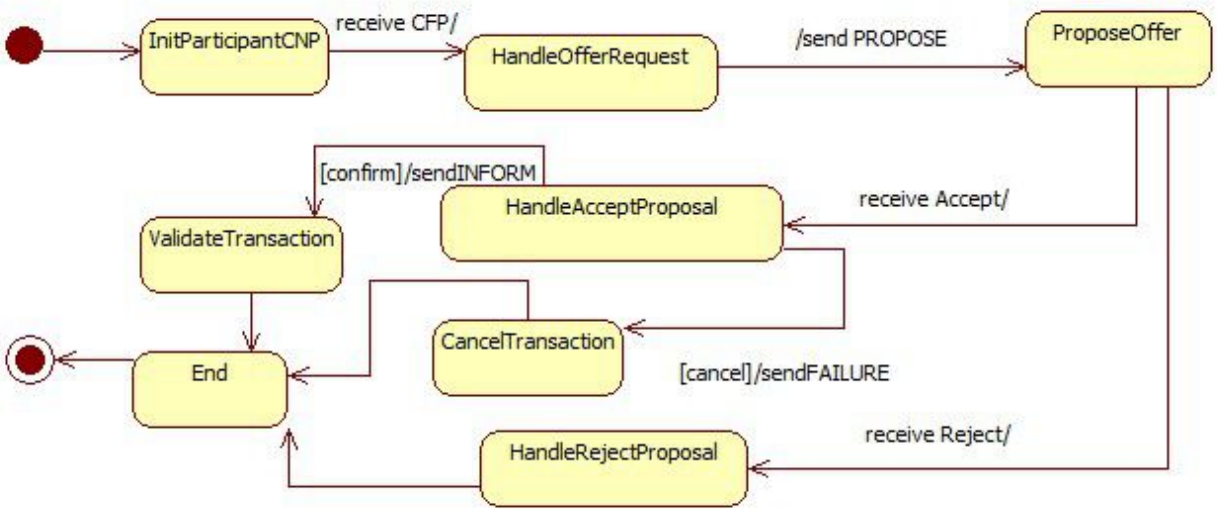


FIGURE 6.8 – Diagramme d'état du role Initiateur du protocole FIPA Contract Net



]

FIGURE 6.9 – Diagramme d'état du rôle Participant du protocole FIPA du Contract Net

— **Première version du diagramme de classe :**
Le diagramme de classe obtenu est représenté dans la Figure6.10. Il est composé d'une part des classes spécifiques du domaine de vente de livres, et d'autre part des classes génériques que nous avons proposé de réutiliser au sein de notre approche afin de faciliter la génération de variantes.

— **Première version du modèle de caractéristiques spécifiques :**
Une fois le diagramme de la première version (variante la plus simple) réalisé, nous devons raffiner le FM générique afin d'obtenir des caractéristiques spécifiques. Ces dernières sont obtenus sur la base de la première version du diagramme de classe.
Le FM obtenu est représenté dans la figure 6.11, les deux niveaux de features sont séparés par une ligne. Chaque ajout de features spécifiques a été réalisé sur la base du diagramme de classe obtenu plus haut. En effet, chaque nouvelle classe spécifique du domaine devra correspondre à une caractéristique. Il est important de préciser qu'une caractéristique spécifique n'est pas forcément associée à une seule et unique classe mais peut être également représentée par un ensemble de classes servant à implémenter cette même caractéristique.

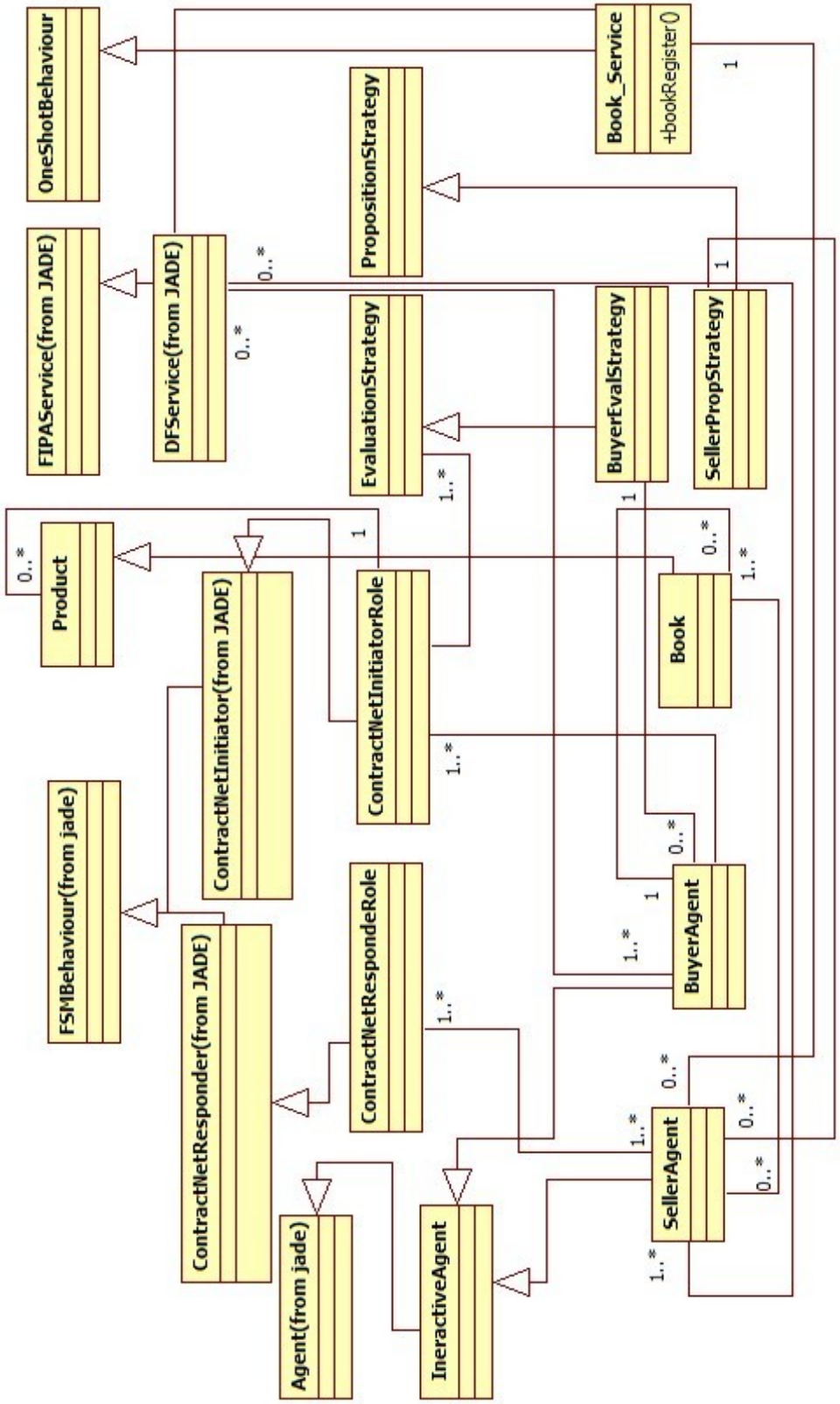


FIGURE 6.10 – Diagramme de classe de vente de livres avec interactions 1 :1 et prix de vente fixe

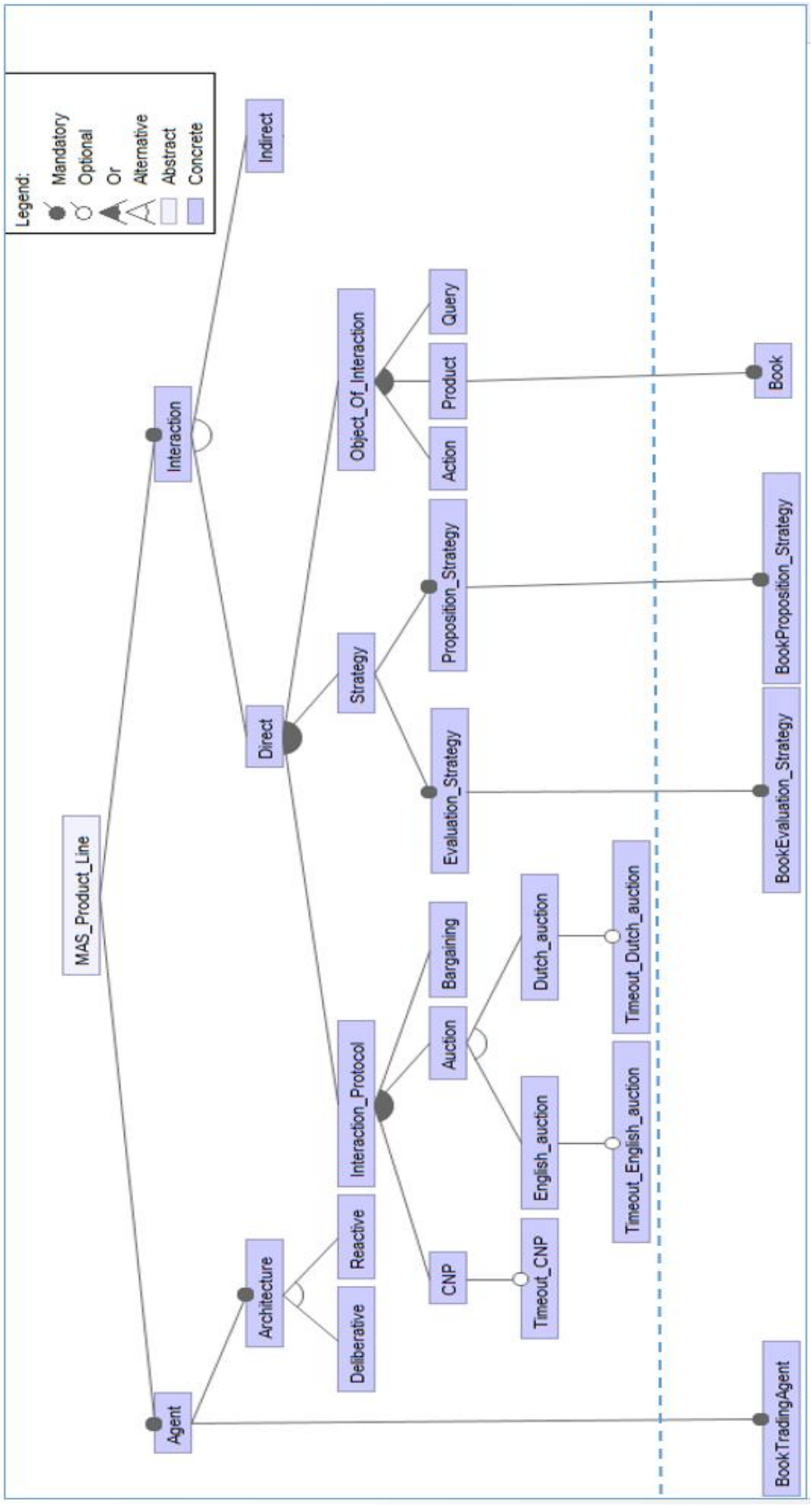


FIGURE 6.11 – Première version du modèle de caractéristiques spécifique à la vente de livres

— **Première association (mapping) entre le diagramme de classe et le FM spécifique :**
La Figure 6.12 représente l'association qui est faite entre chaque nouvelle classe (classes spécifiques à la vente de livres) et la feature qu'il faut introduire au niveau du FM générique. Notre approche est incrémentale (le FM peut être raffiné plusieurs fois), comme nous l'avons précisé lors de la présentation de ses activités. La première version du FM spécifique est obtenue en raffinant le FM générique. Dans notre exemple, les classes spécifiques *SellerAgent* et *BuyerAgent* héritent de la classe générique *interactiveAgent* qui hérite à son tour de la classe générique *Agent* fournie par JADE. Par conséquent les deux classes spécifiques précédentes raffinent la caractéristique *Agent* par le biais de la feature *BookTradingAgent*. Cette dernière va donc être implémentée par deux artefacts, représentant les classes d'agents de vente de livres respectives soit *SellerAgent* et *BuyerAgent* auxquelles elles sont associées.

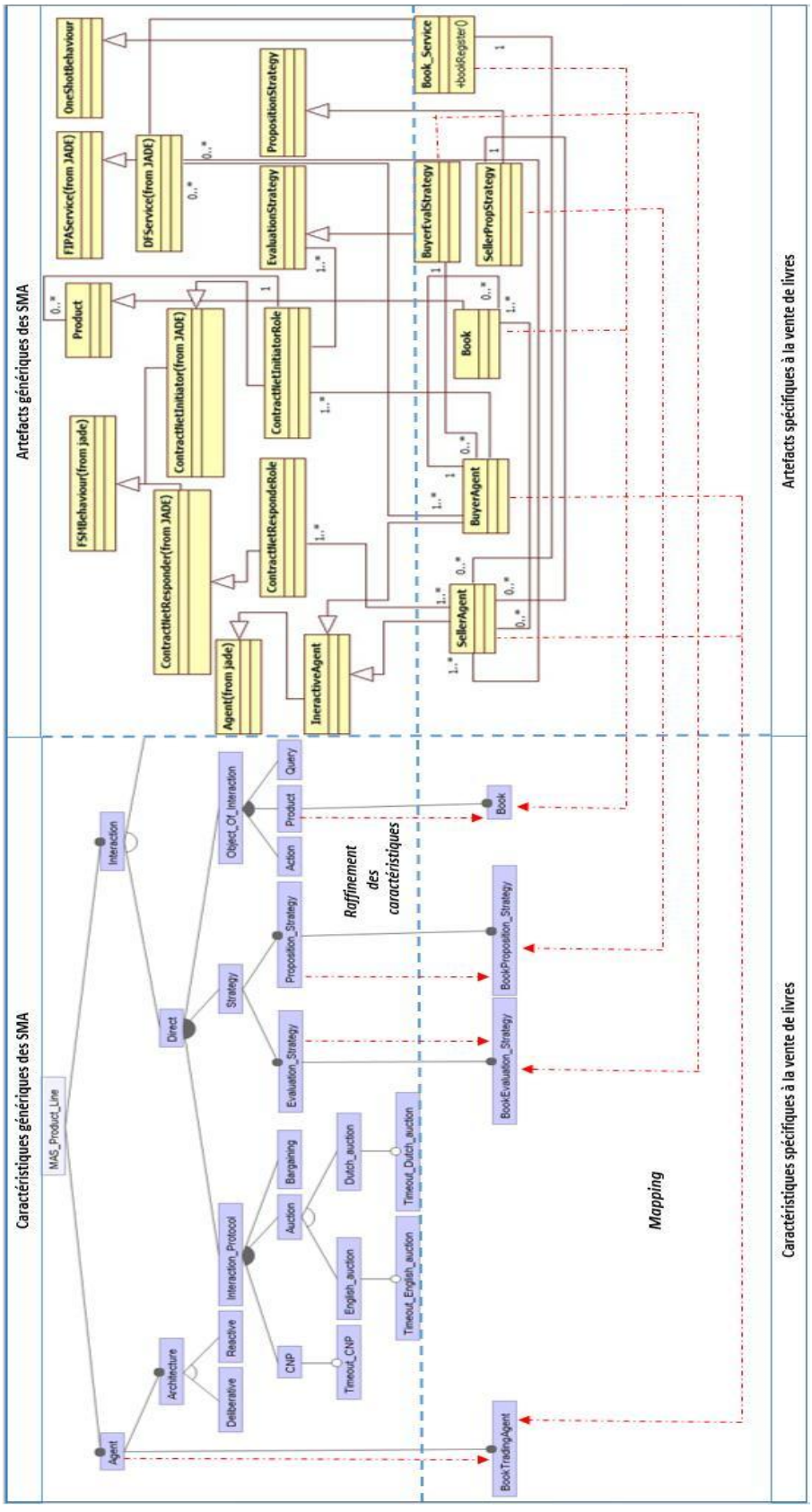


FIGURE 6.12 – Association (mapping) entre le FM spécifique à la vente de livres et le diagramme de classe qui l'implémente

Un autre exemple serait d'associer la classe spécifique *BuyerEvalStrategy* à une nouvelle feature spécifique *BookEvaluation_Strategy*. Etant donné que cette nouvelle classe hérite de la classe générique *EvaluationStrategy*, la nouvelle feature spécifique *BookEvaluation_Strategy* est représentée en tant que caractéristique fille de *EvaluationStrategy*.

Comme cette version du FM spécifique est la plus simple, l'ensemble des features spécifiques raffinant le FM générique seront obligatoires (mandatory).

