L'ingénierie de l'alignement : Concepts, Modèles et Processus.La méthode ACEM pour la correction et l'évolution d'un système d'information aux processus d'entreprise

Anne Etien

THESE DE DOCTORAT
DE L'UNIVERSITE PARIS I – PANTHEON - SORBONNE

Spécialité : Informatique

Anne Etien

Pour l'obtention du titre de :

DOCTEUR DE L'UNIVERSITE PARIS I – PANTHEON - SORBONNE

Ingénierie de l'alignement : Concepts, Modèles et Processus

La méthode ACEM pour l'alignement d'un système d'information aux processus d'entreprise

Soutenue le 13 mars 2006 devant le jury composé de

Mme Colette ROLLAND	Directeur de thèse
M. Camille SALINESI	Co-directeur de thèse
M. Neil A. M. MAIDEN	Rapporteur
M. Michel LEONARD	Rapporteur
Mme Françoise GIRE	Membre du jury
M. Jacky AKOKA	Membre du jury

REMERCIEMENTS

Je voudrais tout d'abord exprimer mes vifs remerciements à Colette Rolland, Professeur à l'Université de Paris 1 Panthéon - Sorbonne pour la confiance qu'elle m'a témoignée en m'accueillant dans son équipe et en acceptant la direction scientifique de mes travaux. Je lui suis reconnaissante de m'avoir fait bénéficier tout au long de ce travail de sa grande compétence, de sa rigueur intellectuelle, de son efficacité et de ses précieux conseils.

Je remercie vivement Camille Salinesi, Maître de Conférences à l'Université de Paris 1 Panthéon - Sorbonne pour sa disponibilité, ses conseils et son soutien pendant ces trois années.

Je remercie sincèrement Michel Léonard, Professeur à l'Université de Genève, et Neil Maiden, Professeur à l'Université de la City à Londres, qui me font eu la gentillesse d'accepter les rôles de rapporteurs.

Je remercie également Françoise Gire, Professeur à l'Université de Paris 1 Panthéon - Sorbonne et Jacky Akoka, Professeur titulaire de la chaire d'informatique d'entreprise du CNAM pour avoir accepté de faire partie du jury de cete thèse.

Je remercie tous les membres de l'équipe du Centre de Recherche en Informatique, et tout particulièrement Rim, Ines, Elena I, Elena K, Deniz, Emmanuel, Ramzi, Clotilde, Iyad, Sondes, Souad, German, Marc-Henri, Fabienne, Sandie, Farida, Selmin, Carine, Daniel, pour leur collaboration, leur gentillesse et leur soutien.

Je remercie tout particulièrement Laure-Hélène et Rébecca pour leur amitié, leur soutien moral et leurs encouragements.

Je suis très reconnaissante à Brigitte, Claire, Fabienne, Laure-Hélène, Matthieu, Rébecca et Rim qui ont eu le courage et la patience de relire mon mémoire.

Je tiens enfin à remercier mes parents pour leur soutien sans faille, Claire, Guillaume et Laure, pour leur complicité fraternelle et tout particulièrement Matthieu pour sa disponibilité, son soutien au quotidien, ses encouragements et son indéfectible patience.

RESUME

Les organisations subissent de fréquents changements. Pour rester compétitives, leurs processus d'entreprise et leur système d'information doivent évoluer de concert, ce qui n'est pas aisé. Bien souvent, l'évolution se fait de manière séparée voire divergente. Or, une rupture de la relation d'alignement entre système et processus entraîne une baisse de la performance de l'organisation. L'organisation a donc besoin de maîtriser l'évolution conjointe du système et des processus, ce qui suppose de savoir si le système et les processus gèrent la même information ou permettent d'atteindre les mêmes buts, pour pouvoir ensuite corriger le système ou les processus. Les directeurs de systèmes d'information considèrent le problème de l'alignement comme leur priorité absolue mais ils admettent qu'il reste encore mal posé et, a fortiori, mal résolu.

De nombreux chercheurs s'intéressent au problème de l'alignement, le plus souvent sous l'angle de l'alignement de la politique du développement du système d'information sur la politique de développement de l'entreprise (alignement stratégique). Nous proposons de nous écarter de cette vision réductrice en abordant le problème de l'alignement, de façon rigoureuse et formelle, dans un contexte d'évolution qui correspond à la réalité quotidienne des organisations et permet de mieux comprendre les enjeux et les concepts de l'ingénierie de l'alignement.

Cette thèse apporte des réponses au problème de l'alignement et de son maintien au cours du temps. Nous définissons précisément et formellement les concepts nécessaires à la mesure, à la correction et à l'évolution de l'alignement entre un système et des processus d'entreprise.

Ainsi, nous proposons dix métriques permettant d'évaluer différents aspects complémentaires de la relation d'alignement. Ces métriques sont bâties sur l'utilisation de modèles pour représenter le système et les processus d'entreprise et la définition de deux types de liens entre les concepts de ces modèles.

Nous proposons également une méthode, la méthode ACEM *(Alignment Correction and Evolution Method)*, pour corriger l'alignement et faire évoluer conjointement le système et les processus d'entreprise. Cette méthode matérialise la relation d'alignement à un niveau intentionnel par le biais d'un modèle pivot. Elle guide les ingénieurs d'alignement dans la réalisation des différentes étapes du processus tout en leur laissant une grande liberté de choix. L'ingénieur fait évoluer le modèle pivot en exprimant explicitement les exigences d'évolution sous forme d'opérateurs d'écart.

Ces métriques et ces opérateurs doivent pouvoir être utilisés dans des contextes variés. C'est pourquoi nous les avons définis à un niveau générique, indépendamment de tout méta-modèle spécifique. Des processus permettent de les générer de façon rigoureuse et aisée pour des méta-modèles particuliers sans faire appel aux connaissances des ingénieurs ou aux particularités d'un projet ou d'une méthode.

ABSTRACT

The organisations frequently change. In order to remain competitive, their business processes and their system have to evolve together what is not easy. Evolution is often performed from a separate or event divergent manner. However, a rupture of the alignment relationship between the system and the business processes leads to a decrease of the business performance. The organisation thus need to master the joint evolution of the system and the business processes what supposes to know if the system and the business processes manage the same information or allow to reach the same goals in order to then correct the system or the processes.

Researchers are interested in alignment issue most often by aligning IT strategy to business strategy. We propose to move away from this simplistic vision by formally and rigorously tackling the alignment issue, in an evolution context corresponding to the business daily reality and allowing to better understand stakes and concepts of Alignment Engineering.

This thesis brings answers to the alignment issue and to its maintain during the time. We precisely and formally define concepts necessary to the measure, the correction and the evolution of alignment between a system and business processes.

Thus we propose ten metrics allowing to evaluate different and complementary aspects of the alignment relationship. These metrics rely on (i) the use of models to represent the system and the business processes and (ii) the definition of two links type between concepts of these models.

We also propose a method, the ACEM method *(Alignment Correction and Evolution Method)*, to correct alignment and to make jointly evolve the system and business processes. This method materialises the alignment relationship through a pivot model. It guides the alignment engineers in the enactment of the different process steps while leaving them a large choice freedom. The engineer makes evolve the pivot model by explicitly expressing evolution requirements under the form of gaps.

These gaps operators and these metrics that we defined, can be used in other contexts. That why, we define them at a generic level, independently of any specific meta-model. Processes allow to easily and rigorously generate them for particular meta-models without using engineer knowledge or special features of a project or a method.

TABLE DES MATIÈRES

CHAPITRE 1 : INTRODUCTION

1. Contexte de la thèse

Aligner le système d'information à la stratégie d'entreprise est depuis plusieurs années la priorité absolue des directeurs des systèmes d'information [Luftman04], [Reich03]. Selon plusieurs auteurs [Chan97], [Croteau01], [Tallon02] cet alignement accroît la performance de l'organisation. En outre, lorsque l'alignement est mauvais, les processus d'entreprises ne profitent pas pleinement des moyens technologiques investis [Henderson93]. Même si on est capable de construire d'un côté, des systèmes performants et d'un autre côté, des processus d'entreprise répondant aux exigences des décideurs, il est indispensable pour le bon fonctionnement et la performance de l'organisation que ses systèmes prennent parfaitement en charge ses processus. De plus, l'alignement entre système d'information et processus d'entreprise facilite le changement en diminuant les coûts et le temps d'adaptation mais aussi en servant de ligne directrice dans la mise en œuvre du changement. En effet, les changements influencent souvent l'organisation dans sa globalité, des processus d'entreprise au système d'information. Or, il est important pour les organisations, si elles veulent rester compétitives, de réagir vite et avec souplesse aux changements des besoins de leurs clients ou des buts de l'organisation.

Si l'intérêt de l'alignement est largement reconnu, sa mise en œuvre reste trop souvent limitée. Peu de dirigeants considèrent que le système et les processus de leur organisation sont alignés [IBM03]. [Luftman04] identifie deux causes principales : (i) les acteurs de l'organisation ne savent pas ce qu'est l'alignement et (ii) il existe une absence de communication et de compréhension entre le monde du "business" et celui des technologies de l'information.

Afin de répondre aux attentes des praticiens, un sous-domaine de l'ingénierie des systèmes d'information émerge : celui de l'ingénierie de l'alignement.

Un nombre croissant de recherches s'intéressent à l'alignement [Henderson93], [Luftman04], [Regev04], [Wegmann05], [Krishna04], [Salinesi03], [Bodhuin04]. De nombreux workshops ont été consacrés à ce thème de recherche [BPMDS04], [REBNITA05]. Les auteurs proposent des méthodes et des outils propres à la gestion des problèmes liés à l'alignement, par exemple pour construire un système et des processus d'entreprise alignés, ou pour l'évolution de la relation d'alignement au cours du temps.

Il semble que l'on puisse parler *d'ingénierie de l'alignement* comme sous-domaine de l'ingénierie des systèmes d'information, au même titre que l'ingénierie des méthodes ou de l'ingénierie des exigences. Nous pensons que l'ingénierie de l'alignement peut et doit devenir une discipline à part entière, compte tenu de l'importance de l'alignement pour les chercheurs, les utilisateurs et les industriels.

L'ingénierie de l'alignement se définit comme l'activité qui consiste à construire, rétablir et faire évoluer l'alignement entre plusieurs entités. Le plus souvent, la relation d'alignement concerne deux entités. Cependant, certaines approches, en particulier les approches d'urbanisme, étudient l'alignement entre plus de deux entités (par exemple cinq dans [Wieringa02]). L'une de ces entités correspond souvent aux stratégies des technologies de l'information, au système ou au logiciel.

L'ingénierie de l'alignement s'intéresse à des problèmes spécifiques, différents de ceux des autres sous-domaines de l'ingénierie des systèmes d'information. La notion même d'alignement implique l'existence de plusieurs entités liées entres elles. Les ingénieurs doivent gérer ensemble ces entités : lors de leur construction, lors de la gestion quotidienne (pour éventuellement réajuster cette relation), lors de projets d'évolution ou tout simplement pour les représenter. L'ingénierie de l'alignement doit donc trouver sa propre autonomie en proposant des outils et des méthodes pour étudier l'alignement sous tous ces aspects, tels que la définition, la représentation, la construction, ou l'évolution.

L'ingénierie de l'alignement doit s'enrichir des autres disciplines déjà existantes, en particulier l'ingénierie des exigences. L'ingénierie des exigences établit une relation entre ce que le système doit faire (le *quoi*, décrit par des spécifications) et ce pourquoi il doit le faire (le *pourquoi* qui traduit les attentes des utilisateurs et des décideurs). L'ingénierie des exigences permet de faire correspondre les fonctionnalités du système et les besoins organisationnels.

Cette thèse s'inscrit dans la discipline de l'ingénierie de l'alignement qu'elle considère dans la perspective de l'ingénierie des exigences.

2. Problématique

Nous nous intéressons plus particulièrement à l'alignement entre deux entités : un processus d'entreprise et un système d'information. Dans la suite, nous utiliserons les termes processus et système.

Considérer l'ingénierie de l'alignement comme une activité à part entière nécessite (i) une définition claire de la notion d'alignement, (ii) des langages pour la représenter, (iii) des méthodes pour gérer l'alignement (c'est-à-dire construire des processus et les systèmes qui les supportent de façon alignée, permettre l'évolution de ces entités sans rompre l'alignement…)

Nous avons identifié six problèmes répartis en deux grands groupes. Le premier groupe relève du concept d'alignement lui même. Il compte trois problèmes :

- l'absence de définition précise de l'alignement ;
- la difficulté de distinguer l'alignement du non alignement ; et
- l'absence de représentation de l'alignement.

Le deuxième groupe regroupe des problèmes liés à l'ingénierie de l'alignement :

- la difficulté de construire des entités alignées ;
- la difficulté de rétablir l'alignement ; et
- l'absence de méthode de maintien de l'alignement face aux changements.

Nous considérons chacune de ces problématiques dans les sous-sections suivantes.

2.1. L'absence de définition précise de l'alignement

Il n'existe pas de définition communément acceptée de l'alignement. En 1989, Henderson considérait que les caractéristiques du concept d'alignement n'étaient pas clairement définies [Henderson89]. Quinze ans plus tard, en 2004, Avison estime que la notion d'alignement reste floue, tant au niveau stratégique (entre stratégies d'entreprise et stratégies des technologies de l'information) qu'au niveau opérationnel (entre système et processus) [Avison04].