— **Prix négociable :**

Dans cette variante l'agent acheteur a la possibilité de négocier le prix de vente si l'offre de l'agent vendeur ne lui convient pas. Ceci est réalisé en réutilisant les classes implémentant les deux rôles d'initiateur et de participant du bargaining. Dans cet exemple nous présentons l'exemple le plus simpliste du bargaining. Le comportement de l'agent vendeur devra être raffiné, afin de lui permettre d'évaluer des nouvelles demandes CFP. Quant au comportement de l'agent acheteur il sera raffiné en lui offrant la possibilité de négocier le prix proposé en faisant une autre demande avec un prix inférieur à celui proposé par l'agent vendeur. Ceci implique la réutilisation des features et artefacts du *Bargain protocol*.

— **Seconde version du diagramme de classe :**

Le raffinement du modèle se fait tout d'abord en analysant la variabilité au niveau du diagramme de classes. En effet, ce dernier est étendu avec les classes d'agents BargainSellerAgent et BargainBuyerpantAgent jouant les deux rôles définis. Relativement à ce processus de raffinement, nous obtenons la nouvelle version du diagramme de classe représentée dans la Figure 6.13.

Il est possible soit d'effectuer les modifications au sein des classes SellerAgent et BuyerAgent en paramétrant les nouveaux comportements spécifiques relatifs à cette variante, ou bien d'introduire de nouvelles classes à associer aux nouvelles features.

— **Raffinement du FM spécifique**

Afin de permettre au modèle de supporter cette nouvelle variante, il est nécessaire d'introduire de nouvelles caractéristiques relativement aux modifications réalisées au sein du diagramme de classe. Le FM raffiné pour la négociation 1 :1 (bargain) est représenté dans la Figure 6.14. La variabilité se situant au niveau des comportements des agents Seller et Buyer, les caractéristiques correspondantes ont été ajoutées en nécessitant d'introduire la caractéristiques *BookCNP* et *BargainCNP* afin que le modèle supporte cette variabilité.

De plus les caractéristiques relatives aux stratégies d'agents permettant de supporter la négociation de prix telles que la caractéristique *NegotiableBookPriceProp* ont également été introduites. Plus de détails concernant l'implémentation de chacune de ces nouvelles

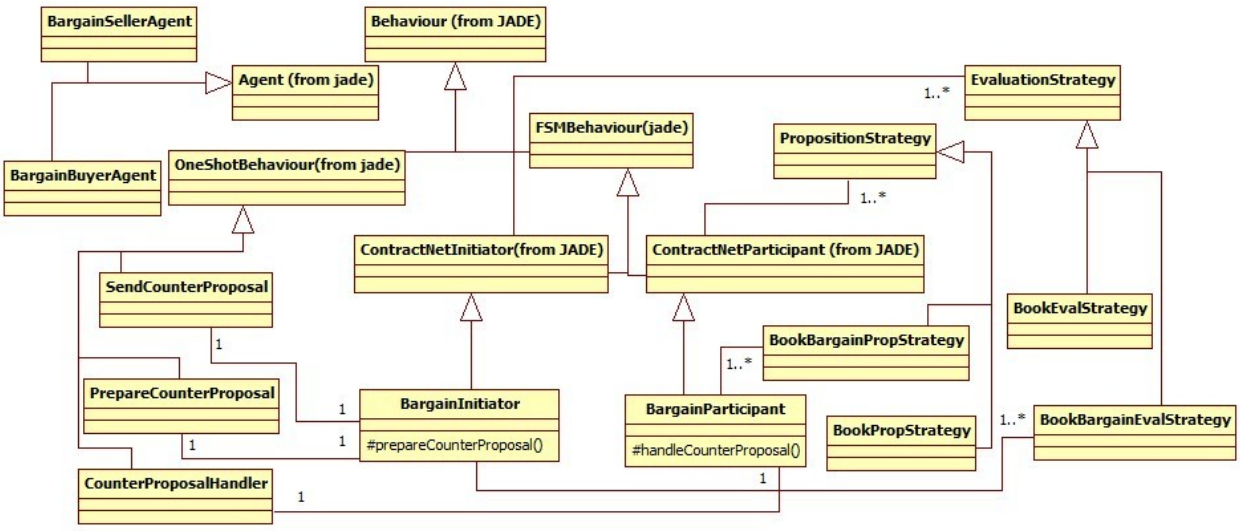


FIGURE 6.13 – Diagramme de classe raffiné pour permettre des négociations 1 :1

caractéristiques, sont donnés dans ce qui suit.

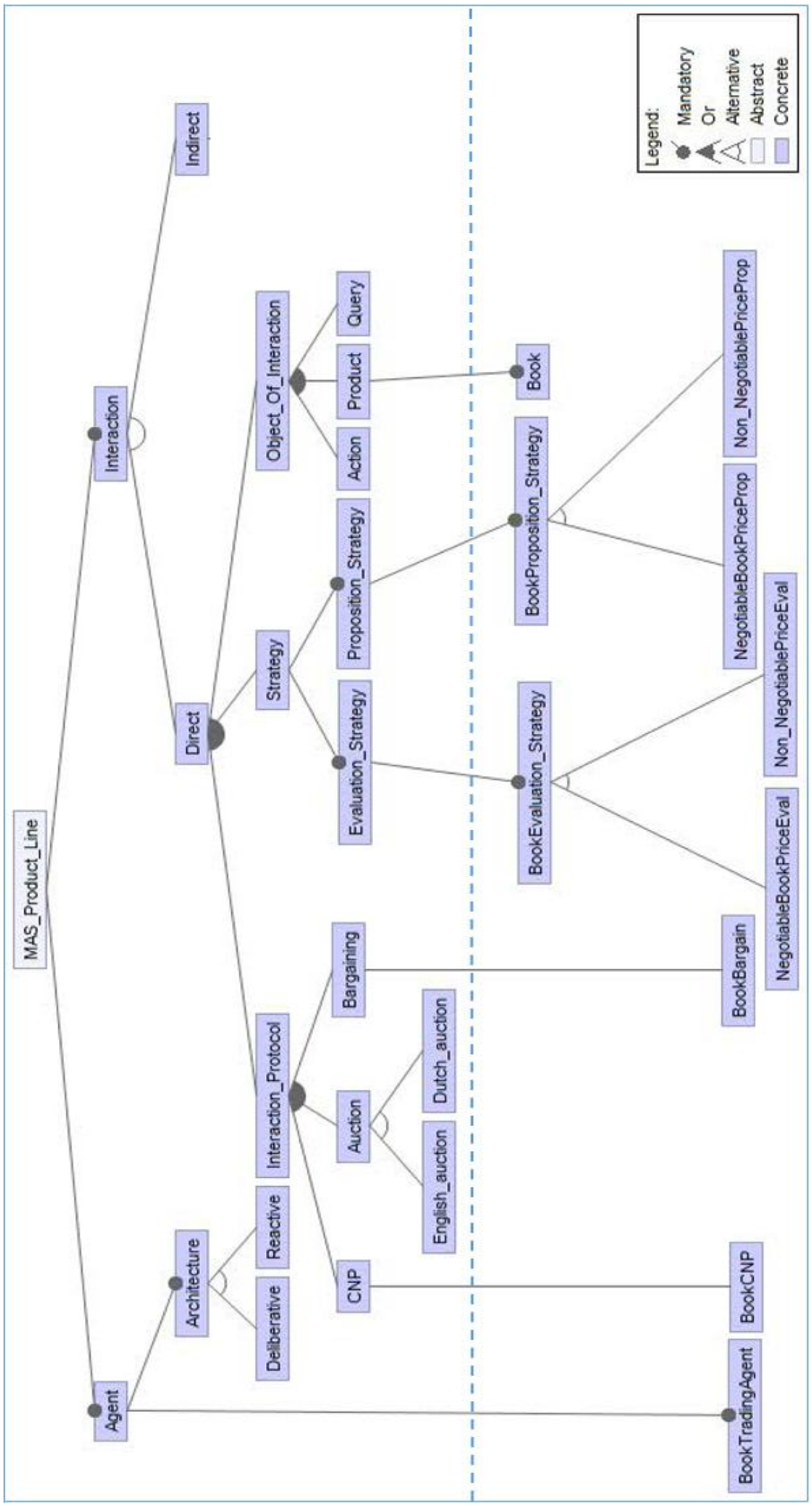


FIGURE 6.14 – Modèle de caractéristiques spécifique raffiné pour permettre des négociations 1 :1

— **Nouvelle association (mapping) entre le diagramme de classe et le FM spécifique :**
L'association entre chacune des nouvelles caractéristiques du modèle ainsi que les nouvelles classes supportant la nouvelle variante de vente de livres est représentée dans la Figure 6.15. Afin de représenter et de supporter les deux variantes que nous avons décrites plus haut, nous avons introduit un nouveau niveau de caractéristiques filles de la feature spécifique au domaine de vente de livres. A partir de la deuxième version du FM spécifique, il est possible de raffiner des caractéristiques spécifiques du FM. Nous avons par exemple raffiné la caractéristique *BookProposition_Strategy* afin de permettre au concepteur de faire un choix alternatif entre une stratégie de proposition d'agent vendeur à prix fixe par le biais de la feature spécifique *Non_NegotiableBookPriceProp* ou à prix négociable grâce à la feature spécifique *NegotiableBookPriceProp*.

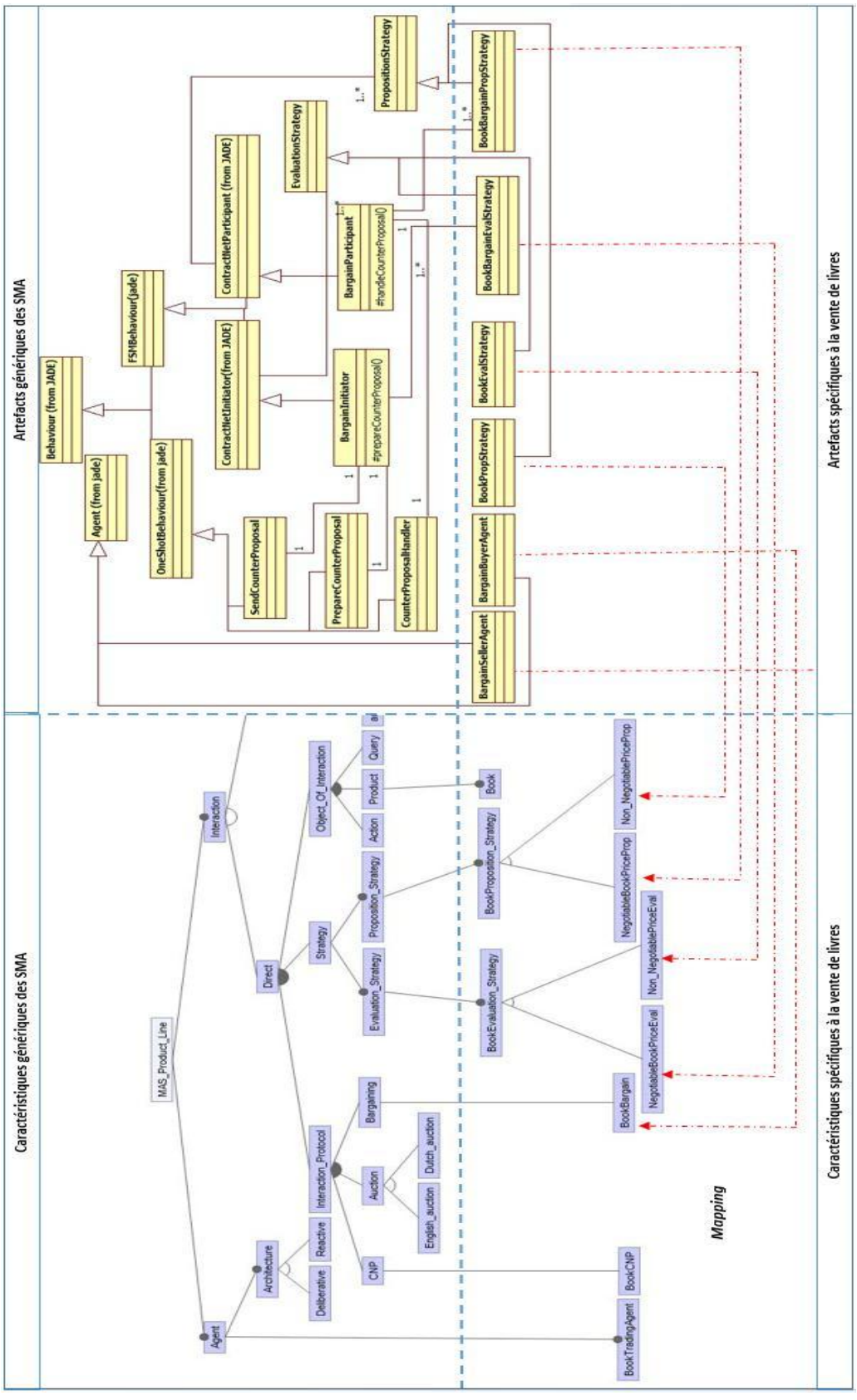


FIGURE 6.15 – Association (mapping) entre le FM spécifique de négociations 1 :1 aux artefacts spécifiques l'implémentant

6.2.2.2 Raffinement (passage aux interactions 1 :n)

Dans ce qui suit nous raffinons notre modèle de vente de livres en passant à des interactions de type 1 à n entre les agents acheteurs et vendeurs. Dans un premier temps nous allons considérer des interactions entre un vendeur et plusieurs acheteurs. Et dans un second temps nous allons considérer des interactions entre un agent acheteur et plusieurs agents vendeurs.

Cependant, comme lors du passage à des interactions de la première catégorie(1 vendeur et n acheteurs) avec prix fixe, aucune variabilité n'est observée, vu que le protocole du contract net reste applicable à des interactions de type 1 :n. C'est la raison pour laquelle, nous allons considérer uniquement des prix négociables. On observe ainsi une variabilité générique liée au protocole d'interaction. Les variantes que nous présentons représentent des stratégies différentes de négociations de vente aux enchères. On peut ainsi retrouver une variabilité concernant le type d'enchère comme par exemple l'enchère anglaise. Le comportement des enchérisseurs peut présenter également une variabilité des stratégies utilisées par l'agent jouant le rôle d'un enchérisseur. Nous réutilisons donc les spécifications des protocoles tels que définis par FIPA.

— **Enchères Anglaise (1 vendeur et n acheteurs)**
Cette variante propose de réutiliser un protocole d'enchères Anglaises. Dans ce type d'enchères, l'agent vendeur jouant le rôle du commissaire priseur initie l'enchère en annonçant un premier prix pour le livre représentant l'objet de l'interaction. Ce prix annoncé est initialement inférieur à la valeur estimée du livre.
Après chaque annonce, l'agent vendeur devra attendre selon un temps déterminé pour voir s'il y a des agents acheteurs acceptant cette offre.
Dés qu'un acheteur se manifeste, l'agent vendeur annonce une nouvelle proposition de prix égale au dernier prix incrémenté d'une valeur appelée : *pas d'incrémentation*. Dès qu'aucun acheteur ne se manifeste au prochain tour, l'enchère est ainsi clôturée. L'agent vendeur va ensuite comparer le dernier prix accepté à celui de l'offre. Si cette valeur est supérieure à la valeur estimée du livre alors l'enchère est accordée. Dans le cas contraire, la vente est annulée.

— **Nouvelle version du diagramme de classe :**
Les nouvelles classes introduites afin de supporter cette variante sont représentées dans la figure 6.16. Le travail est réalisé en réutilisant l'ensemble des artefacts génériques permettant d'implémenter les features d'enchères Anglaise. Les deux nouvelles classes introduites sont la *Classe EnglishAuctionSellerAgent* et la *Classe EnglishAuctionBuyerAgent*.