L'absence de définition précise de ce concept explique l'absence de consensus sur le terme utilisé. Plus d'une dizaine de termes sont utilisés dans la littérature pour définir la relation de correspondance entre le système et les processus [Regev04]. Ainsi, on nomme ce concept aussi bien : "fit" [Porter96], "fitness relationship" [Potts97] et "matching" (qui sont des expressions anglaises difficilement traduisibles en français), que "alignement" [Wegmann05b], "intégration" [Weill98], "pont" [Ciborra97], "harmonie" [Luftman96] ou "fusion" [Smaczny01], "lien" [Henderson93], mais aussi "correspondance" [Knoll94] ou "congruence" [Bergeron99].

2.2. La difficulté de distinguer l'alignement du non alignement

Les chercheurs, les dirigeants d'entreprise, les consultants et les utilisateurs s'accordent à dire qu'une rupture de l'alignement entraîne une baisse de la performance de l'organisation [IBM03], [Henderson89]. L'état actuel de la recherche sur l'alignement ne permet pas de déterminer pour un projet donné, à un instant précis, si les processus et le système sont alignés. Pourtant, cette information pourrait (i) servir aux décideurs de critère de décision au même titre que le coût ou la flexibilité pour choisir parmi les cibles alternatives de conception ou (ii) les guider dans l'obtention d'une telle situation.

2.3. L'absence de représentation de l'alignement

Très peu d'approches considèrent la relation d'alignement comme un concept en soi [Salinesi03]. Elle est donc rarement représentée. Cette absence de représentation de la relation d'alignement montre (i) d'un point de vue conceptuel que les chercheurs ne pensent pas en termes d'ingénierie de l'alignement et (ii) d'un point de vue pragmatique, que les modèles de processus et les modèles du système sont traditionnellement exprimés dans des langages différents, à des niveaux d'abstraction distincts et dans des documents séparés. Les modèles de processus utilisent des concepts tel que but, processus, acteur et rôle, alors que les modèles du système décrivent des objets, des opérations, des événements, etc. Les fonctionnalités du système sont spécifiées de façon précise et détaillée à un bas niveau d'abstraction, le niveau opérationnel. Les processus sont représentés à un haut niveau d'abstraction, le niveau intentionnel.

Sans moyen pour représenter l'alignement, il est difficile de raisonner sur ce concept, de construire ou de faire évoluer un système et un processus alignés.

2.4. La difficulté de construire des processus et un système alignés.

De nombreux directeurs des systèmes d'information considèrent que la stratégie d'entreprise et la stratégie relative aux technologies de l'information de leur organisation ne font qu'un. Pourtant, ils sont également nombreux à considérer qu'au niveau opérationnel, systèmes et processus ne sont pas alignés [IBM03]. Ceci relève pour partie du fait que ces deux entités n'ont pas été construites de manière alignée.

La difficulté de construire un système et un processus alignés relève en grande partie de l'utilisation de langages, de niveaux d'abstraction et de documents différents pour modéliser chacune de ces deux entités [Salinesi03].

2.5. La difficulté de rétablir l'alignement

On constate fréquemment qu'au cours de leurs cycles de vie, le système et les processus ne sont plus alignés. Cela arrive par exemple, quand les processus s'adaptent pour répondre aux objectifs des dirigeants, mais ne sont plus gérés convenablement par le système. Il est donc nécessaire de rétablir l'alignement en modifiant le système et éventuellement les processus afin de mieux faire correspondre ces deux entités. Peu d'approches proposent de guider les décideurs, les experts systèmes et les utilisateurs dans une telle démarche.

2.6. L'absence de méthode de maintien de l'alignement face aux changements

Les entreprises sont de plus en plus soumises à des changements. Si elles veulent rester compétitives, elles doivent faire évoluer leur système et/ou leur processus pour atteindre une nouvelle situation de synergie, où ces deux entités sont alignées. Maintenir l'alignement dans un contexte d'évolution suppose (i) d'identifier l'impact des changements d'une entité sur l'autre, (ii) de savoir propager ces changements et (iii) de déterminer quand a lieu cette propagation. En d'autres termes, l'évolutivité de l'alignement impose d'être capable de faire évoluer conjointement le système d'information et les processus.

Cette thèse vise à traiter les six problèmes énoncés ci-dessus. Comme le montre la section suivante, la solution apportée s'articule autour de la mesure et de l'ingénierie de l'alignement.

3. Définition, représentation, mesure, adaptation et évolution de la relation d'alignement

Afin de répondre à ces différents problèmes, nous proposons une approche de correction et d'évolution de l'alignement entre un système d'information et des processus d'entreprise. Cette approche nécessite une définition précise et formelle du concept d'alignement.

Nous proposons une *définition* de l'alignement qui s'appuie sur les principes de la modélisation et de la méta-modélisation et s'inscrit dans la lignée des travaux de l'OMG (Object Management Group) [OMG] sur UML [UML] et le MOF [MOF]. Cette définition :

1. est établie de façon générique, indépendamment de tout projet et de tout modèle spécifique, entre les éléments de deux méta-modèles ;
2. s'appuie sur trois éléments clé : les concepts nécessaires à la modélisation des processus (les éléments du méta-modèle de processus), ceux permettant de représenter le système (les éléments du méta-modèle du système) et des liens types formellement définis entre ces concepts.

En s'appuyant sur les concepts des différents méta-modèles, cette définition permet de présenter différents aspects de la relation d'alignement selon la nature des concepts mis en relation, par exemple entre les buts, les besoins ou la structure des composants des processus et du système.

Afin de corriger l'alignement entre un système et des processus, il est nécessaire de distinguer l'alignement du non-alignement dans une situation donnée. En effet, le système et les processus ne sont, le plus souvent, que partiellement alignés. Il existe toute une gamme de possibilités entre "parfaitement" et "pas du tout" aligné. Pour saisir ces nuances, nous proposons un *ensemble de métriques*. Cet ensemble structuré comprend dix métriques reposant sur les trois éléments clé de la définition de la relation d'alignement. Chaque métrique mesure plus particulièrement un aspect de la relation d'alignement et se présente sous la forme d'un ratio qui prend les concepts des processus en référence (dénominateur) et les compare à ceux pris en charge par le système (numérateur). Ainsi, plus la valeur obtenue pour chaque métrique est élevée, plus le système et les processus sont alignés. Ces métriques permettent donc d'évaluer le degré d'alignement

Une valeur de métrique inférieure à 1 signifie que des actions correctives doivent être entreprises sur les différents éléments impliqués dans la métrique. Il est possible d'exprimer ces actions correctives sous forme d'exigences d'évolution traduisant une volonté d'adaptation pour rétablir l'alignement.

Pour rétablir l'alignement, on peut agir de deux façons : (i) sur les processus en recherchant si les éléments non supportés par le système doivent être conservés dans les processus ou (ii) sur le système afin qu'il prenne en charge des éléments des processus qui ne l'étaient pas jusqu'alors. On peut entreprendre conjointement ces deux types de correction.