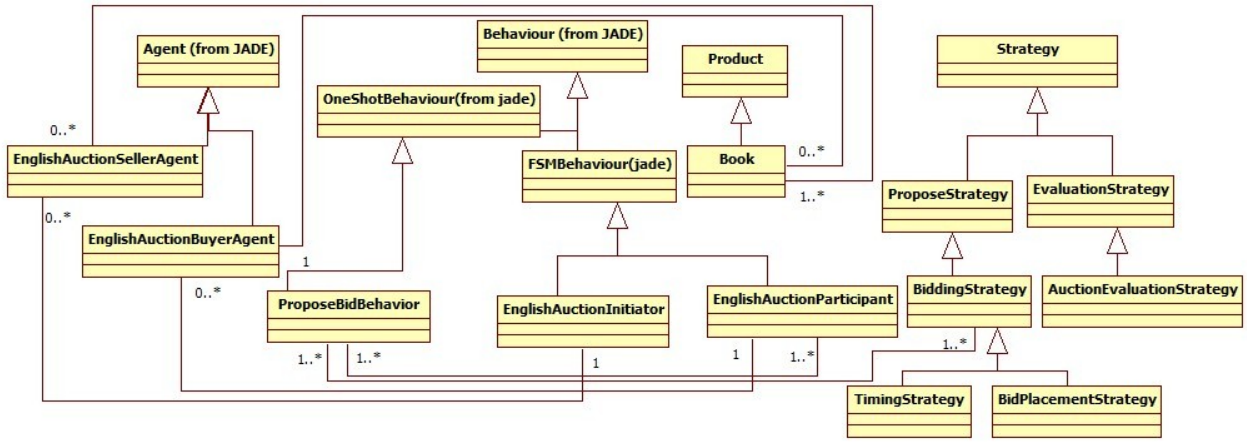


FIGURE 6.16 – Diagramme de classe raffiné pour permettre l'enchère Anglaise avec cardinalité 1 :n

— **Raffinement du FM spécifique**

Afin de permettre au modèle de supporter cette nouvelle variante, il est nécessaire d'introduire de nouvelles caractéristiques relativement aux modifications réalisées au sein du diagramme de classe. C'est ainsi que nous raffinons le FM de la Figure 6.14 par celui de la Figure 6.17. La variabilité se situant au niveau des comportements des agents Seller et Buyer, il a suffit d'ajouter la caractéristique *BookEnglishAuction*. Cette caractéristique raffine la caractéristique *English_ auction* qui est implémentée par les classes des rôles relatives au commissaire-priseur et aux enchérisseurs.

Nous réutilisons également les autres caractéristiques génériques permettant de réaliser l'enchère anglaise soit *Auction_ Evaluation_ Strategy* et *Bidding_ Strategy*.

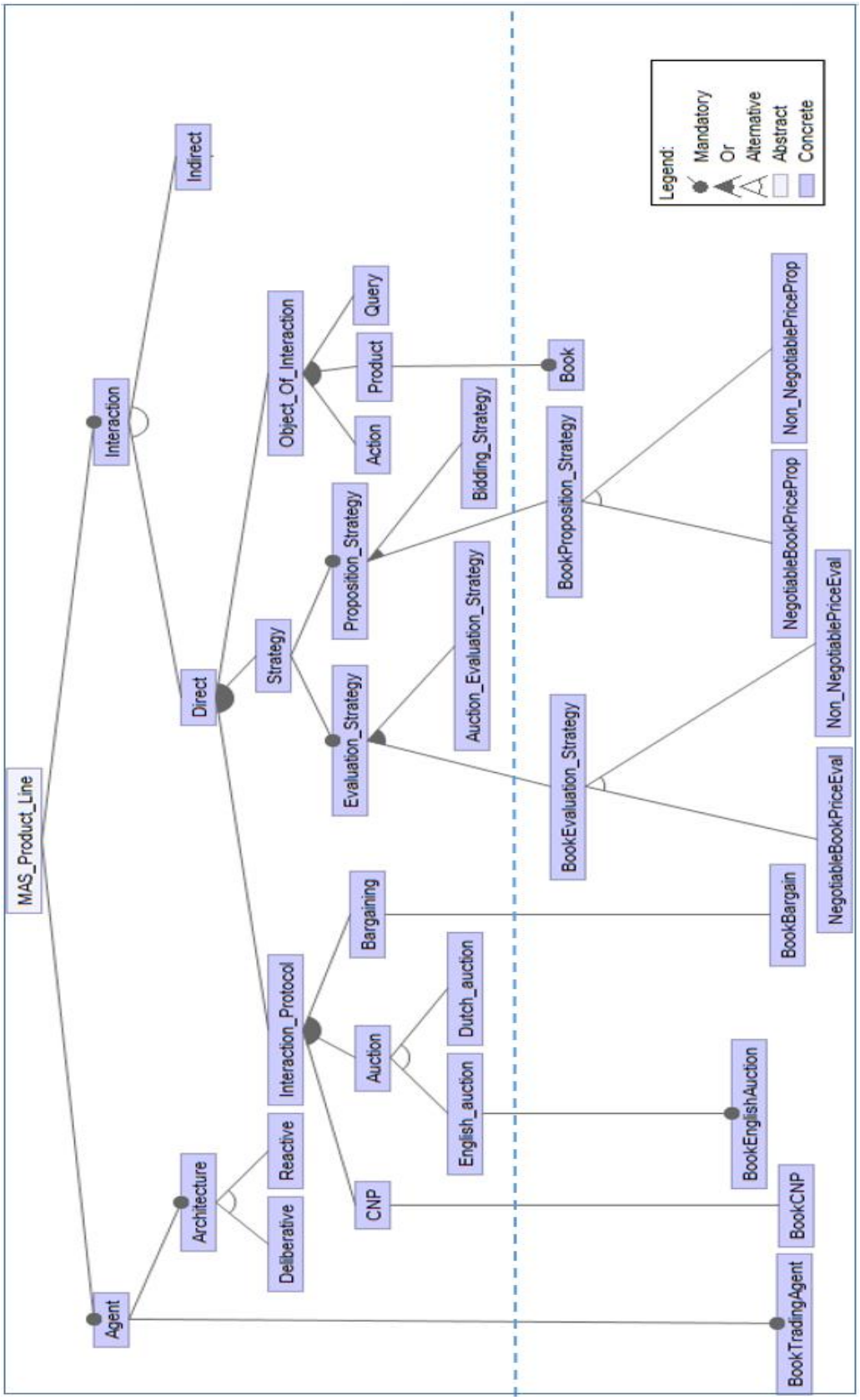


FIGURE 6.17 – FM spécifique raffiné pour permettre l'enchère Anglaise pour interactions 1 :n

— **Nouvelle association (mapping) entre le diagramme de classe et le FM spécifique :**
L'association entre chacune des nouvelles caractéristiques du modèle ainsi que les nouvelles classes supportant la nouvelle variante de vente de livres par enchère Anglaise est représentée dans la Figure 6.18.

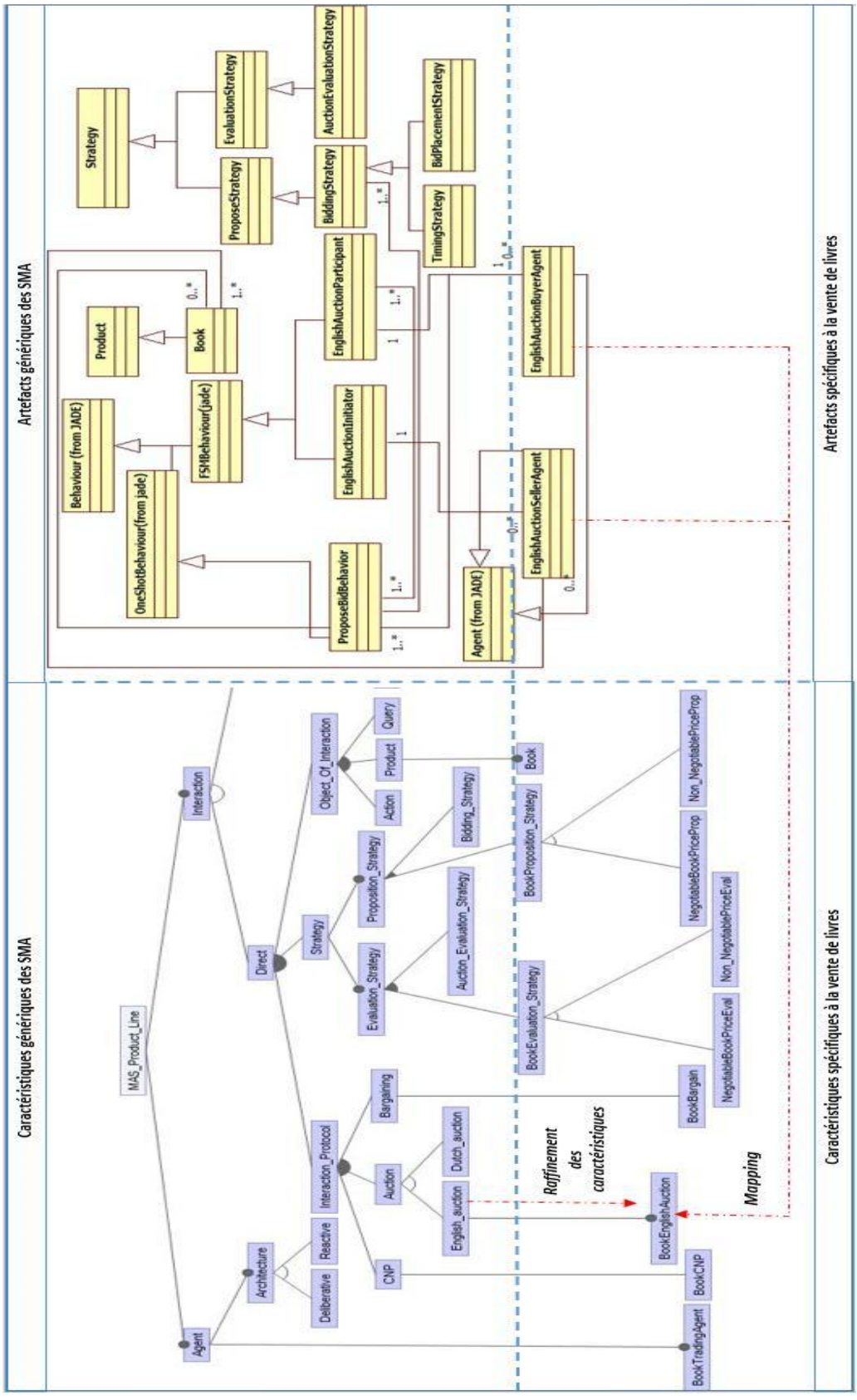


FIGURE 6.18 – Association (mapping) entre le FM spécifique d'interactions 1 :n aux artefacts spécifiques implémentant l'enchère Anglaise

6.2.2.3 Ajout des contraintes de validité des configurations et des variantes

Avant de dériver les variantes relatives à la vente de livres, il est nécessaire d'intégrer au sein du modèle les contraintes spécifiques au domaine d'application. Un exemple des contraintes est donné dans la figure 6.19.

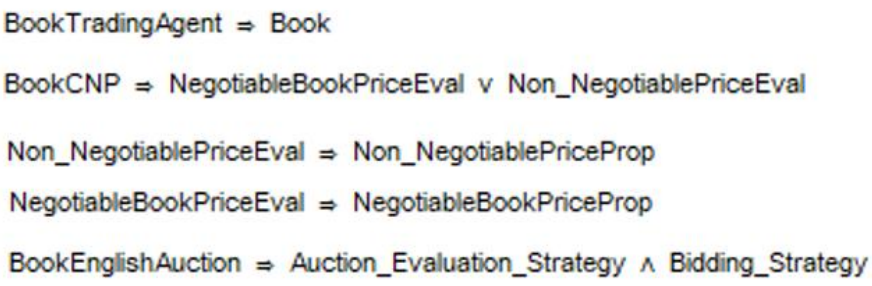


FIGURE 6.19 – Exemple de contraintes spécifiques à la vente de livres

6.2.3 Contest

Le cas d'étude du multi-agent contest 2016,2017 et 2018, a été utilisé dans le chapitre précédent afin de présenter les étapes de notre approche. Pour rappel, les agents de contest se déplacent au niveau d'un environnement représenté par une ville réaliste et ce afin d'effectuer le maximum de jobs tels que le transport de marchandise, la livraison ou encore l'assemblage. Afin de représenter la ligne de produits multi-agents du Contest nous avons tout d'abord considéré la variabilité au niveau de la caractéristique Agent. S'agissant d'Agents délibératifs suivant l'architecture BDI nous commençons par des agents minimaliste dits mono_goal effectuant un seul type de job relativement au premier algorithme de wooldridge, et par la suite nous varions la cardinalité en passant à des agents *multi_goal* pouvant effectuer d'avantages de jobs. De plus, nous proposons différentes stratégies décisionnelles d'agents afin d'observer la variabilité de la ligne et de la spécifier pour connaître les artefacts génériques qu'il est nécessaire de réutiliser. Pour cet exemple, contrairement aux deux exemples précédents, nous n'allons pas présenter l'ensemble des associations (mapping) réalisé ni l'ensemble des diagrammes de classes obtenus dès lors que le processus reste le même que celui présenté en détail dans les deux cas d'études précédents. Nous allons donc présenter uniquement les features models spécifiques obtenues après raffinement. Cependant, nous allons pour cet exemple donner plus de détails relatifs à l'implémentation spécifique.

6.2.3.1 Cas simple : agents mono-goal

La première version de la ligne considérée, contient des agents minimalistes n'ayant qu'un seul but à atteindre dits *mono goal* agents. Par conséquent, l'ensemble des actions, croyances et perceptions des agents constituant l'équipe sont limités.

La première version obtenue du FM spécifique, est représentée dans la figure 6.20. Nous réutilisons d'un point de vue implémentation le premier algorithme du BDI proposé par wooldridgne [58]. Nous réutilisons également d'autres artefacts génériques comme ceux relatifs à l'implémentation de la perception active.

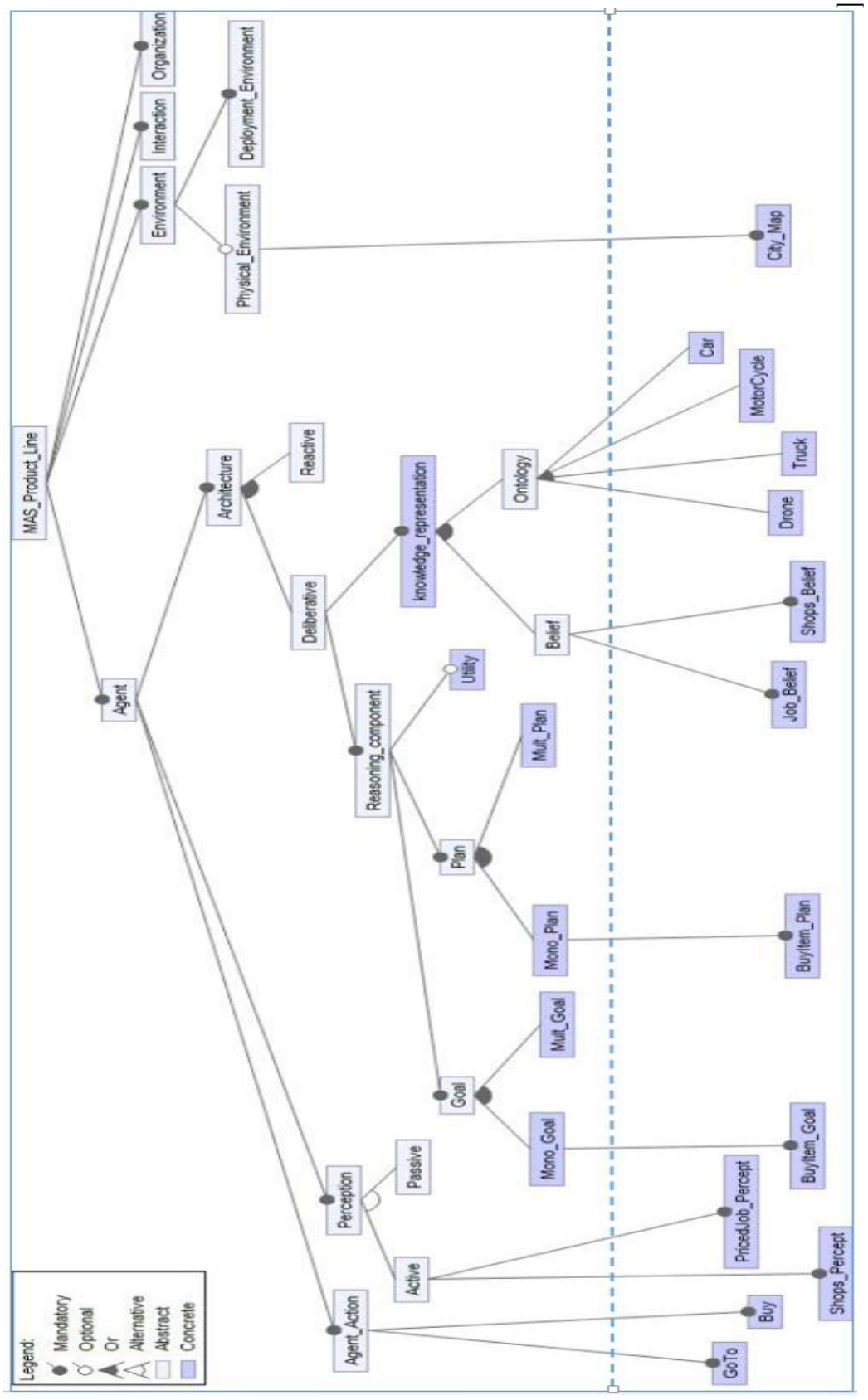


FIGURE 6.20 – FM spécifique pour les agents mono goal du Contest Contest

La Figure 6.21 représente un exemple d'association (mapping) au niveau code source entre les artefacts d'implémentation, ainsi que les caractéristiques correspondantes. Cet exemple représente un récapitulatif de l'ensemble des quatre (4) activités de l'ingénierie du domaine proposées par notre approche. Nous pouvons ainsi distinguer les éléments suivants : 1) La caractéristique *Filtering_ Contest_ Perception* permet de filtrer les perceptions actives des agents au sein de l'environnement du Contest. Cette caractéristique raffine donc la caractéristique générique *Active* qui est implémentée par un ensemble d'artefacts génériques relatifs aux modules de capteurs (function *Sensing*), d'interprétation (function *interpreting*), et de filtrage (function *filtering*) ; 2) L'implémentation de la caractéristique *Filtering_ Contest_ Perception* afin de filtrer l'ensemble des perceptions de l'agent tels que les *shops*, et *facilities* par réutilisation de l'artefact d'implémentation générique de la fonction de *filteringPercepts()* de la perception active. Cette fonction est invoquée lors du comportement de l'agent relatif à la perception (gettingAllPercepts()).