Nous proposons d'utiliser un *modèle pivot* pour représenter de façon unifiée les processus et le système et matérialiser la relation d'alignement. Le méta-modèle pivot utilise le méta-modèle de Carte [Rolland01] orienté-but et l'enrichit afin de représenter les objets métiers et les objets manipulés par le système. Comme le montre la Figure 1, le modèle pivot permet de réaliser un couplage direct entre les objectifs de l'entreprise et les fonctionnalités du système tous deux définis au niveau opérationnel.

Grâce au mécanisme d'affinement, la relation d'alignement est représentée non pas de façon plate, mais à différents niveaux de détails.

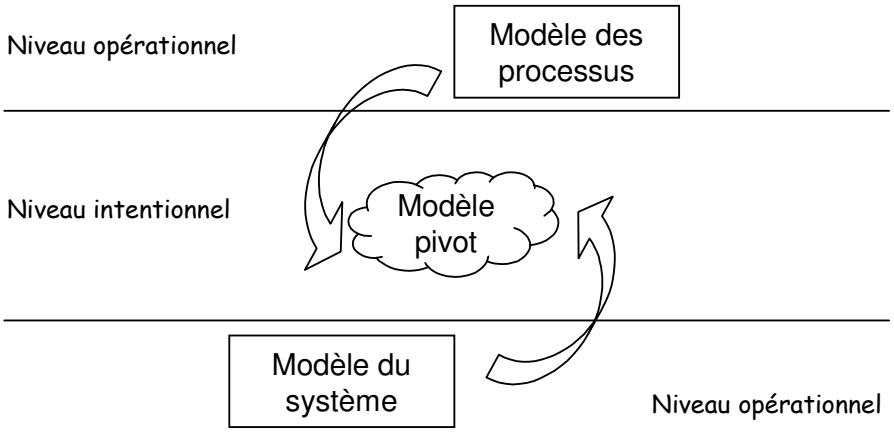


Figure 1 : Représentation de l'alignement

Le maintien de la relation d'alignement nécessite de savoir répercuter les changements des processus sur le système (et vice-versa), et donc de gérer l'évolution. Notre expérience dans des projets industriels nous amène à penser que, pour pouvoir gérer correctement l'évolution, il faut en premier lieu savoir exprimer explicitement les exigences d'évolution. Une telle approche doit se focaliser sur ce qui change en évitant de décrire ce qui reste inchangé. Il est en effet souvent plus facile pour les parties prenantes d'indiquer ce qui doit changer entre les versions du système ou des processus plutôt que de définir complètement le système ou les processus cibles à atteindre. Les exigences d'évolution peuvent ainsi être efficacement exprimées par des *écarts*, matérialisés au moyen d'opérateurs, entre deux versions de modèles.

Ainsi, pour faire évoluer simultanément le modèle de processus et le modèle du système, nous proposons de spécifier des exigences d'évolution sur le modèle pivot, comme le montre la Figure 2.

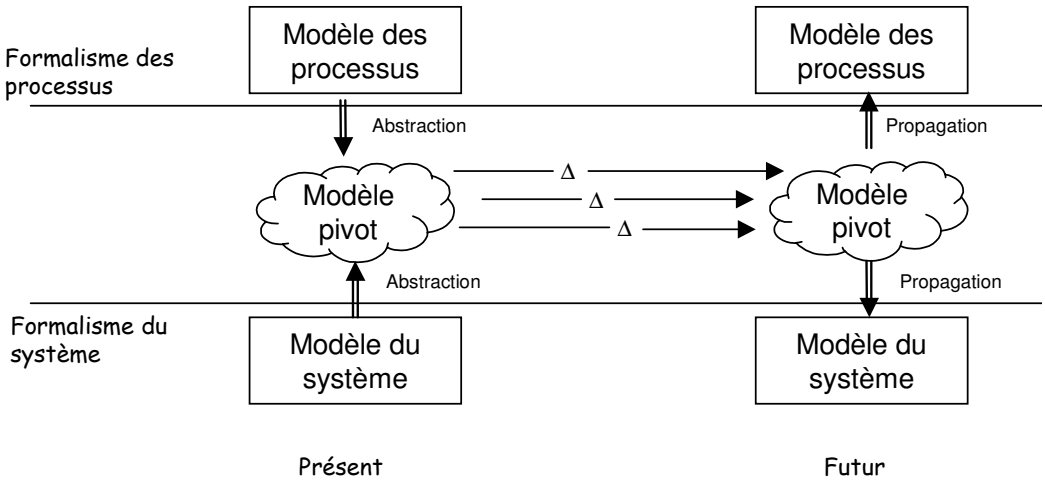


Figure 2 : Schéma de notre approche de co-évolution

Les écarts identifiés sur le modèle pivot (représentés par la lettre grecque Δ à la Figure 2) sont ensuite propagés sur les modèles du système et des processus. Cette approche exploite les liens entre les éléments du modèle pivot et ceux du modèle de processus (respectivement du système).

4. Contributions

Cette thèse présente les résultats suivants :

- une définition formelle de la relation d'alignement entre un système d'information et des processus.
- un ensemble de métriques génériques rassemblées au sein d'un cadre et un processus d'adaptation à des modèles spécifiques. Ces métriques s'intéressent à différents aspects de l'alignement et permettent d'évaluer le degré d'alignement dans un spectre de nuances entre parfaitement et pas du tout aligné.
- l'utilisation d'un modèle pivot pour matérialiser l'alignement et représenter conjointement le système et les processus. Ce modèle permet aux experts métiers et aux ingénieurs de communiquer et de se comprendre en adoptant le même langage. Grâce au modèle pivot, notre approche permet : (i) de faire évoluer des processus et le système associé même quand les modèles de la situation présente sont inexploitables (en definissant les besoins d'alignement relatifs à cette situation) et (ii) de construire des processus d'entreprise et un système alignés.
- la spécification des exigences d'évolution matérialisées par des opérateurs d'écart. Ces opérateurs sont organisés au sein d'une typologie et sont définis de façon générique. Un processus permet de les instancier pour un méta-modèle spécifique.
- une proposition de démarche formellement définie pour le guidage de la co-évolution. Afin d'aider les décideurs à définir les besoins d'alignement, à déterminer les actions correctives à mettre en place et/ou à faire co-évoluer le système et les processus, nous proposons une démarche définie formellement : la méthode ACEM (**A**lignment **C**orrection and **E**volution **M**ethod). Cette méthode est composée de trois étapes correspondant à : (1) la construction du modèle pivot ; (2) la découverte d'exigences d'évolution correspondant au rétablissement de l'alignement ou à une réelle volonté de changement et (3) la propagation de ces évolutions sur les modèles de système et de processus.

Comme le montre le schéma présenté Figure 3, la démarche consomme des modèles des processus et du système dans sa situation présente, et produit les modèles de processus et du système dans la situation future. La démarche est en trois étapes centrées sur le modèle pivot : (i) construire le modèle afin de matérialiser la relation d'alignement, (ii), le faire évoluer afin de définir les besoins de changement, et (iii) propager les évolutions définies afin de spécifier les modèles de la situation future.

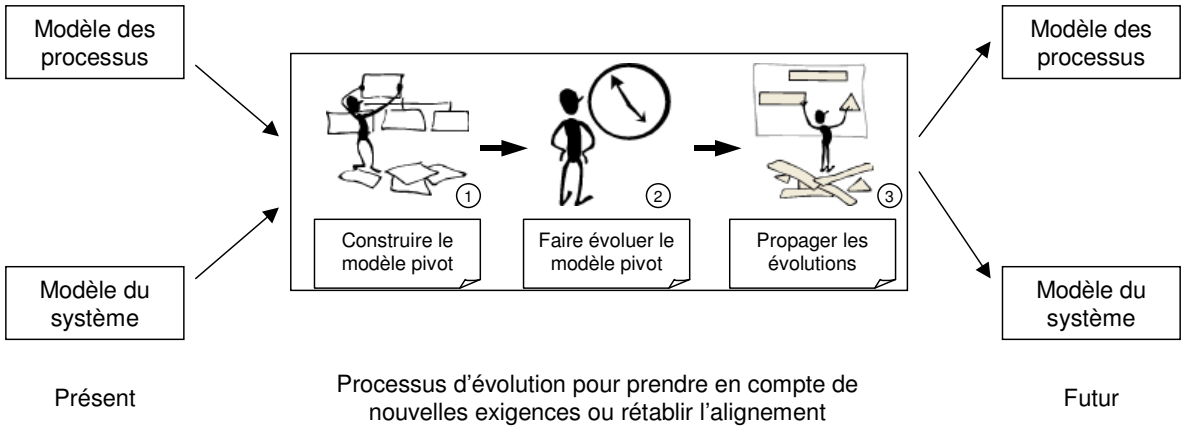


Figure 3 : Schéma de la démarche

5. Plan de la thèse

Le Chapitre 2 présente un état de l'art des approches traitant de l'alignement. Cet état de l'art est organisé selon un cadre de référence qui permet de présenter différents aspects de l'alignement et de positionner des approches les unes par rapport aux autres.

Le Chapitre 3 définit la relation d'alignement et présente l'ensemble des métriques. Ces métriques sont organisées dans un cadre qui permet de montrer les différentes facettes de la relation d'alignement. Ces métriques sont d'abord définies entre les concepts de méta-modèles représentant les processus d'entreprise et le système à un niveau générique. Elles sont ensuite adaptées à un niveau spécifique pour deux modèles particuliers.

Le Chapitre 4 présente un cas d'application illustrant l'utilisation de ces métriques.

Le Chapitre 5 décrit la méthode ACEM dans les grandes lignes.

Le Chapitre 6 présente le méta-modèle pivot qui permet de mettre en évidence les concepts et la structure du modèle pivot servant, dans la méthode ACEM, à représenter conjointement les processus et le système.

Le Chapitre 7 précise la notion d'exigences d'évolution matérialisées par des opérateurs et permettant la mise en œuvre de l'évolution. Dans ce chapitre, nous proposons un processus pour aider les ingénieurs d'évolution à définir un ensemble d'opérateurs d'écart associé à un méta-modèle particulier. Ce processus s'appuie sur une typologie générique associée à un méta-modèle générique.

Le Chapitre 8 définit formellement la méthode ACEM à l'aide du méta-modèle de Carte. Nous proposons un guidage qui repose sur un ensemble de directives indiquant à la fois comment s'orienter dans la méthode et comment réaliser les étapes qu'elle propose de suivre.

Le Chapitre 9 illustre notre approche de co-évolution par une étude de cas de réservation de produits hôteliers.

Le Chapitre consacré à la conclusion résume l'approche proposée dans cette thèse et propose de nouveaux développements sur ce thème.

CHAPITRE 2 : ETAT DE L'ART

1. Introduction

Depuis quelques années, de nombreux chercheurs s'intéressent à l'alignement des technologies de l'information à la stratégie d'entreprise. De nombreux ateliers de travail, dont [BPMDS04] et [REBNITA05], ont été consacrés à ce sujet. Plusieurs raisons peuvent expliquer cet intérêt :

- de nombreux projets échouent parce que le système n'est pas conforme aux besoins des utilisateurs [Meta03] ;
- un mauvais alignement entre le système et les processus d'entreprise engendre une baisse de la performance de l'organisation [Sabherwal01], [Henderson93]. Or une telle situation est fréquente. Elle survient souvent après l'évolution de l'une de ces deux entités car aucun ajustement n'est effectué sur la deuxième entité.

Les travaux sur l'alignement sont très variés. Ils portent sur l'alignement entre différentes entités qui peuvent être les stratégies d'entreprise, les stratégies relatives aux technologies de l'information (TI), l'architecture, le code, l'environnement, les processus d'entreprise ou l'organisation. Ils traitent de la construction, de l'évaluation, de l'évolution ou du maintien de la relation d'alignement en situation de changement…

Ce chapitre a pour but de définir un cadre multidimensionnel utile à l'analyse des approches sur l'alignement. Ce cadre de référence s'inspire de [Rolland98] et est composé de quatre vues. Chacune de ces vues explore des caractéristiques de l'alignement. Ainsi, ce cadre permet (1) d'identifier des problèmes sous-jacents à l'alignement et (2) de positionner les approches de recherche les unes par rapport aux autres.

La suite du chapitre est organisée de la façon suivante : la section 2 décrit le cadre de référence pour l'alignement ; la section 3 correspond à l'étude de différentes approches suivant ce cadre de référence avant de conclure à la section 4.

2. Cadre de référence pour l'alignement et l'évolution des systèmes

Comme le montre la Figure 4, le cadre de référence suggère de considérer l'alignement suivant quatre vues. Chaque vue permet d'analyser un aspect particulier de l'alignement en posant une question fondamentale. A l'inverse, chaque approche d'alignement peut être analysée selon les quatre vues. Les quatre questions posées sont celles du "quoi", du "pourquoi", du "comment", et enfin du "par quel moyen". Elles permettent de s'intéresser (1) à l'objet de l'alignement, c'est-à-dire, aux entités que l'on cherche à aligner et aux liens entre ces entités, (2) aux buts de l'approche, (3) à la méthode mise en œuvre pour atteindre ces buts et (4) aux outils utilisés.