6.2.3.2 Raffinement avec passage aux agents multi-goals

Le raffinement du modèle est réalisé par le passage à des agents pouvant avoir plusieurs différents buts à atteindre. Ces agents dits *multi goal* devront exécuter le second algorithme de wooldrige [58].

La nouvelle version obtenue pour le FM spécifique est représentée dans la figure 6.22.

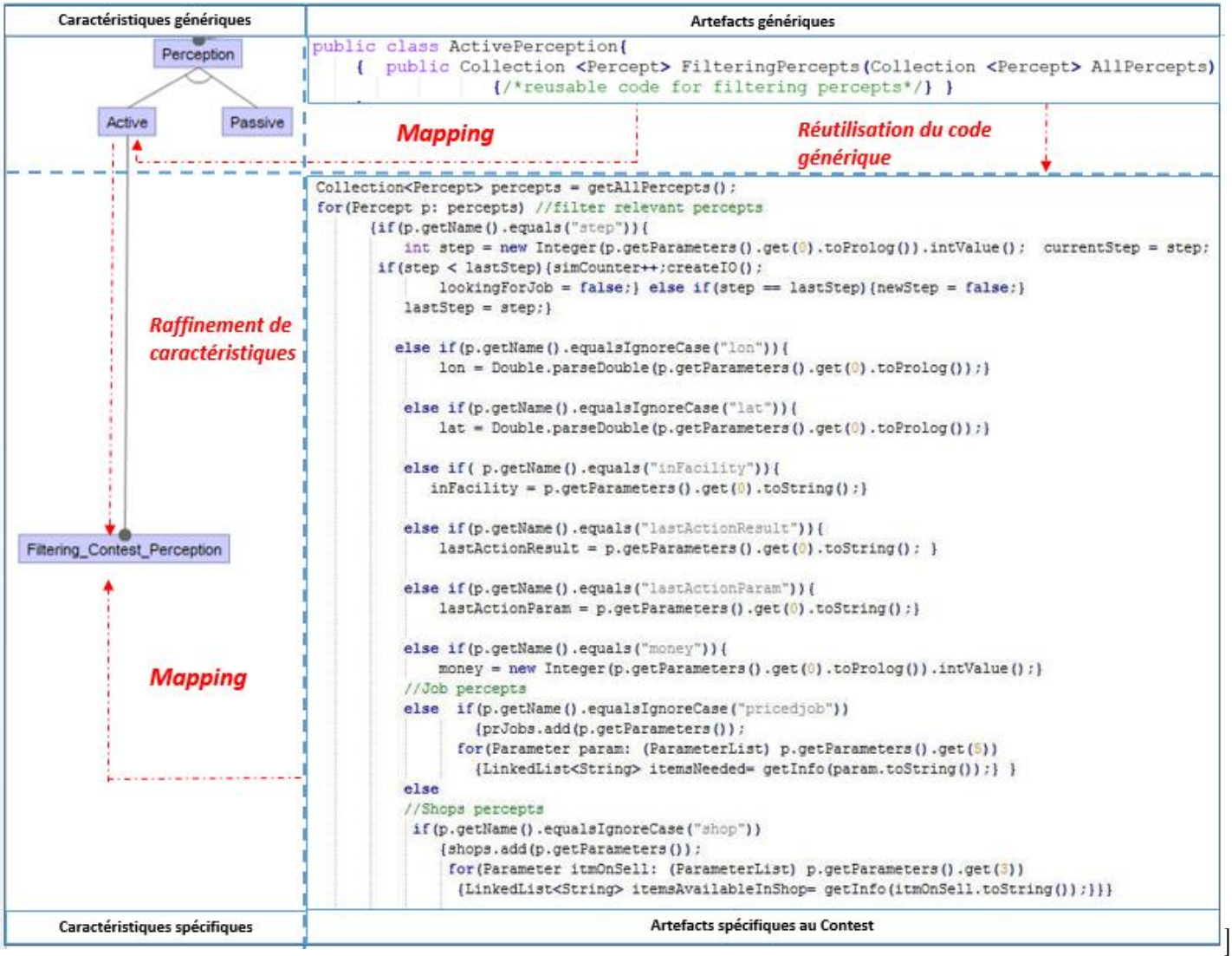


FIGURE 6.21 – Raffinement et implémentation spécifique avec association (mapping) pour filtrer les perceptions actives des agents de Contest

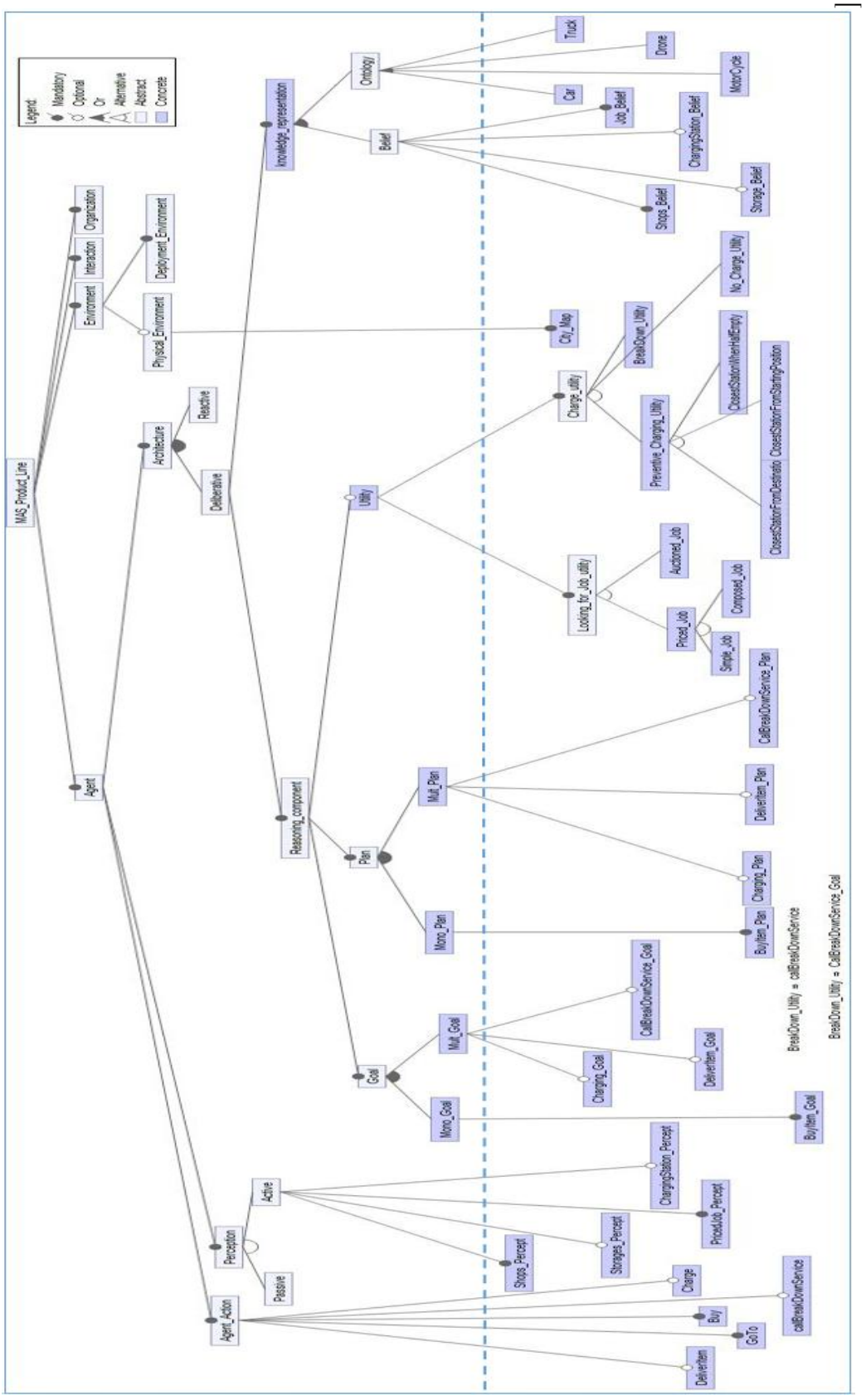


FIGURE 6.22 – Raffinement spécifique du feature model pour les agents multi goal du Contest

En analysant de plus près la variabilité des features et artefacts de la ligne, mise à part celle relative à l'algorithme d'exécution du BDI, nous observons un bon nombre d'éléments variant tels que la perception d'agents ou encore les actions que l'agent peut exécuter. Par exemple, des actions telles que *charge*, ou encore *callBreakdownService_ action* étendent le modèle.

La Figure 6.23 représente une implémentation spécifique qui permet de supporter la variabilité des actions de Contest. En effet, la première variante présentée (voir cas simple) suppose que les agents n'exécutent que des actions d'acquisition de marchandise ("buy") et de déplacements ("goto"), cependant afin de permettre aux agents d'effectuer d'autres actions celles-ci seront représentées de façon optionnelle vu qu'elles ne sont pas supportées par la version minimaliste d'agents ; comme la possibilité d'assemblage ("assemby") ou encore gérer les pannes d'essence en appelant le service de dépannage ("callBreakDownService"). Cette dernière est donc implémentée par la caractéristique *callBreakdownService_ action*, et la caractéristique *Contest_ Agent_ basic* permettant de l'inclure de façon optionnelle. Dans le cas contraire c'est à dire si la caractéristique optionnelle n'est pas sélectionnée, la fonction *replaceGoalCallbreakdownserviceAction()* sera invoquée avec un corps vide et de ce fait exclura la possibilité d'exécution de ce type d'action par les agents de la variante dérivée et déployée au sein du Contest.

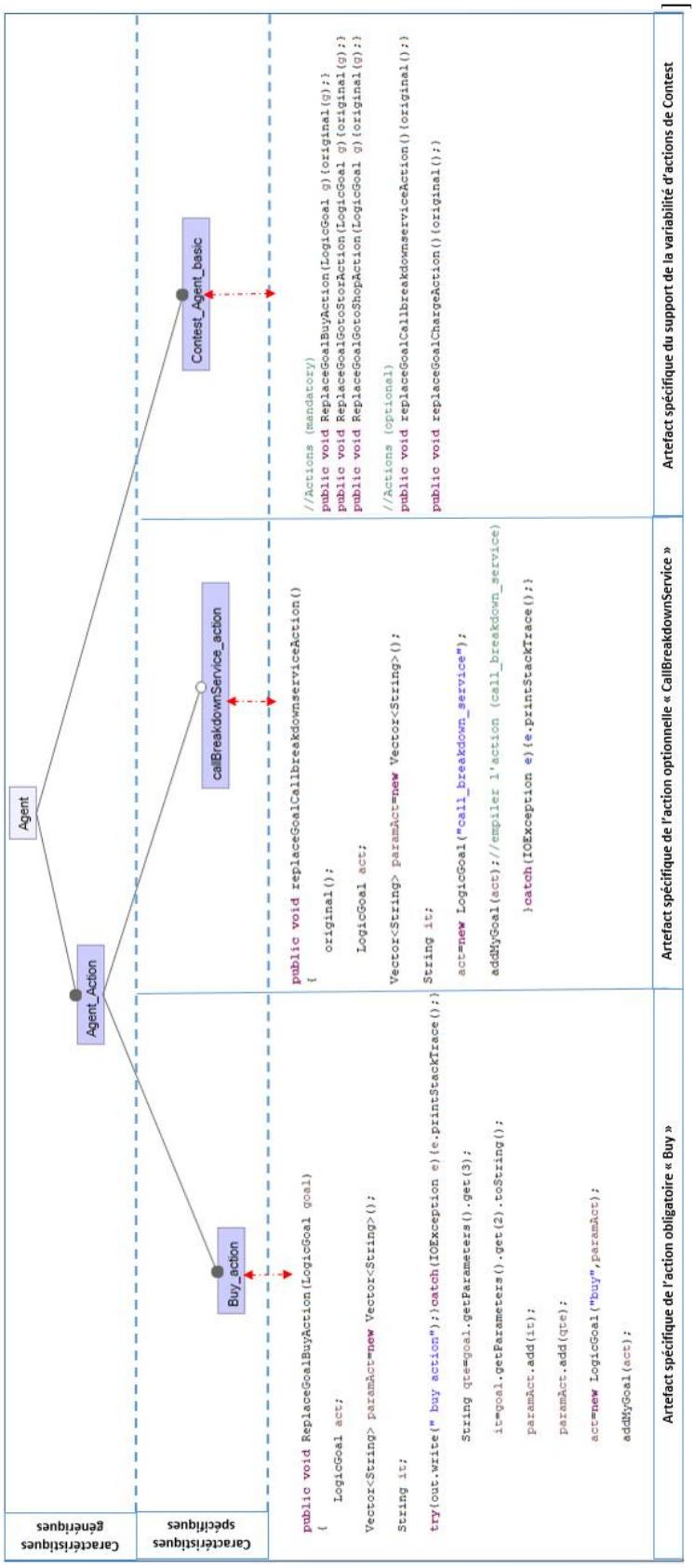


FIGURE 6.23 – Implémentation spécifique pour le support d'actions obligatoires et optionnelles au sein des agents de Contest

6.2.3.3 Ajout des contraintes de validité des configurations et des variantes

Avant de dériver les variantes du multi-agent Contes, il est nécessaire d'intégrer au sein du modèle les contraintes spécifiques au domaine d'application. Un exemple des contraintes est donné dans la figure 6.24.

FIGURE 6.24 – Exemple de contraintes spécifiques au multi-agent Contest

6.2.4 Analyse des résultats obtenus

L'une des principales questions de recherches auxquelles tente de répondre la présente thèse, concerne les solutions qui pourraient faciliter le développement d'applications multi-agents similaires. Les trois cas d'études présentés dans cette évaluation de la faisabilité de notre approche, tentent de répondre à cette problématique entre autres. Cette évaluation, a montré que pour chacun des problèmes présentés au niveau de chaque cas d'étude, il a été possible grâce à notre approche de faciliter le développement, et par conséquent la construction d'une ligne de produits multi-agents, répondant aux besoins spécifiques des cas d'applications de gestion de l'emploi du temps, de ventes de livres et d'équipes de jeu (contest) respectifs. Les résultats obtenus, ont montré la possibilité de supporter des variantes à configurations minimales, et la possibilité de construire chacune d'entre elles grâce au premier niveau du modèle de caractéristiques que propose notre approche, ainsi que de son implémentation. Celui-ci rappelons-le, est issu d'une analyse des points communs et variables des SMA, et qui sont indépendants du domaine d'application. Plus encore, les cas d'études de cette évaluation ont montré que l'ensemble des artefacts d'implémentations associés aux caractéristiques du premier niveau de variabilité (générique) que nous avons proposés, répondaient au problème de recherche lié à la réutilisation, et plus précisément à la réutilisation du code source pour le développement d'applications multi-agents similaires. En effet, cette évaluation a montré la faisabilité de réutiliser les artefacts d'implémentations associés à chacun des concepts issus du premier niveau du modèle de caractéristiques. Ce qui permet à son tour, de répondre au problème de recherche relatif à la minimisation de l'écart existant entre les concepts et leurs implémentations. En effet, dès

lors que derrière la réutilisation de chaque caractéristique se cache une réutilisation des implémentations des différents concepts de des SMA, les variantes spécifiques aux trois cas d'études présentés réutilisent également le code source associé. Les résultats que nous avons obtenus, ont montré également la faisabilité de notre approche à procéder au raffinement du modèle afin de répondre à des nouveaux besoins de la ligne de produits plus facilement. La facilité à réaliser un tel raffinement, réside dans le fait que l'ensemble des artefacts nécessaires est fourni par notre approche, et permet ainsi aux développeurs de maintenir l'ajout de nouveaux produits au sein de la ligne sans avoir à démarrer de zéro.

6.3 Évaluation prospective

Dans cette section, nous présentons les résultats obtenus suite à l'implication de groupes d'étudiants pour une évaluation prospective. Cette évaluation a pour but d'observer les résultats d'étudiants qui présenteraient le plus grand nombre de variantes valides pour le Contest.

Nous expliquons dans ce qui suit, la démarche adoptée ainsi que les conditions dans lesquels l'évaluation a eu lieu. Nous finirons par une analyse des différents résultats obtenus.

6.3.1 Démarche et conditions de déroulement de l'évaluation

La démarche adoptée dans cette évaluation avait pour but d'observer et d'analyser les résultats obtenus par une une quinzaine d'étudiants de master. Nous devions analyser leurs résultats en termes de caractéristiques considérées issues des deux niveaux (générique et spécifique) ; afin de voir leur impact sur l'ensemble des variantes générées pour lesquelles les simulations avaient été lancées.

Les étudiants ayant participé à l'expérimentation sont tous de même niveau, et possèdent les mêmes connaissances en systèmes multi-agents, ainsi qu'en lignes de produits logiciels. Cependant, il s'agit de leur première expérience en termes d'approche de lignes de produits multi-agents ou MAS-PL.

Les participants au projet, étaient répartis en groupes de 2 à 3 étudiants, sur une durée de quatre mois. Ils avaient pour but de développer une ligne de produits multi-agents du même domaine d'application que celui qui a servi à l'évaluation de la faisabilité de notre approche, et dont les détails ont été présentés dans la section précédente (voir 6.2.3). Le domaine en question étant celui du multi-agent Contest.

Les étudiants ont été guidés par les étapes du processus général de développement de lignes de produits logiciels. Ce processus issu du framework général de lignes de produits logiciels, étant celui sur lequel est fondée notre approche (voir 4.2.1).

L'ensemble des étapes que devaient suivre les étudiants, a été généré par l'approche «Meduse » [159]. Cette approche, permet d'adapter les processus de développement en fonction des besoins du projet. Sa particularité, est qu'elle est basée sur les lignes de produits logiciels et les techniques d'ingénierie des méthodes. Elle tient donc compte des similitudes et des différences entre les processus ; ainsi que les fragments de processus réutilisables.

Concernant les outils utilisés par les étudiants, ils comprenaient les mêmes outils utilisés durant l'évaluation de la faisabilité de notre approche. Ces derniers étaient constitués d'une part de l'outil featureIDE sous éclipse, qui leur a servi pour la modélisation de la ligne de produits, et son implémentation. Et d'autres part, du simulateur MASSim(Multi-Agent Systems Simulation) [161]. Ce dernier, ayant permis aux étudiants de lancer les simulations relatives à chacune des variantes Contest, afin de s'assurer de leur validité et conformité par rapport aux besoins spécifiques de chaque SMA dérivé.

Il est à noter qu'aucun des modèles, ni implémentations de départ proposés par notre approche n'ont été remis aux étudiants. Ces derniers, devaient par eux mêmes procéder à une analyse des similitudes et de la variabilité qu'ils pourraient ensuite exploiter durant la réalisation du modèle de caractéristiques spécifique au Contest, ainsi que son implémentation. L'objectif, rappelons-le étant d'obtenir le plus grand nombre possible de variantes valides pouvant ainsi être déployées.

6.3.2 Analyse des résultats obtenus

Dans notre démarche d'analyse des résultats obtenus, nous nous intéressions particulièrement à l'impact que pourrait avoir la détection des caractéristiques réutilisables indépendemment du domaine d'application sur les variantes obtenues. Notre intérêt s'est porté également sur le degré de difficulté des étudiants sur la détection de chacun des niveaux de variabilité.

Comme résultat, nous avons pu distinguer trois catégories d'étudiants après analyse de leur travaux respectifs : CAT1, CAT2 et CAT3. Ces résultats sont représentés dans le tableau 6.1.

CAT1 représente les étudiants qui ont échoué à détecter une variabilité indépendante du domaine d'application. Cet échec de détection durant l'analyse a eu un impact sur les variantes qu'il est possible de déployer ; et qui représentent le plus faible nombre de configurations valides, et qui a atteint le nombre maximal de 10. La majorité des étudiants se trouve dans cette catégorie, et représente 44.44 % des groupes constitués. Les étudiants se sont focalisés lors de leur travail uniquement sur une analyse de variabilité spécifique au domaine du multi-agent Contest, ce qui ne leur a donc pas permis d'obtenir des caractéristiques et artefacts réutilisables pour d'autres domaines d'application.

CAT2 comporte les groupes d'étudiants, qui ont détecté certains aspects de variabilité indépendante du domaine d'application. Cependant, aucun résultat de leur analyse n'a été inclus

au sein de leur modèle de caractéristiques. Cette catégorie représente 33.33 % de l'ensemble des groupes d'étudiants. Ces groupes ont donc réussi partiellement à atteindre l'objectif, car ils ont réussi à détecter des caractéristiques génériques réutilisables des systèmes multi-agents, mais ne les ont pas exploitées au sein de leur modèle. Par exemple, certains groupes ont proposé une variabilité organisationnelle. Ils ont proposé entre autres des organisations d'équipes de façon centralisée ou décentralisée. Cependant, leur modèle ne permet pas la sélection de telles caractéristiques au niveau des configurations possibles de l'équipe.

CAT3 concerne les groupes qui ont détecté une variabilité indépendante du domaine d'application, tout en l'introduisant au sein du modèle de caractéristiques. Ces groupes, ont par conséquent intégré au sein de leur modèle de caractéristiques, les deux niveaux de variabilité. Bien que cette catégorie représente le plus faible taux qui est celui de 22.22 % des groupes constitués ; elle représente néanmoins les meilleurs résultats obtenus, et qui représentent un nombre maximal de 32 configurations valides. Le nombre total de caractéristiques réutilisables proposées par ces groupes comporte au maximum le nombre de huit (8) caractéristiques réutilisables indépendamment du Contest, et qui s'articulent principalement autour des caractéristiques *d'Interaction* et *d'Organization*.

Catégories de groupes d'étudiants	Scoping de la variabilité indépendante du domaine d'application	Inclusion de la variabilité indépendante du domaine au sein du FM	Scoping de la variabilité spécifique au domaine d'application	Nombre total de caractéristiques génériques détectées	Nombre total d'artefacts génériques implémentés	% d'étudiants de chaque catégorie	Nombre maximal de configurations valides
CAT1	-	-	x	0	0	44.44	12
CAT2	x	-	x	0	0	33.33	12
CAT3	x	x	x	8	2	22.22	32

TABLE 6.1 – Quelques résultats d'expérimentations de groupes d'étudiants

Les résultats obtenus montrent bien le besoin d'avoir un point de départ pour chacune des

activités de modélisation et d'implémentation de la ligne de produits multi-agents, dès lors que plus de la moitié des étudiants n'ayant pas réussi à construire le premier niveau de modèle de caractéristiques et d'implémentation indépendamment du domaine d'application, présentaient de moins bons résultats en comparaison avec les groupes qui avaient exploité les deux niveaux de variabilité au sein de leur modèles.

Notre approche pourrait ainsi fournir à la catégorie CAT1 un point de départ pour la représentation d'une variabilité indépendante du domaine d'application. Elle permettrait aussi à CAT2 d'exploiter les caractéristiques qu'ils avaient détectées afin de dériver des variantes multi-agents. De plus, notre approche fournirait à CAT3 un plus grand nombre de caractéristiques et d'artefacts réutilisables que celui de ceux qu'ils avaient réussi à détecter.

Pour une évaluation future, il serait intéressant de réaliser une évaluation approfondie avec les étudiants. L'évaluation en question permettrait d'observer l'impact que pourrait avoir la réutilisation du modèle de caractéristique générique (resp. implémentation) proposé par notre approche, sur l'effort d'implémentation des groupes d'étudiants.

Dans la prochaine section, c'est d'ailleurs ces aspects de notre approche que nous allons évaluer, en observant entre autres les métriques relatives aux efforts de développement de la ligne de produits multi-agents du Contest.

Notre approche MAS-PL utilise des notations connues couvrant des aspects de conception et d'implémentation, ce qui facilite son utilisation et adoption par les développeurs multi-agents. De plus, dès lors que nous fournissons un FM et artefacts génériques réutilisables, les concepteurs ainsi que les développeurs des SMA n'auront ni à construire le FM spécifique ni à développer le code source relatif de zéro.

Une des limites de notre approche étant que nous n'avons pas considéré des artefacts réutilisables relatifs à l'organisation du système multi-agents. Un autre inconvénient c'est que il y-a plus de variabilité spécifique au domaine comparé à celle qui est générique, et le nombre total d'artefacts réutilisables nécessite d'être augmenté.

Cependant, dès lors que notre approche est incrémentale, il serait possible d'étendre notre travail en raffinant la variabilité indépendamment du domaine d'application. Comme par exemple analyser les aspects d'auto-organisation du système afin de capitaliser leur utilisation. Il serait également intéressant d'évaluer notre approche pour d'autres domaines d'application par des groupes d'étudiants afin d'améliorer notre approche en fonction de leurs retours.

6.4 Évaluation de notre approche

6.4.1 Choix des outils d'implémentation et d'évaluation

Avant d'évaluer notre approche, nous l'avons tout d'abord implémentée grâce à l'outil Feature IDE qui est un plugin à ajouter au niveau de l'environnement de développement d'éclipse. Cet outil nous a permis de spécifier les deux niveaux de nos modèles de caractéristiques FM (générique et spécifique), d'intégrer l'ensemble des implémentations réutilisables indépendamment du domaine d'application, d'implémenter les artefacts spécifiques au domaine d'application, d'ajouter des contraintes génériques et spécifiques, de tester différentes configurations, et d'être en mesure de dériver les variantes multi-agents.

Comme notre implémentation utilise une approche compositionnelle pour la construction de variantes, nous avons utilisé FeatureHouse [32], qui est un composeur de code, qui nous a permis de dériver les variantes multi-agents avant de pouvoir les déployer.

Le déploiement de chacune des variantes obtenues durant cette évaluation, après configuration et dérivation, a été fait au niveau de la plateforme du multiagent Contest 2016, 2017 et 2018 [1]. Plus précisément, les simulations de chacune des variantes obtenues par notre approche, ont été lancées grâce à MASSim (Multi-Agent Systems Simulation), qui est une plateforme spécialement conçue pour les besoins du Multi-agent Programming Contest. Cet environnement ayant comme objectif principal de permettre une évaluation des approches de coordination et de coopération des SMA constitués d'agents délibératifs assez complexes [161].

Afin de récolter l'ensemble des résultats obtenus pour chacune des métriques que nous avons choisiesd'évaluer pour chaque variante multi-agents, nous avons utilisé eclipse Metrics [2].

6.4.2 Choix des métriques de l'évaluation

Comme nous avons pu le voir, les premiers résultats de notre évaluation, ont montré la faisabilité de notre approche pour la dérivation des variantes pour les trois cas d'études dont celui du multi-agent Contest (voir 6.2.3). Ce dernier, pour lequel au niveau de cette section, d'autres aspects sont évalués. La question de recherche à laquelle cette évaluation tente de répondre est celle du problème de réutilisation au niveau des lignes de produits multi-agents.

En effet, l'ensemble des métriques que nous avons choisies ont pour but d'évaluer le gain d'effort de spécification et d'implémentation des similitudes et de la variabilité des lignes de produits multi-agents. Notre objectif, est d'observer l'impact que peut avoir la réutilisation des modèles et implémentations génériques que notre approche propose, sur l'effort de développe-

1. http ://multiagentcontest.org
2. http ://eclipse-metrics.sourceforge.net

ment d'une ligne de produits multi-agents dans un domaine particulier. Les métriques choisies par conséquent sont les suivantes :

- Le nombre de caractéristiques spécifiques et non spécifiques : Pour chaque variante, nous avons retenu cette métrique afin de nous permettre de calculer le pourcentage des caractéristiques issues de notre modèle générique, et qui sont réutilisées.
- Le pourcentage des caractéristiques non spécifiques réutilisées : Ce pourcentage est calculé grâce aux premières métriques. Il indique le degré de réutilisation qui a permis d'obtenir chaque variante, et par conséquent le gain de temps pour la spécification des similitudes et de la variabilité de la ligne de produits multi-agents.
- Le nombre de lignes de code et de méthodes générées : Nous avons retenu cette métrique, car elle sert de référentiel pour le calcul du pourcentage de lignes de code source et de méthodes réutilisées au sein de chaque variante.
- Le pourcentage de méthodes non spécifiques réutilisées : Ce pourcentage est calculé sur la base de la métrique précédente, et donne une indication sur le degrés de réutilisation de code source apporté par notre approche. En effet, cette métrique indique le gain d'effort durant l'activité d'implémentation spécifique au domaine d'application grâce à la réutilisation d'implémentations non spécifiques pour chaque variante dérivée.

6.4.3 Dérivation et simulation des variantes du multi-agent Contest :

Afin de s'assurer que l'ensemble des variantes qui ont servi à notre évaluation soient valides, et conformes aux besoins exprimés, nous les avons déployées au sein de l'environnement fourni par Contest, après les avoir dérivées. Après la fin de chacune des simulations, une fois les missions des variantes d'équipes de jeu terminées, les métriques ont été récoltées.

Pour cette évaluation nous avons considéré deux types de variantes de la ligne de produits multi-agents. La première prenant en considération uniquement des caractéristiques d'agents, et la seconde prenant en considération des paramètres d'interactions et d'organisation d'agents au sein de l'équipe de jeu.

6.4.3.1 Analyse des résultats relatifs à la variabilité des caractéristiques d'Agents :

La première version de la ligne de produits ne comporte aucun aspect d'interaction ni d'organisation de l'équipe d'agents, et considère seulement des agents devant accomplir leur jobs à prix fixe (priced jobs).

Ce cas d'utilisation offre plus de 5948 configurations possibles comme représenté dans la figure 6.25.

Nous présentons dix (10) des variantes d'agents dérivées et simulées. Les configurations sur lesquelles se basent chacune des variantes obtenues, ont été faites après ajout des contraintes

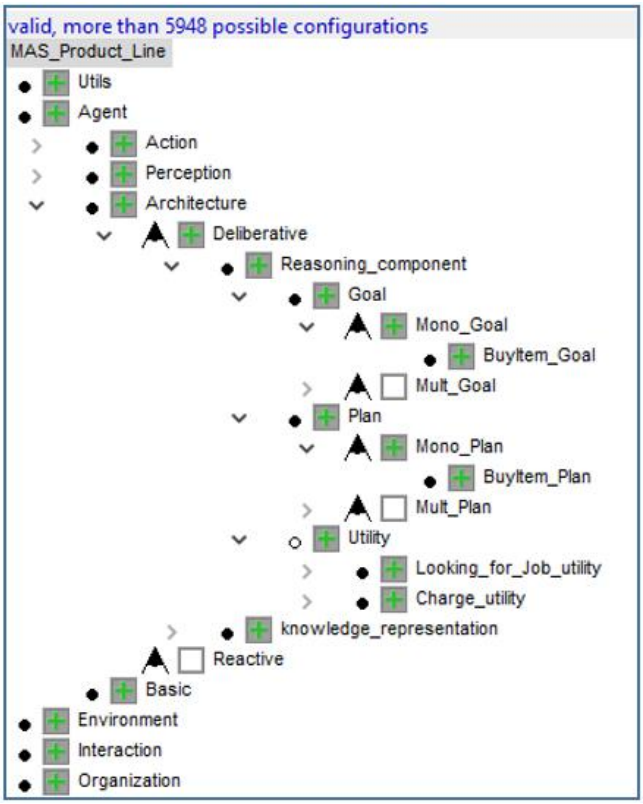


FIGURE 6.25 – Configurations valides des variantes d'Agent de Contest

spécifiques au domaine d'application, et ce afin de garantir la dérivation de variantes sur la base de configurations valides. De plus, afin de garantir que les variantes obtenues par le biais de notre approche, répondaient bien aux besoins, les simulations relatives à chacune de ces variantes, ont été lancées jusqu'à la fin d'exécution des missions devant être effectuées par chaque équipe de jeu (variante). Les métriques ayant servi à notre évaluation proviennent donc de variantes pour lesquelles la réponse aux besoins est garantie.