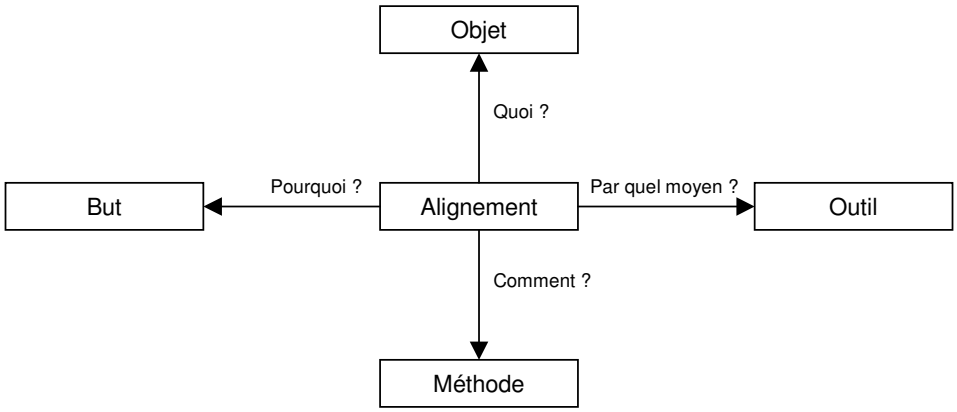


Figure 4 : Les quatre vues du cadre de référence

Chaque vue est caractérisée et mesurée à l'aide d'un ensemble d'attributs. Tous les attributs possèdent des valeurs définies dans un domaine qui peut être un type prédéfini (entier, booléen…), un type énuméré (Enum {x, y}) ou un type structuré (Ensemble ()).

Par exemple, la vue objet comprend trois attributs :

- le nombre d'entités à aligner défini par un entier.
- les entités auxquelles on s'intéresse, prises dans l'ensemble énuméré suivant : stratégie d'entreprise, stratégie des techniques de l'information, processus d'entreprise, système, exigence, environnement, architecture, logiciel, code.
- les relations entre ces entités qui peuvent être des liens de traçabilité, une typologie de règles, une typologie de liens ou qui peuvent également ne pas être définies.

Une approche d'alignement peut-être positionnée selon la vue objet avec les valeurs d'attributs suivantes :

Attribut	Domaine	Exemple de valeur
Nombre d'entités	Entier	2
Entités	Set of (stratégie d'entreprise, stratégie des techniques de l'information, processus d'entreprise, système, exigence, environnement, architecture, logiciel, code)	Stratégie d'entreprise, système
Relations entre les entités	Enum {liens de traçabilité, règles, liens, non définis}	liens

Tableau 1 : Vue Objet : Attributs, domaines et exemples de valeurs

Le domaine de chacun des attributs permettant de décrire les différentes vues est détaillé dans la suite de cette section.

2.1. La vue objet

Regev définit de manière simple l'alignement comme "*la correspondance entre un ensemble de composants*" [Regev 04]. Cette définition met en perspective deux aspects que l'on retrouve dans toutes les approches : d'une part, l'alignement met en jeu plusieurs ensembles de composants et d'autre part elle repose sur des relations définies entre ceux-ci. Le nombre d'ensembles de composants (on parlera plus simplement d'entités) à aligner n'est cependant pas le même selon les approches. Nous proposons d'associer à la vue objet les trois attributs suivants : (i) le nombre d'entités, (ii) les entités et (iii) les relations entre ces entités. Le reste de la section présente en détail chacun des attributs.

2.1.1 Nombre d'entités

Par définition, il ne peut y avoir d'alignement entre moins de deux entités. A l'inverse, il peut être possible de chercher à en aligner davantage. Ainsi, Regev utilise les expressions d'alignement "bivariant" lorsqu'il met en jeu deux entités, et d'alignement "multivariant", lorsqu'il en fait intervenir plus de deux [Regev04]. Le nombre d'entités considérées peut varier en fonction du degré de détail de l'analyse. Ainsi, on peut dire que deux entités sont en jeu lorsque l'on considère l'alignement entre un processus d'entreprise et le système, alors que si l'on considère l'alignement entre les acteurs impliqués, la structure organisationnelle, les processus, les produits, le nombre d'éléments est de fait supérieur à deux. On est alors dans le cas d'un alignement multivariant.

Les approches d'urbanisme ("Enterprise Architecture" en anglais) ont pour but d'assurer l'alignement du système d'information avec la stratégie d'entreprise [Longépé01], [Bonne04]. Ce sont alors tous les composants de la stratégie et du SI que l'on cherche à aligner. De même, le cadre d'Enterprise Architecture de Zachman [Zachman99] compte 36 modèles répartis selon six perspectives (présentées en lignes) et selon six aspects (représentés par les colonnes) de l'entreprise et du système. L'ensemble des modèles de chaque ligne et de chaque colonne doivent être alignés.

Plutôt que de chercher un alignement global, certains auteurs définissent des alignements bivariants entre des paires d'entités. Par exemple, [Camponovo04] préfère définir l'alignement entre trois entités à aligner au moyen de trois alignements bivariants plutôt qu'au moyen d'un alignement multivariant plus complexe et moins expressif.

L'attribut du nombre d'entités est défini comme suit :

Nombre d'entités : Entier

2.1.2 Entités

Les premières études sur l'alignement sont celles de Henderson et Venkatraman qui ont proposé le modèle d'alignement stratégique (SAM) présenté Figure 5. Dans ce modèle, les flèches bidirectionnelles représentent les relations d'alignement [Hederson93].

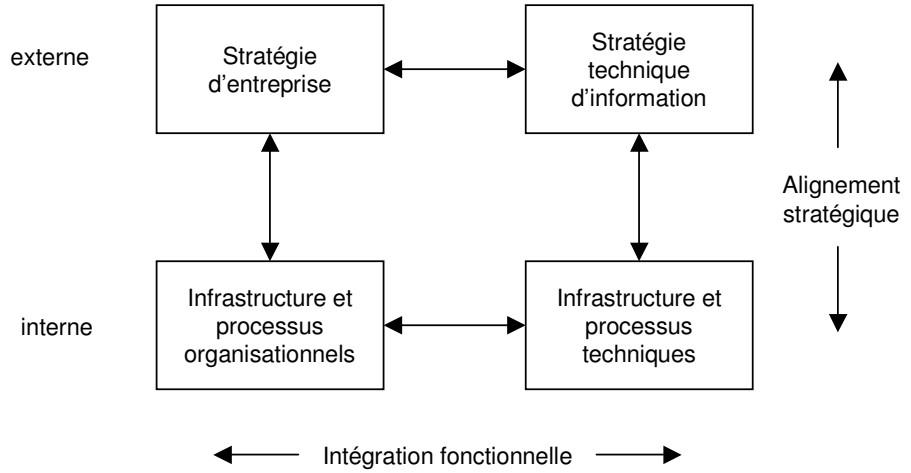


Figure 5 : Modèle de l'alignement stratégique

Le modèle d'alignement stratégique de Henderson et Venkatraman s'intéresse à l'intégration des *technologies de l'information* dans la *stratégie d'entreprise* en recommandant l'alignement entre quatre domaines : (1) la stratégie d'entreprise, (2) la stratégie des technologies de l'information (TI), (3) l'infrastructure et les processus organisationnels et (4) l'infrastructure et les processus techniques. La stratégie d'entreprise et la stratégie technique d'information sont dites externes. Les deux autres domaines sont qualifiés d'internes. Le domaine externe fait référence à la position d'une organisation par rapport à ses concurrents tout en respectant ses combinaisons produits/marché, ses partenaires, son intégration. Le domaine interne concerne la structure de l'organisation, ses départements, ses processus. Un découpage orthogonal est fait entre le domaine d'entreprise et le domaine des technologies de l'information. La fonction des technologies de l'information s'intéresse aux compétences, à l'architecture, aux processus d'une organisation. Le domaine d'entreprise s'intéresse aux mêmes aspects mais dans la perspective de la mission de l'organisation.L'alignement inter domaines est défini selon deux axes : "l'alignement stratégique" entre le domaine interne et le domaine externe et "l'intégration fonctionnelle" entre le domaine d'entreprise et le domaine des technologies de l'information.