Les variantes sont nommées de CAV1 (Contest Agent Variant) à CAV10. Le tableau 6.2 représente les configurations de chaque variante. Chaque configuration est réalisée par la sélection de caractéristiques génériques (telle que *Mono_ Goal*) et spécifiques au Contest (telle que *Car*). La sélection des caractéristiques se fait afin de répondre à chacun des besoins (voir tableau 4.5). Les caractéristiques GMF font référence aux caractéristiques génériques (Generic Mas Features), tandis que les caractéristiques CMF font référence aux features spécifiques au contest (Contest Mas Features).

Les variantes CAV1 et CAV2 présentent une configuration minimale qui comporte un total de six (6) caractéristiques obligatoires (mandatory). CAV1 correspond à l'agent minimaliste (version la plus simple), représenté dans la figure 4.19.

CAV3 à CAV10 représentent les variantes avec un nombre maximal de huit (8) caractéristiques sélectionnées. Ces variantes suivent le second algorithme du BDI [58] ; et représentent une variabilité concernant leur utilité de recharge. En fonction des différents points de variabilité, plus ou moins de méthodes et de lignes de code source sont dérivées. Nous les avons calculées pour chacune des variantes avec eclipse Metrics.

Le pourcentage de caractéristiques et méthodes réutilisées à partir de l'ensemble des caractéristiques et artefacts réutilisables, montre les deux principaux avantages de notre approche. En effet, cette dernière offre la possibilité de réutiliser les caractéristiques et artefacts génériques. Le pourcentage des caractéristiques réutilisées parmi l'ensemble des caractéristiques que nous offrons varie de 25% à 33 %, tandis que le pourcentage d'artefacts d'implémentation (méthodes) réutilisés varie de 7% à 10.18 %.

Nos résultats montrent également l'avantage d'utilisation de lignes de produits logiciels au sein de notre approche. Il y-a un gain d'effort de développement de 354 lignes de code sources dés lors qu'elles sont directement dérivées au lieu d'être ré-implémentées de zéro. Ce gain, concerne les variantes de CAV1 à CAV10, en plus d'un gain d'effort de développement relatif à 25 méthodes.

Plus encore il nous est possible de comparer les variantes en fonction de leurs stratégies décisionnelles utilisées au sein de leur utilité. En effet, en utilisant notre approche, il est possible de comparer les performances des différentes variantes d'agents plus facilement dès lors que les agents sont automatiquement dérivés et peuvent ainsi être déployés plus rapidement. Par exemple, lors des simulations, nous avions pu distinguer entre la meilleure et plus mauvaise des stratégies d'agents utilisées pour leur rechargement de batteries pour effectuer leurs déplacements. De plus, il nous a été possible de comparer nos variantes en les faisant jouer les unes contre les autres afin de distinguer l'équipe la plus performante pour effectuer les jobs.

6.4.3.2 Analyse des résultats relatifs à la variabilité des caractéristiques d'organisation d'agents :

La seconde version de la ligne de produits logiciels comporte certains aspects d'interactions et d'organisation d'agents au sein de l'équipe de jeu. La ligne de ces variantes multi-agents est composée de quatre (4) variantes désignées de CTV1 (Contest Team Variant) jusqu'à CTV4. Les configurations de chacune de ces variantes sont réalisées sur la base des besoins représentés dans le tableau 4.5. Quant aux résultats obtenus sur l'ensemble de ces équipes, ils sont représentés dans le tableau 6.3. Ces résultats montrent un taux de réutilisation de caractéristiques qui varie de 27.27 % à 30.76%. Ces variantes qui considèrent certains aspects d'interactions et d'organisation présentent un meilleur taux de réutilisation en comparaison aux variantes d'agents de contest présentées plus haut.

Variantes d'Agents du Contest	Caractéristiques MAS-PL																	Métriques				
	GMF				*CMF*																	
	F1	F2	F3	F4	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13					
	Mono_Goal	Multi_Goal	Mono_Plan	Multi_Plan	BuyItem_Goal	Charging_Goal	CallBreakDownService_Goal	Item_Plan	Charging_Plan	CallBreakDownService_Plan	No_Charge_Utility	BreakDown_Utility	ClosestStationFromDestination	ClosestStationFromStartingPosition	ClosestStationWhenHalfEmpty	Car	Truck	Nombre total de caractéristiques	Lignes de code source	Nombre de méthodes	% des caractéristiques réutilisées	% des méthodes réutilisées
CAV1	x		x		x			x			x					x		6	1506	120	33	7
CAV2	x		x		x			x			x						x	6	1565	120	33	7
CAV3		x		x	x		x	x		x		x				x		8	1523	108	25	10.18
CAV4		x		x	x		x	x		x		x					x	8	1654	127	25	9
CAV5		x		x	x	x		x	x				x			x		8	1760	130	25	8.46
CAV6		x		x	x	x		x	x				x				x	8	1799	133	25	8.27
CAV7		x		x	x	x		x	x					x		x		8	1850	133	25	8.27
CAV8		x		x	x	x		x	x					x			x	8	1821	130	25	8.46
CAV9		x		x	x	x		x	x						x	x		8	1815	133	25	8.27
CAV10		x		x	x	x		x	x						x		x	8	1860	133	25	8.46

TABLE 6.2 – Quelques métriques des dix (10) variantes d'agents du Contest

Variantes d'équipes de Contest	Caractéristiques MAS-PL														Métriques	
	GMF					*CMF*										
	F1	F2	F3	F4	F5	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	
	Role	Group	Hierarchical	Non_Hierarchical	English_auction	Job_Researcher	Buyer	Decider	Bidder	Giver	Receiver	Carrier	Group_Supervisor	Car_Group	Truck_Group	Nombre total des caractéristiques	% des caractéristiques réutilisées
CTV1	x			x		x	x			x	x	x				7	28.57
CTV2	x			x		x	x			x	x	x	x	x	x	11	27.27
CTV3	x		x		x	x	x	x	x	x	x	x				10	30
CTV4	x		x		x	x	x	x	x	x	x	x	x		x	13	30.76

TABLE 6.3 – Quelques métriques de variantes d'équipes d'agents de contest

6.5 Conclusion

Dans ce chapitre nous avons montré l'ensemble des résultats obtenus suite aux différentes évaluations de notre approche. Nous avons commencé par évaluer la faisabilité de notre approche, et la facilité de son application. Nos résultats, ont montré qu'il était possible de construire une première version du modèle spécifique de caractéristiques, et de réutiliser les artefacts génériques afin d'implémenter les caractéristiques spécifiques au domaine d'application des trois cas d'études proposés. De plus, nous avons présenté les différentes associations (mapping) entre les caractéristiques génériques (resp. spécifiques) avec leurs implémentations génériques (resp. spécifiques)à chaque étape.

Nos résultats ont montré également la faisabilité relative au raffinement du modèle et de son implémentation, afin de supporter de nouvelles variantes de la ligne de produits multi-agents. Nous avons en premier lieu, commencé par le cas d'application de la gestion d'un emploi du temps, pour lequel nous avons considéré comme point de départ de la ligne de produits sans contraintes sur les salles de classes. Pour ensuite les inclure par la suite afin d'observer la variabilité et de raffiner le modèle. En second lieu, nous avons considéré le cas d'étude de la vente de livres, en démarrant avec des interactions simples entre un agent vendeur et un agent acheteur, pour ensuite passer à une cardinalité supérieure entre un agent vendeur et n agents acheteurs. Ce qui a montré, que notre approche permet de tenir compte de la variabilité au niveau de chaque dimension d'un SMA, et dans ce cas particulier du niveau des protocoles d'interactions, et des stratégies d'agents utilisés. Le troisième et dernier cas d'étude, étant celui du Contest, qui a permis de représenter des agents mono_goal, puis multi_goal, montrant la faisabilité de notre approche, à supporter des variabilités au niveau de l'agent également.

Nous avons ensuite procédé à une évaluation prospective, impliquant des étudiants qui avaient été guidés grâce aux étapes générées par l'approche Meduse [159]. Suite à cette évaluation, les résultats ont bien montré le besoin de la mise en place d'une approche capable d'améliorer les résultats obtenus par certains groupes d'étudiants qui n'avaient pas intégré de premier niveau de variabilité au sein de leur modèle, et qui par conséquent avaient de moins bons résultats que ceux qui avaient intégré les deux niveaux de variabilité au sein de leur modèle.

Pour finir, nous avons évalué notre approche relativement aux aspects réutilisables qu'elle propose. Les résultats ont montré qu'il était possible de dériver et de simuler des variantes multi-agents pour le Contest conformes aux besoins exprimés. Nous avons montré que nos travaux ont permis d'augmenter le nombre de variantes possibles allant jusqu'à 5948 configurations valides.

Notre analyse des résultats sur la base des métriques choisies, a montré que les principaux avantages de notre approche, étaient l'amélioration des aspects de réutilisation. En effet, notre approche a permis de réduire d'une part l'effort de spécification des similitudes et de la variabilité des produits mutli-agents grâce au modèle de caractéristique générique que nous

proposons.

D'autres part, notre approche a permis de réduire l'effort de développement de la lignes de produits multi-agents grâce aux artefacts réutilisables que nous proposons. Ce qui contribue aussi à réduire l'écart existant entre la variabilité et son implémentation, présentant ainsi un pas vers une capitalisation de l'expertise d'implémentation.

Bien que notre approche ait contribué à l'amélioration du processus d'ingénierie LdP multi-agents, il serait intéressant d'aller jusqu'à la validation de l'ensemble des variantes multi-agents en procédant par exemple, pour le cas du Contest au lancement de simulations entre les différentes équipes de jeu provenant des différentes variantes. En effet, les simulations réalisées ont permis de valider la conformité de chaque variante relativement aux besoins, tout en respectant l'ensemble des contraintes supportées par le modèle. Cependant, aucune simulation n'a été lançée entre les variantes de façon concurrentielle au niveau du Contest. Cette perspective, permettrait un gain de temps relatif au choix de l'équipe la plus performante parmi toutes les variantes générées.

Mise à part l'intégration des contraintes au sein du modèle, il faudrait trouver une solution pour garantir la conformité et la validité des configurations servant de base à la dérivation des variantes dérivées. En effet, il faudrait que l'approche proposée puisse intégrer un processus de validation des configurations relativement aux besoins fonctionnels, et non fonctionnels des variantes multi-agents.

Il serait intéressant aussi, d'appliquer notre approche à plus grande échelle avec une évaluation adéquate plus poussée que celle que nous avons réalisée avec les étudiants.

Pour finir, une évaluation systématique des gains de temps, ou encore des gains économiques entre autres, en comparaison aux autres approches MAS-PL existantes, fait également partie de nos perspectives.

Sixième partie

Conclusion générale

Conclusion générale

Dans ce contexte de travail de recherche, nous avons visé à promouvoir la réutilisation au sein des approches de lignes de produits multi-agents. Pour cela, nous avons proposé une approche MAS-PL qui s'inscrit parmi la classe d'extensions d'approches. Cependant, à l'inverse des extensions PASSI-PL [93], ou Gaia-PL [101] par exemple, notre approche tient compte d'une variabilité liée aux concepts du domaine des SMA. De plus, ces approches proposent principalement des extensions sous forme de stéréotypes pour l'annotation de la variabilité au niveau des différents modèles conceptuels impliqués par l'approche étendue. A l'inverse notre approche propose une extension de l'approche VOYELLES [28, 29] sous forme de modèles de caractéristiques communes et variables pour chacune des dimensions d'agent, d'environnement, d'interaction et d'organisation.

De plus, l'approche proposée est une approche incrémentale comme nous avons pu le montrer au sein de notre manuscrit, qui grâce au processus de raffinement permet d'étendre la ligne de produits multi-agents en supportant l'ajout de nouvelles caractéristiques et implémentations afin de répondre à de nouveaux besoins de la ligne. Les trois cas d'études ont permi de mettre en avant cet aspect incrémental important dans le cas de développement d'applications multi-agents similaires.

Nous avons également procédé au découpage des activités d'analyse et d'implémentation du domaine afin d'être en mesure de construire un modèle de caractéristiques, et artefacts d'implémentations à deux niveaux. Ces derniers, ont permis comme nos résultats l'ont montré, d'améliorer le processus de réutilisation, ce qui représente un pas vers un FM générique des SMA, et c'est ce qui a fait partie des objectifs que avions établis.

Notre approche a permis, de faciliter la gestion des similitudes et de la variabilité grâce à l'organisation du FM selon l'approche voyelles. Cela a également contribué à minimiser l'écart entre les concepts et leurs implémentations comme nous nous l'étions fixé au départ. Ce qui représente aussi un pas vers une capitalisation de l'expertise d'implémentation des LdP multi-agents.

Concernant l'évaluation de notre approche, cette dernière a été réalisée par le biais de trois cas d'études, parmi eux le multi-agent Contest, pour lequel nous avons dérivé la ligne de produits multi-agents correspondantes à un ensemble de variantes d'équipes jeu devant

réaliser un ensemble de jobs d'achat d'items, et de livraisons entre autres. L'ensemble des variantes multi-agents ont été dérivées sur la base de configurations valides constituées des caractéristiques issues des deux niveaux du modèle (générique et spécifique).

Les variantes ont été déployées après leur dérivation montrant ainsi que chacune d'entre elles est simulée au sein de l'environnement fourni par le Contest. Les résultats ont montré que notre approche a permis d'augmenter les taux de réutilisation des caractéristiques et d'artefacts.

Nous pensons que l'originalité de notre approche réside dans le fait que le FM spécifique au domaine n'est pas construit de zéro, dès lors qu'il peut être obtenu en réutilisant le FM des SMA que nous avons construit.

Notre approche souligne aussi l'importance de la réutilisation des artefacts afin que l'implémentation spécifique au domaine d'application ne soit pas réalisée de zéro contrairement aux approches existantes. Le développeur peut réutiliser les artefacts génériques que nous proposons afin que cela lui serve de base pour implémenter la ligne de produits en réutilisant ces artefacts.

Perspectives

Il est utile de souligner que le présent travail de recherche demeure susceptible d'être soumis à d'éventuelles améliorations possibles.

Notre approche peut être étendue peut être étendue afin de s'appliquer à un plus grand nombre d'applications multi-agents. Il serait par exemple interessant de considérer des agents mobiles, d'autres architectures d'agents ou encore des implémentations d'organisation et de mécanismes d'auto-organisation. Ainsi qu'une variabilité liée aux plateformes multi-agents.

Il serait intéressant d'évaluer aussi le passage à l'échelle ; en appliquant notre approche à un problème particulier, en réalisant une évaluation adéquate et plus poussée que celle que nous avions réalisée avec les étudiants. Il serait intéressant dans ce contexte de considérer le domaine de l'ioT (internet of Things) qui a fait l'objet de beaucoup de travaux de recherche en MAS-PL.

Nous pensons également ajouter un processus permettant de vérifier la validité des configurations, afin de s'assurer que les variantes dérivées répondent bien aux besoins fonctionnels et non fonctionnels.

Nous pourrions aussi étendre notre approche afin qu'elle supporte la variabilité dynamique au sein de la ligne de produits multi-agents.

D'autres perspectives sont détaillées dans ce qui suit.

Caractéristiques réutilisables pour l'auto-organisation du système

Au cours de notre travail, nous avons analysé et représenté les caractéristiques réutilisables d'un des mécanismes d'auto-organisation possibles, celui exploitant la *stigmergie* et plus précisé-

ment des modèles à base de phéromones. Cependant, d'autres mécanismes d'auto-organisation pourraient être exploités et intégrés au sein de notre modèle de caractéristiques. Ces mécanismes pourraient partager certaines caractéristiques avec le mécanisme analysé au sein de ce travail de recherche. Dans certaines approches par exemple, l'auto-organisation est basée sur les capacités des agents à adapter leur comportement en fonction d'une fonction d'utilité individuelle, qui détermine une récompense qu'ils essaient de maximiser en modifiant leur comportement au fil du temps. Le SMA dans son ensemble a également une fonction d'utilité globale qui n'est pas nécessairement connue par ses agents [154]. Ces approches sont généralement basées sur des techniques d'apprentissage par renforcement distribué, et pourraient bien s'intégrer dans une analyse du domaine de renforcement d'apprentissage des systèmes multi-agents.

Analyse de la variabilité d'agents mobiles

Les agents mobiles sont des agents qui ont la particularité d'être capables de décider quand et où déplacer leur code, état et données entre les différents périphériques connectés par le biais de réseau, et ce afin de reprendre leur dernière exécution au niveau de chaque nouveau périphérique. Cette caractéristique de mobilité, pourrait en effet être considérée comme une caractéristique optionnelle, et de ce fait, étendre le modèle de caractéristiques générique ; ce qui impliquerait la possibilité aux agents mobiles de substituer les appels de procédures à distance entre un client et un serveur, et d'exécuter directement leur code côté serveur. Cependant, l'analyse du domaine d'agents mobiles devrait couvrir également les différentes alternatives de ce paradigme tel que le fait d'exploiter d'autres modèles de communication.

Intégration de notre approche au sein d'une méthodologie de lignes de produits multi-agents

Bien que notre approche MAS-PL permette de concevoir et d'implémenter des familles multi-agents de façon indépendante du domaine d'application, elle n'a pas encore été intégrée au sein d'une méthodologie MAS-PL. Elle pourrait aussi bien être exploitée au sein de méthodes telles que GAIA-PL ou encore PASSI-PL que nous avons présentées dans la partie d'état de l'art de ce travail de thèse, mais également faire partie d'une nouvelle méthodologie indépendante des méthodes multi-agents existantes. La flexibilité et l'aspect modulable qu'offre notre approche au sein de son modèle de caractéristique générique ainsi que de son ensemble d'artefacts, pourrait en effet rendre plus facile ce type d'intégration. Il serait intéressant d'explorer cela en pratique. Par exemple, voir la possibilité de permettre aux concepteurs et développeurs de la ligne de produits multi-agents de paramétrer l'approche multi-agents qu'ils souhaiteraient utiliser, ainsi que l'ensemble des artefacts d'implémentation qu'ils désirent utiliser afin de générer l'ensemble

des étapes à suivre. Ce qui nous fait penser à l'approche MEDUSE [159] qui a guidé les étudiants durant l'évaluation prospective, et qui pourrait être exploitée à ce niveau afin de générer la méthodologie MAS-PL qui répond aux besoins des concepteurs et développeurs de la ligne de produits multi-agents.

Bibliographie

[1] Klaus Pohl, Günter Böckle, and Frank J van Der Linden. Software product line engineering : foundations, principles and techniques. Springer Science & Business Media, 2005.

[2] CharlesW Krueger. Easing the transition to software mass customization. In International Workshop on Software Product-Family Engineering, pages 282–293. Springer, 2001.

[3] Sven Apel, Don Batory, Christian Kästner, and Gunter Saake. Feature-Oriented Software Product Lines. Springer, 2013.

[4] Tewfik Ziadi and Jean-Marc Jézéquel. Product line engineering with the uml : deriving products, 2006.

[5] Andreas Metzger, Klaus Pohl, Patrick Heymans, Pierre-Yves Schobbens, and Germain Saval. Disambiguating the documentation of variability in software product lines : A separation of concerns, formalization and automated analysis. In 15th IEEE International Requirements Engineering Conference (RE 2007), pages 243–253. IEEE, 2007.

[6] Franck Fleurey, Øystein Haugen, Birger Møller-Pedersen, Gøran Klepp Olsen, Andreas Svendsen, and Xiaorui Zhang. A generic language and tool for variability modeling. 2009.

[7] Emmanuelle Rouillé. Gestion de la variabilité et automatisation des processus de développement logiciel. PhD thesis, Université Rennes 1, 2014.

[8] Sahid Mohd Zanes, Md Sultan Abu Bakar, Abdul Ghani Abdul Azim, and Baharom Salmi. Combinatorial interaction testing of software product lines : A mapping study. journal of Computer Science, 128(8) :379–398, 2016.

[9] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. Staged configuration through specialization and multilevel configuration of feature models. Software process : improvement and practice, 10(2) :143–169, 2005.

[10] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. Artificial intelligence : a modern approach, volume 2. Prentice hall Upper Saddle River, 2003.

[11] Danny Weyns, Elke Steegmans, and Tom Holvoet. Towards active perception in situated multi-agent systems. Applied Artificial Intelligence, 18(9-10) :867–883, 2004.

[12] Alessandro Ricci, Claudio Buda, and Nicola Zaghini. An agent-oriented programming model for soa & web services. In Industrial Informatics, 2007 5th IEEE International Conference on, volume 2, pages 1059–1064. IEEE, 2007.

[13] Stefania Bandini, Sara Manzoni, and Giuseppe Vizzari. Agent based modeling and simulation : an informatics perspective. Journal of Artificial Societies and Social Simulation, 12(4) :4, 2009.

[14] M. Reck. In Types of electronic auctions. the international conference on Information and communications technologies in tourism, 1994.

[15] Tarek Jarraya and Zahia Guessoum. Reuse interaction protocols to develop interactive agents. In Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology, pages 411–415. IEEE Computer Society, 2006.

[16] Danny Weyns, Andrea Omicini, and James Odell. Environment as a first class abstraction in multiagent systems. Autonomous agents and multi-agent systems, 14(1) :5–30, 2007.

[17] Jacques Ferber. Les systèmes multi-agents : Vers une intelligence collective, intereditions, 1995. Connaissances et compétences requises Cette thèse s' adresse à des étudiants ayant un profil Intelligence Artificielle/Base de Données et elle nécessite d'avoir des compétences théoriques dans les domaines, 1, 1995.

[18] Ioan Susnea and Cristian Axenie. Cognitive maps for indirect coordination of intelligent agents. Studies in Informatics and Control, 24(1) :111–118, 2015.

[19] Francis Heylighen. Stigmergy as a universal coordination mechanism : components, varieties and applications. Human Stigmergy : Theoretical Developments and New Applications ; Springer : New York, NY, USA, 2015.

[20] Jacques Ferber, Olivier Gutknecht, and Fabien Michel. From agents to organizations : an organizational view of multi-agent systems. In Agent-Oriented Software Engineering IV, pages 214–230. Springer, 2004.

[21] Jomi Fred Hübner, Jaime Simão Sichman, and Olivier Boissier. Moise+ : towards a structural, functional, and deontic model for mas organization. In Proceedings of the first international joint conference on Autonomous agents and multiagent systems : part 1, pages 501–502. ACM, 2002.

[22] Ingrid Nunes, Uirá Kulesza, Camila Nunes, Elder Cirilo, and Carlos Lucena. Extending PASSI to model multi-agent systems product lines. In Proceedings of the 2009 ACM symposium on Applied Computing, pages 729–730. ACM, 2009.

[23] Joaquin Peña, Michael G Hinchey, Manuel Resinas, Roy Sterritt, and James L Rash. Designing and managing evolving systems using a mas product line approach. Science of Computer Programming, 66(1) :71–86, 2007.

[24] Inmaculada Ayala, Mercedes Amor, Jose-Miguel Horcas, and Lidia Fuentes. A goal-driven software product line approach for evolving multi-agent systems in the internet of things. Knowledge-Based Systems, 184 :104883, 2019.

[25] Josh Dehlinger and Robyn R Lutz. A product-line approach to promote asset reuse in multi-agent systems, software engineering for multi-agent systems iv : research issues and practical applications, 2006.

[26] Zahia Guessoum, Massimo Cossentino, and Juan Pavón. Roadmap of agent-oriented software engineering. In Methodologies and software engineering for agent systems, pages 431–450. Springer, 2004.

[27] O Zohreh Akbari. A survey of agent-oriented software engineering paradigm : Towards its industrial acceptance. International Journal of Computer Engineering Research, 1(2) :14–28, 2010.

[28] Y Demazeau. La méthode voyelles, dans systèmes multi-agents. des théories organisationnelles aux applications industrielles. Hermès, Oslo, Norway, 2001.

[29] Yves Demazeau. From interactions to collective behaviour in agent-based systems. In In : Proceedings of the 1st. European Conference on Cognitive Science. Saint-Malo. Citeseer, 1995.

[30] Clements Paul and Northrop Linda. Software product lines : practices and patterns, 2001.

[31] David M Weiss et al. Software product-line engineering : a family-based software development process. 1999.

[32] Sven Apel, Christian Kästner, and Christian Lengauer. Language-independent and automated software composition : The featurehouse experience. IEEE Transactions on Software Engineering, 39(1) :63–79, 2013.

[33] Kyo C Kang, Sholom G Cohen, James A Hess, William E Novak, and A Spencer Peterson. Feature-oriented domain analysis (foda) feasibility study. Technical report, DTIC Document, 1990.

[34] David M Weiss et al. Software product-line engineering : a family-based software development process. 1999.

[35] Mikael Svahnberg, Jilles Van Gurp, and Jan Bosch. A taxonomy of variability realization techniques. Software : Practice and experience, 35(8) :705–754, 2005.

[36] Pamela Zave and Michael Jackson. Four dark corners of requirements engineering. ACM transactions on Software Engineering and Methodology (TOSEM), 6(1) :1–30, 1997.

[37] Klaus Schmid, Rick Rabiser, and Paul Grünbacher. A comparison of decision modeling approaches in product lines. In Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems, pages 119–126. ACM, 2011.

[38] Hassan Gomaa. Designing software product lines with UML. IEEE, 2005.

[39] Tewfik Ziadi, Loïc Hélouët, and Jean-Marc Jézéquel. Towards a uml profile for software product lines. PFE, 3 :129–139, 2003.

[40] S Buhne, Günter Halmans, and Klaus Pohl. Modeling dependencies between variation points in use case diagrams. In REFSQ, volume 3, pages 59–69. Citeseer, 2003.

[41] Günter Halmans and Klaus Pohl. Communicating the variability of a software-product family to customers. Informatik Forschung Und Entwicklung, 18(3-4) :113–131, 2004.

[42] Dániel Fey, Róbert Fajta, and András Boros. Feature modeling : A meta-model to enhance usability and usefulness. In International Conference on Software Product Lines, pages 198–216. Springer, 2002.

[43] Kyo C Kang, Jaejoon Lee, and Patrick Donohoe. Feature-oriented product line engineering. IEEE software, 19(4) :58–65, 2002.

[44] Øystein Haugen, Andrzej Wasowski, and Krzysztof Czarnecki. Cvl : common variability language. In SPLC (2), pages 266–267, 2012.

[45] Stan Buhne, Kim Lauenroth, and Klaus Pohl. Modelling requirements variability across product lines. In 13th IEEE International Conference on Requirements Engineering (RE'05), pages 41–50. IEEE, 2005.

[46] Lars Geyer and Martin Becker. On the influence of variabilities on the application-engineering process of a product family. In International Conference on Software Product Lines, pages 1–14. Springer, 2002.

[47] Fabricia Roos-Frantz, David Benavides, Antonio Ruiz-Cortés, André Heuer, and Kim Lauenroth. Quality-aware analysis in product line engineering with the orthogonal variability model. Software Quality Journal, 20(3) :519–565, 2012.

[48] Øystein Haugen, Birger Møller-Pedersen, Jon Oldevik, Gøran K Olsen, and Andreas Svendsen. Adding standardized variability to domain specific languages. In 2008 12th International Software Product Line Conference, pages 139–148. IEEE, 2008.

[49] Franck Fleurey, Øystein Haugen, Birger Møller-Pedersen, Andreas Svendsen, and Xiaorui Zhang. Standardizing variability–challenges and solutions. In International SDL Forum, pages 233–246. Springer, 2011.

[50] Krzysztof Czarnecki, Paul Grünbacher, Rick Rabiser, Klaus Schmid, and Andrzej Wąsowski. Cool features and tough decisions : a comparison of variability modeling ap-

proaches. In Proceedings of the sixth international workshop on variability modeling of software-intensive systems, pages 173–182, 2012.

[51] U Frank. Multilevel modeling-toward a new paradigm of conceptual modeling and information systems design. bus inf syst eng, 6 (6), 2014.

[52] Hua Wang. Multi-level requirement model and its implementation for medical device. PhD thesis, 2018.

[53] Christian Kästner and Sven Apel. Integrating compositional and annotative approaches for product line engineering. In Proc. GPCE Workshop on Modularization, Composition and Generative Techniques for Product Line Engineering, pages 35–40, 2008.

[54] F Benduhn, R Schröter, A Kenner, C Kruczek, T Leich, and G Saake. Migration from annotation-based to composition-based product lines : Towards a tool-driven process.

[55] Andrea Omicini, Alessandro Ricci, and Mirko Viroli. Artifacts in the a&a meta-model for multi-agent systems. Autonomous agents and multi-agent systems, 17(3) :432–456, 2008.

[56] Jacques Ferber. Multi-agent systems : an introduction to distributed artificial intelligence, volume 1. Addison-Wesley Reading, 1999.

[57] Juan Pavón, Jorge J Gómez-Sanz, and Rubén Fuentes. The ingenias methodology and tools. In Agent-oriented methodologies, pages 236–276. IGI Global, 2005.

[58] Michael Georgeff, Barney Pell, Martha Pollack, Milind Tambe, and Michael Wooldridge. The belief-desire-intention model of agency. In International Workshop on Agent Theories, Architectures, and Languages, pages 1–10. Springer, 1998.

[59] MV Dignum. A model for organizational interaction : based on agents, founded in logic. SIKS, 2004.

[60] Lin Padgham and Michael Winikoff. Prometheus : A methodology for developing intelligent agents. In Proceedings of the first international joint conference on Autonomous agents and multiagent systems : part 1, pages 37–38. ACM, 2002.

[61] Utku Görkem Ketenci, Jean-Michel Auberlet, Roland Brémond, and Emmanuelle Grislin-Le Strugeon. Improved road crossing behavior with active perception approach. In Proc. Transportation Research Board Annual Meeting, 2012.

[62] Raymond So and Liz Sonenberg. The roles of active perception in intelligent agent systems. In Pacific Rim International Workshop on Multi-Agents, pages 139–152. Springer, 2005.

[63] Niv Rafaeli and Gal A Kaminka. Active perception at the architecture level. In Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems,

pages 1708–1710. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[64] Danny Weyns and Tom Holvoet. A formal model for situated multi-agent systems. Fundamenta Informaticae, 63(2-3) :125–158, 2004.

[65] Stan Franklin and Art Graesser. Is it an agent, or just a program ? : A taxonomy for autonomous agents. In International Workshop on Agent Theories, Architectures, and Languages, pages 21–35. Springer, 1996.

[66] Stefania Bandini, Sara Manzoni, and Giuseppe Vizzari. A spatially dependent communication model for ubiquitous systems. In International Workshop on Environments for Multi-Agent Systems, pages 74–90. Springer, 2004.

[67] Franco Zambonelli, Nicholas R Jennings, and Michael Wooldridge. Developing multiagent systems : The gaia methodology. ACM Transactions on Software Engineering and Methodology (TOSEM), 12(3) :317–370, 2003.

[68] Pablo Noriega and Carles Sierra. Electronic institutions : Future trends and challenges. In International Workshop on Cooperative Information Agents, pages 14–17. Springer, 2002.

[69] Lev Semenovich Vygotsky. Mind in society : The development of higher psychological processes. Harvard university press, 1980.

[70] Robert Engelmore and Tony Morgan. Blackboard systems : edited by robert engelmore, tony morgan. Addison Wesley Publishing Company, 1988.

[71] Michael N Huhns and Larry M Stephens. Multiagent systems and societies of agents. Multiagent systems : a modern approach to distributed artificial intelligence, 1 :79–114, 1999.

[72] Daniel D Corkill. Collaborating software : Blackboard and multi-agent systems & the future. In Proceedings of the International Lisp Conference, volume 10, page 33, 2003.

[73] Rudy Rucker. A new kind of science, 2003.

[74] Jan Dijkstra, Harry JP Timmermans, and AJ Jessurun. A multi-agent cellular automata system for visualising simulated pedestrian activity. In Theory and Practical Issues on Cellular Automata, pages 29–36. Springer, 2001.

[75] ACL Fipa. Fipa acl message structure specification. Foundation for Intelligent Physical Agents, http ://www. fipa. org/specs/fipa00061/SC00061G. html (30.6. 2004), 2002.

[76] Massimo Cossentino. From requirements to code with the passi methodology. In Brian Henderson-Sellers and Paolo Giorgini, editors, Agent-Oriented Methodologies, chapter 4, pages 79–106. Idea Group Publishing, Melbourne, 2005.

[77] MV Dignum. A model for organizational interaction : based on agents, founded in logic. SIKS, 2004.

[78] Reid G Smith. The contract net protocol : High-level communication and control in a distributed problem solver. IEEE Transactions on computers, (12) :1104–1113, 1980.

[79] FIPA Fipa. specification part 2 : Agent communication language. Technical report, Technical report, FIPA-Foundation for Intelligent Physical Agents, 1997.

[80] Dimple Juneja, Ankit Jagga, and A Singh. A review of fipa standardized agent communication language and interaction protocols. Journal of Network Communications and Emerging Technologies, 5(2) :179–191, 2015.

[81] Lien F Lai. A knowledge engineering approach to knowledge management. Information Sciences, 177(19) :4072–4094, 2007.