Le modèle SAM est devenu un outil de gestion dont l'enjeu est de faire du système d'information un atout au service de la stratégie de l'entreprise. Il a inspiré de nombreux modèles [Luftman96], [Maes99], [Goedvolk00] et pratiques des cabinets de conseil en alignement [Maes00]. Le modèle SAM propose quatre approches différentes pour établir un alignement suivant que la stratégie d'entreprise ou la stratégie TI est prise comme force directrice.

Ce modèle montre que la notion d'alignement est complexe et que l'objet de l'alignement peut dépasser le lien entre technologie de l'information et stratégie d'entreprise.

Certains auteurs, comme [Barclay97], perçoivent l'alignement uniquement d'un point de vue stratégique. Il s'agit alors de s'intéresser aux stratégies d'entreprise et aux stratégies des technologies de l'information. De nombreux auteurs perçoivent cependant un intérêt à définir l'alignement entre les stratégies et d'autres entités. Par exemple, [Bleistein05a] s'intéresse à l'alignement à un niveau stratégique, mais entre les *stratégies d'entreprise* et les composants du *système d'information*, et non pas au niveau de la stratégie qui régit son évolution.

Regev *et al.* [Regev04] définit l'alignement interne comme un alignement entre les composants internes d'une organisation donnée, par exemple entre les *processus d'entreprise* et le système qui les supporte (l'alignement externe correspondant par ailleurs à l'alignement entre l'organisation et son environnement). [Mitleton00] et [Camponovo04] considèrent et étudient ce type d'alignement entre le *système d'information* et son *environnement*. L'alignement entre le système et les processus d'entreprise est le sujet de nombreux travaux tels que [Arsanjani01], [Bodhuin04], [Soffer04b], [Kardasis98], [Giaglis99], [Wegmann05]], ou bien [Aerts04] et [Wieringa03] qui s'intéressent à l'alignement entre *l'architecture logicielle* et *l'architecture des processus d'entreprise*.

De nombreux auteurs s'intéressent à l'alignement entre les exigences du système, son architecture et le logiciel. Ainsi, [Lamsveerde03], [Landtsheer03], [Krishna04] et plus généralement la communauté de l'ingénierie des exigences dans son ensemble étudient comment passer du modèle *d'exigences* aux spécifications de l'architecture du système afin que celui-ci réponde parfaitement aux exigences. [Rosemann04] et [Zoukar05] adoptent une

démarche proche mais légèrement différente en évaluant l'alignement entre les exigences de l'organisation et les fonctions offertes par des progiciels de gestion intégré.

Les entités mises en jeu dans le cadre d'un alignement peuvent être de neuf types différents comme le montre la définition ci-dessous :

Entités : set of {stratégie d'entreprise, stratégie des techniques de l'information, processus d'entreprise, système, exigence, environnement, architecture, logiciel, code}

2.1.3 Relations entre les entités

La correspondance entre les entités à aligner peut se définir de différentes façons : par des *règles*, des *liens* ou des *liens de traçabilité*.

La *traçabilité* des exigences permet de lier les exigences à leurs sources (pré-traçabilité) et aux artefacts créés durant le développement du système à partir de ces exigences (post-traçabilité) [Ramesh01]. Il est important que, durant la phase de découverte des exigences, les raisons d'être et les sources des exigences soient spécifiées afin de comprendre leur évolution [Ramesh01]. De la même façon, la post-traçabilité permet d'identifier, à partir d'un changement d'exigence, les éléments conceptuels et les codes qui sont affectés par ce changement [Clarke99].

Les *règles* aident à définir la dynamique de passage entre des entités, par exemple lorsqu'elles sont exprimées dans des langages différents. Ainsi, Landtsheer et al. [Landtsheer03] proposent une technique pour dériver des spécifications conceptuelles événementielles (écrites dans le langage tabulaire SCR) à partir de spécifications d'exigences formulées avec le langage orienté-but KAOS.

Il est également possible de définir, grâce à des *liens*, une correspondance entre les parties d'entités que l'on cherche à aligner. Ainsi, [Wan-Kadir04] définit un modèle de liens entre les classes, les relations et les contraintes du système définis avec UML et les règles de gestion afin de répercuter facilement les changements de règles de gestion sur le système. [Wegmann05] spécifie formellement deux liens d'alignement respectivement entre deux niveaux fonctionnels et entre deux niveaux organisationnels. Les liens définis par [Krishna04a] aident à spécifier la correspondance entre des concepts du méta-modèle i* et ceux du langage Z.

Dans certaines approches, la nature des relations entre les différentes entités n'est *pas précisée*. Cette situation apparaît fréquemment dans les études de l'alignement stratégique. Ainsi, [Luftman00] ne définit pas les relations qui peuvent exister entre les stratégies d'entreprise et les stratégies relatives aux technologies de l'information. Il identifie cinq facteurs de succès et cinq facteurs d'échec à partir de son expérience et d'enquêtes auprès de directeurs généraux et de directeurs des systèmes d'information. On trouve par exemple parmi les facteurs d'échec le manque de relation entre les gestionnaires et les ingénieurs ou le fait que les technologies de l'information n'honorent pas leurs engagements. Parmi les facteurs de succès, on trouve le fait que les cadres supérieurs soutiennent les technologies de l'information ou que les ingénieurs sont impliqués dans la stratégie de développement.

Nous proposons de définir la facette relation entre entités comme suit :

Relation entre entités : enum {traçabilité, règles, liens, pas défini}

2.2. La vue but

Il y a peu de diversité dans les buts recherchés par les approches d'alignement. La vue but ne compte qu'un seul attribut, intitulé but, qui apporte une réponse directe à la question « pourquoi l'alignement est-il étudié ? »

L'alignement fait l'objet de méthodes essentiellement centrées autour de trois problématiques :

- *construire* l'alignement,
- *évaluer* l'alignement entre les entités concernées, et
- faire *évoluer* l'alignement (ou le maintenir) lorsque l'une des entités évolue.

De nombreux chercheurs s'intéressent aux mécanismes nécessaires à l'établissement de l'alignement [Coakley96] [Camponovo04]. Il s'agit de construire un système aligné avec les processus d'entreprise [Wegmann05] ou les exigences de l'organisation [Lamsweerde01]. Ces approches ont pour but de *construire un alignement* entre les différentes entités en jeu "from scratch"[1].