[82] Robert Neches, Richard E Fikes, Tim Finin, Thomas Gruber, Ramesh Patil, Ted Senator, and William R Swartout. Enabling technology for knowledge sharing. AI magazine, 12(3) :36, 1991.

[83] Dieter Fensel. Ontology-based knowledge management. Computer, 35(11) :56–59, 2002.

[84] Tarek Jarraya and Zahia Guessoum. Towards a model driven process for multi-agent system. In International Central and Eastern European Conference on Multi-Agent Systems, pages 256–265. Springer, 2007.

[85] O Simonin. Le modèle satisfaction-altruisme : coopération et résolution de conflits entre agents réactifs, application à la robotique. PhD thesis, PhD thesis, UM2, 2001.

[86] Sana Moujahed. Approche multi-agents auto-organisée pour la résolution des contraintes spatiales dans les problèmes de positionnement mono et multi-niveaux. These de l'université de Technologie de Belfort-Montbéliard et de l'Université de Franche-Comté, 2007.

[87] Plerre-P Grassé. La reconstruction du nid et les coordinations interindividuelles chezbellicositermes natalensis etcubitermes sp. la théorie de la stigmergie : Essai d'interprétation du comportement des termites constructeurs. Insectes sociaux, 6(1) :41–80, 1959.

[88] H Van Parunak, Michael Purcell, and Robert O'Connell. Digital pheromones for autonomous coordination of swarming uav's. In 1st UAV Conference, page 3446, 2002.

[89] Massimo Cossentino, Nicolas Gaud, Vincent Hilaire, Stéphane Galland, and Abderrafiâa Koukam. Aspecs : an agent-oriented software process for engineering complex systems. Autonomous Agents and Multi-Agent Systems, 20(2) :260–304, 2010.

[90] Olivier Gutknecht and Jacques Ferber. The madkit agent platform architecture. In Workshop on Infrastructure for Scalable Multi-Agent Systems at the International Conference on Autonomous Agents, pages 48–55. Springer, 2000.

[91] Jacques Ferber and Olivier Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In Multi Agent Systems, 1998. Proceedings. International Conference on, pages 128–135. IEEE, 1998.

[92] Arthur Koestler. The ghost in the machine. 1967. London : Hutchinson, 1967.

[93] Ingrid Nunes, Carlos JP De Lucena, Donald Cowan, Uirá Kulesza, Paulo Alencar, and Camila Nunes. Developing multi-agent system product lines : from requirements to code. International Journal of Agent-Oriented Software Engineering, 4(4) :353–389, 2011.

[94] Jürgen Lind. Patterns in agent-oriented software engineering. In AOSE, volume 2, pages 47–58. Springer, 2002.

[95] Rosario Girardi. Reuse in agent-based application development. In Proc. 1st Int'l Workshop on Software Engineering for Large-Scale Multi-Agent Systems, 2002.

[96] Hideki Hara, Shigeru Fujita, and Kenji Sugawara. Reusable software components based on an agent model. In Parallel and Distributed Systems : Workshops, Seventh International Conference on, 2000, pages 447–452. IEEE, 2000.

[97] Tom Holvoet and Elke Steegmans. Application-specific reuse in multi-agent system development. In Proceedings of the International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (A. Garcia and C. de Lucena, eds.), pages 51–55, 2002.

[98] Luca Sabatucci, Massimo Cossentino, and Salvatore Gaglio. A semantic description for agent design patterns. In Proceedings of the Sixth International Workshop" From Agent Theory to Agent Implementation"(AT2AI-6) at The Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008), 2008.

[99] Cheah Wai Shiang, Fu Swee Tee, Alfian Abdul Halin, Ng Keng Yap, and Puah Chin Hong. Ontology reuse for multiagent system development through pattern classification. Software : Practice and Experience, 48(11) :1923–1939, 2018.

[100] Loh Chee Wyai, Cheah WaiShiang, and Marlene Valerie AiSiok Lu. Agent negotiation patterns for multi agent negotiation system. Advanced Science Letters, 24(2) :1464–1469, 2018.

[101] Josh Dehlinger and Robyn R Lutz. Gaia-pl : A product line engineering approach for efficiently designing multiagent systems. ACM Transactions on Software Engineering and Methodology (TOSEM), 20(4) :17, 2011.

[102] Alessandro Garcia, Cláudio Sant'Anna, Christina Chavez, Viviane Torres da Silva, Carlos JP de Lucena, and Arndt von Staa. Separation of concerns in multi-agent systems : An empirical study. In International Workshop on Software Engineering for Large-Scale Multi-agent Systems, pages 49–72. Springer, 2003.

[103] Josh Dehlinger and Robyn R Lutz. A product-line requirements approach to safe reuse in multi-agent systems. In ACM SIGSOFT Software Engineering Notes, volume 30, pages 1–7. ACM, 2005.

[104] Joaquín Peña Siles, Michael G Hinchey, Antonio Ruiz Cortés, and Pablo Trinidad Martín Arroyo. Building the core architecture of a nasa multiagent system product line. Lecture Notes in Computer Science, 4405, 208-224, 2007.

[105] Josh Dehlinger and Robyn R Lutz. Supporting requirements reuse in multi-agent system product line design and evolution. In Software Maintenance, 2008. ICSM 2008. IEEE International Conference on, pages 207–216. IEEE, 2008.

[106] Ingrid Nunes, Uirá Kulesza, Camila Nunes, and Carlos JP Lucena. A domain engineering process for developing multi-agent systems product lines. In Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2, pages 1339–1340. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[107] H. Gomaa. Designing Software Product Lines with UML : From Use Cases to Pattern-Based Software Architectures. Addison-Wesley, Reading, 2004.

[108] Ingrid Nunes, Elder Cirilo, and C Lucena. Developing a family of software agents with fine-grained variability : an exploratory study. In V Workshop on Software Engineering for Agent-oriented Systems, pages 71–82, 2009.

[109] Elder Cirilo, Ingrid Nunes, Uirá Kulesza, Camila Nunes, and Carlos Lucena. Automatic product derivation of multi-agent systems product lines. In Proceedings of the 2009 ACM symposium on Applied Computing, pages 731–732, 2009.

[110] Inmaculada Ayala, Jose Miguel Horcas, Mercedes Amor, and Lidia Fuentes. Using models at runtime to adapt self-managed agents for the iot. In German Conference on Multiagent System Technologies, pages 155–173. Springer, 2016.

[111] Inmaculada Ayala, Mercedes Amor, and Lidia Fuentes. The sol agent platform : Enabling group communication and interoperability of self-configuring agents in the internet of things. Journal of Ambient Intelligence and Smart Environments, 7(2) :243–269, 2015.

[112] Brian Henderson-Sellers. Agent-oriented methodologies. IGI Global, 2005.

[113] Piermarco Burrafato and Massimo Cossentino. Designing a multi-agent solution for a bookstore with the passi methodology. In AOIS@ CAiSE, 2002.

[114] Luca Cernuzzi, Thomas Juan, Leon Sterling, and Franco Zambonelli. The gaia methodology. In Methodologies and Software Engineering for Agent Systems, pages 69–88. Springer, 2004.

[115] Franco Zambonelli, Nicholas R Jennings, and Michael Wooldridge. Multi-agent systems as computational organizations : the gaia methodology. Agent-oriented methodologies, 6 :136–171, 2005.

[116] J Pena. On improving the modelling of complex acquaintance organisations of agents. A method fragment for the analysis phase. PhD thesis, PhD thesis, University of Seville, 2005.

[117] Viviane Torres Da Silva, Ricardo Choren, and Carlos JP De Lucena. Mas-ml : a multiagent system modelling language. International Journal of Agent-Oriented Software Engineering, 2(4) :382–421, 2008.

[118] Mark A Ardis and David M Weiss. Defining families : the commonality analysis (tutorial). In Proceedings of the 19th international conference on Software engineering, pages 649–650. ACM, 1997.

[119] Inmaculada Ayala, Mercedes Amor, Lidia Fuentes, and José M Troya. A software product line process to develop agents for the iot. Sensors, 15(7) :15640–15660, 2015.

[120] Inmaculada Ayala, Mercedes Amor, and Lidia Fuentes. Using spl to develop aal systems based on self-adaptive agents. In Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection, pages 263–275. Springer, 2016.

[121] Elder Cirilo, Uirá Kulesza, and Carlos José Pereira de Lucena. A product derivation tool based on model-driven techniques and annotations. J. UCS, 14(8) :1344–1367, 2008.

[122] Elder Cirilo, Ingrid Nunes, Uirá Kulesza, and Carlos Lucena. Automating the product derivation process of multi-agent systems product lines. Journal of Systems and Software, 85(2) :258–276, 2012.

[123] Alexander Pokahr, Lars Braubach, and Winfried Lamersdorf. Jadex : A bdi reasoning engine. Multi-agent programming, pages 149–174, 2005.

[124] Ingrid Nunes, Uirá Kulesza, Camila Nunes, and Carlos José Pereira de Lucena. Documenting and modeling multi-agent systems product lines. In SEKE, volume 2008, pages 745–751, 2008.

[125] Ingrid Nunes, Camila Nunes, Uirá Kulesza, and Carlos Lucena. Developing and evolving a multi-agent system product line : An exploratory study. In International Workshop on Agent-Oriented Software Engineering, pages 228–242. Springer, 2008.

[126] Joaquin Pena, Michael G Hinchey, Roy Sterritt, James L Rash, et al. Managing the evolution of an enterprise architecture using a mas-product-line approach. 2006.

[127] Dounia Boufedji, Zahia Guessoum, Anarosa Brandão, Tewfik Ziadi, and Aicha Mokhtari. Towards a mas product line engineering approach. In International Workshop on Engineering Multi-Agent Systems, pages 161–179. Springer, 2017.

[128] Anarosa Brandao, Dounia Boufedji, Tewfik Ziadi, and Zahia Guessoum. Vers une approche d'ingénierie multiagent à base de ligne de produits logiciels. In 23es Journées Francophones sur les Systèmes Multi-Agents (JFSMA'15), pages 49–58. Cépaduès, 2015.

[129] Tristan Behrens. Towards Building Blocks for Agent-Oriented Programming. PhD thesis, Clausthal University of Technology, 2012.

[130] Jacques Ferber, Fabien Michel, and José Báez. Agre : Integrating environments with organizations. In International Workshop on Environments for Multi-Agent Systems, pages 48–56. Springer, 2004.

[131] Christian Kastner, Thomas Thum, Gunter Saake, Janet Feigenspan, Thomas Leich, Fabian Wielgorz, and Sven Apel. Featureide : A tool framework for feature-oriented software development. In 2009 IEEE 31st International Conference on Software Engineering, pages 611–614. IEEE, 2009.

[132] Olivier Boissier, Rafael H Bordini, Jomi F Hübner, Alessandro Ricci, and Andrea Santi. Multi-agent oriented programming with jacamo. Science of Computer Programming, 78(6) :747–761, 2013.

[133] Olivier Boissier, Jomi F Hübner, and Alessandro Ricci. The jacamo framework. In Social coordination frameworks for social technical systems, pages 125–151. Springer, 2016.

[134] Alessandro Ricci, Michele Piunti, and Mirko Viroli. Environment programming in multi-agent systems : an artifact-based perspective. Autonomous Agents and Multi-Agent Systems, 23(2) :158–192, 2011.

[135] Rafael H Bordini, Jomi Fred Hübner, and Michael Wooldridge. Programming multi-agent systems in AgentSpeak using Jason, volume 8. John Wiley & Sons, 2007.

[136] Jomi F Hübner, Olivier Boissier, and Rafael H Bordini. From organisation specification to normative programming in multi-agent organisations. In International Workshop on Computational Logic in Multi-Agent Systems, pages 117–134. Springer, 2010.

[137] Mehdi Dastani. 2apl : a practical agent programming language. Autonomous agents and multi-agent systems, 16(3) :214–248, 2008.

[138] Frank S de Boer, Koen V. Hindriks, Wiebe van der Hoek, and J-J Ch Meyer. Agent programming with declarative goals. arXiv preprint cs/0207008, 2002.

[139] Eric Sun. The effects of auctions parameters on price dispersion and bidder entry on ebay : a conditional logit analysis. Bachelor Thesis, 2005.

[140] Preetinder Kaur, Madhu Goyal, and Jie Lu. A comparison of bidding strategies for online auctions using fuzzy reasoning and negotiation decision functions. IEEE Transactions on Fuzzy Systems, 25(2) :425–438, 2017.

[141] Ravi Bapna, Paulo Goes, Alok Gupta, and Yiwei Jin. User heterogeneity and its impact on electronic auction market design : An empirical exploration. Mis Quarterly, pages 21–43, 2004.

[142] Jarrod Trevathan and Wayne Read. Detecting shill bidding in online english auctions. Handbook of research on social and organizational liabilities in information security, 46 :446–470, 2009.

[143] Patricia Anthony and Nicholas R Jennings. Developing a bidding agent for multiple heterogeneous auctions. ACM Transactions on Internet Technology (TOIT), 3(3) :185–217, 2003.

[144] Olivier Gutknecht, Jacques Ferber, and Fabien Michel. MadKit : une architecture de plate-forme multi-agent générique. PhD thesis, Lirmm, University of Montpellier, 2000.

[145] Scott A DeLoach. The mase methodology. In Methodologies and software engineering for agent systems, pages 107–125. Springer, 2004.

[146] Juan M Corchado and Rosalía Laza. Constructing deliberative agents with case-based reasoning technology. International Journal of Intelligent Systems, 18(12) :1227–1241, 2003.

[147] M Glez-Bedia, JM Corchado, ES Corchado, and C Fyfe. Analytical model for constructing deliberative agents. Engineering Intelligent Systems for Electrical Engineering and Communications, 10(3) :173–185, 2002.

[148] Javier Bajo, Dante I Tapia, Ana de Luis, Sara Rodríguez, Juan F de Paz, and Juan M Corchado. Hybrid architecture for a reasoning planner agent. In International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, pages 461–468. Springer, 2007.

[149] Jaime G Carbonell. Learning by analogy : Formulating and generalizing plans from past experience. In Machine learning, pages 137–161. Springer, 1983.

[150] Carole Bernon, Marie Pierre Gleizes, Gauthier Picard, and Pierre Glize. The adelfe methodology for an intranet system design. AOIS@ CAiSE, 57, 2002.

[151] Fausto Giunchiglia, John Mylopoulos, and Anna Perini. The tropos software development methodology : processes, models and diagrams. In International Workshop on Agent-Oriented Software Engineering, pages 162–173. Springer, 2002.

[152] Andrea Omicini. Soda : Societies and infrastructures in the analysis and design of agent-based systems. In International Workshop on Agent-Oriented Software Engineering, pages 185–193. Springer, 2000.

[153] Nicholas R Jennings, Peyman Faratin, Alessio R Lomuscio, Simon Parsons, Michael J Wooldridge, and Carles Sierra. Automated negotiation : prospects, methods and challenges. Group Decision and Negotiation, 10(2) :199–215, 2001.

[154] Juan C Burguillo. Self-organizing coalitions for managing complexity.

[155] Regina Frei and Giovanna Di Marzo Serugendo. Self-organizing assembly systems. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 41(6) :885–897, 2011.

[156] Huaglory Tianfield and Rainer Unland. Towards self-organization in multi-agent systems and grid computing. Multiagent and Grid Systems, 1(2) :89–95, 2005.

[157] Jomi F Hubner, Jaime S Sichman, and Olivier Boissier. Developing organised multiagent systems using the moise+ model : programming issues at the system and agent levels. International Journal of Agent-Oriented Software Engineering, 1(3-4) :370–395, 2007.

[158] Vincent Hilaire, Abder Koukam, Pablo Gruer, and Jean-Pierre Müller. Formal specification and prototyping of multi-agent systems. In International Workshop on Engineering Societies in the Agents World, pages 114–127. Springer, 2000.

[159] Sara Casare, Tewfik Ziadi, Anarosa Alves Franco Brandão, and Zahia Guessoum. Meduse : an approach for tailoring software development process. In Engineering of Complex Computer Systems (ICECCS), 2016 21st International Conference on, pages 197–200. IEEE, 2016.

[160] M P Gleizes C. Bernon, P. Glize, and G. Picard. Le problème de l'emploi du temps cahier des charges. In Ingénierie des AMAS pour l'emploi du temps. ETTO Emergent TimeTableOrganization. ASA-PRC IA, IRIT, 2002.

[161] Tristan Behrens, Mehdi Dastani, Jürgen Dix, Michael Köster, and Peter Novák. The multi-agent programming contest from 2005–2010. Annals of Mathematics and Artificial Intelligence, 59(3) :277–311, 2010.