Soffer [Soffer04b] suggère que l'identification d'un non-alignement nécessite de mesurer la relation d'alignement. Cette mesure peut être faite au moyen de métriques telles que celles proposées par [Bodhuin04]. Un exemple de métrique est la "couverture technologique" qui indique le pourcentage d'activités du processus supportées adéquatement par un système. Un autre exemple est "l'adéquation technologique" qui détermine dans quelle mesure les composants du système sont en adéquation avec l'activité *i* qu'ils supportent. Le non-alignement est détecté si l'une ou l'autre de ces mesures génère une valeur inférieure à un seuil fixé d'avance par l'organisation.

[Rosemann04] étudie la distance entre le monde des exigences organisationnelles et le monde du système dans le cadre de l'installation d'un ERP. Les auteurs caractérisent l'écart entre ces deux mondes en utilisant l'ontologie de Bunge, Wand et Weber. Ils cherchent à mesurer et à réduire la distance entre ces deux derniers ensembles afin d'obtenir un ERP qui soit aligné avec les besoins organisationnels. [Zoukar05] adopte une approche similaire en analysant les similarités entre les exigences du SI et les fonctionnalités proposées par l'ERP.

[Soffer04b] souligne que l'alignement entre deux entités est souvent rompu quand l'une des entités évolue. C'est alors dans le but de gérer l'évolution conjointe de ces deux entités qu'une gestion de l'alignement est entreprise. Les approches qui gèrent cette évolution conjointe supposent, contrairement aux méthodes d'alignement "from scratch" évoquées précédemment, que les différentes entités sont alignées au départ (au moins partiellement). Il existe un domaine de la biologie, la co-évolution, qui étudie l'interaction entre les espèces, c'est-à-dire l'influence qu'ont certaines espèces sur l'évolution d'autres espèces. Par analogie, des recherches ont été réalisées dans le domaine de l'informatique pour explorer les influences réciproques des systèmes d'information (SI) ou des logiciels sur d'autres entités comme l'organisation, les processus d'entreprise, les systèmes complexes auxquels ils appartiennent, etc [Bodhuin04], [Krishna04], [Kardasis98]. Dans le cadre de l'ingénierie des

[1] A partir de rien.

systèmes, le concept de co-évolution permet d'assurer le maintien de la relation d'alignement entre le système d'information et les processus de l'entreprise.

L'expérience montre qu'il est également possible que ce soit non pas les entités qui évoluent, mais la relation qui existe entre elles. Ceci est particulièrement le cas pour l'alignement entre les stratégies d'entreprise et les stratégies des TI qui nécessite d'adopter ce que [Luftman00] appelle un "comportement de l'alignement". Ce terme correspond entre autres au fait qu'il existe un partenariat entre des ingénieurs TI et des responsables business ou que les ingénieurs comprennent les problèmes managériaux.

En résumé, concernant l'évolution, ce sont (i) soit les entités qui évoluent ce qui engendre une modification de l'alignement ou une évolution des autres entités pour maintenir cet alignement, (ii) soit la relation d'alignement qui évolue, entraînant un changement des entités en jeu.

Il est intéressant de noter que dans de nombreuses approches, l'évaluation de l'alignement est une étape pour construire ou faire évoluer celui-ci. On peut néanmoins considérer l'évaluation de l'alignement comme une fin en soi car elle permet aux différentes parties prenantes de rendre compte de la situation indépendamment de toute décision subséquente.

En conclusion, l'attribut but de la vue but est défini comme suit :

But : set of {construire, évaluer, évoluer}

2.3. La vue méthode

La méthode utilisée dans les approches d'alignement dépend du but à atteindre. Nous avons donc identifié trois attributs : (i) les méthodes *d'évaluation*, (ii) les méthodes de *construction* et (iii) les méthodes *d'évolution*. Chacun de ces trois attributs est détaillé dans l'une des trois sous-sections ci-dessous.

2.3.1 Méthodes d'évaluation

Les méthodes d'évaluation de l'alignement définissent un certain nombre de critères et se donnent les moyens pour les mesurer.

Nous proposons de classer les approches d'évaluation de l'alignement selon leur nature qualitative ou quantitative. [Beeson03], [Luftman00] ou même [Renner03] proposent des approches d'évaluation qui reposent sur l'interprétation, le jugement et la connaissance des acteurs de l'entreprise. Ce sont les appréciations subjectives de ces acteurs qui permettent de conclure à l'alignement ou au non alignement. Dans [Luftman00] les critères sont définis précisément, ce qui réduit le degré de subjectivité. La valeur retenue pour chaque critère dépend néanmoins de l'appréciation personnelle des experts impliqués.

D'autres approches proposent des critères d'évaluation associés à des mesures quantitatives. Il s'agit par exemple d'énumérer le nombre d'activités prises en charge par le système [Bodhuin04].

L'attribut méthode d'évaluation est défini comme suit :

Méthode d'évaluation : Enum {quantitative, qualitative}

2.3.2 Méthodes de construction

Les méthodes de construction peuvent être classées en trois catégories : les approches *top-down*, les approches *bottom-up* et les approches *mixtes*.

La plupart des méthodologies de développement de SI proposent un processus par étapes pour s'assurer que le système d'information conçu correspond bien aux besoins et aux stratégies de l'organisation. Ainsi, des méthodologies anciennes comme les méthodes participatives [Munford81] et plus récemment la communauté d'ingénierie des exigences dans son ensemble [Dardenne93] proposent des approches qui reposent sur des modèles de buts pour capter les stratégies d'entreprises et sur des règles de transition pour opérationnaliser des buts par des spécifications du système. Ces méthodes de construction sont par nature *top-down*. Par exemple [Bleistein05a] définit le système à partir des stratégies de l'organisation en affinant les buts jusqu'à atteindre des buts correspondant à des besoins opérationnels.

A l'inverse, il est possible de construire l'alignement entre différentes entités en suivant une approche *bottom-up*. Il s'agit alors d'identifier des buts à partir des détails techniques et opérationnels en essayant de répondre à la question "pourquoi". [Krishna04] considère par exemple qu'il est possible de dériver les modèles i* représentant les exigences de l'organisation à partir de spécifications formelles du système écrites en Z.

Certaines approches combinent les deux types d'approches précédentes. Ainsi les approches d'urbanisme sont top-down sur le plan global, mais l'alignement entre deux niveaux est analysé de façon bottom-up, comme dans [Wieringa03]. Nous qualifions ce type d'approche de *mixte*.

L'attribut méthode de construction est défini comme suit :

Méthode de construction : Enum {Top-down, bottom-up, mixte}

2.3.3 Méthodes d'évolution

Nous avons identifé quatre façons de maintenir la relation d'alignement : (i) les méthodes à base de *scénarios*, (ii) les méthodes reposant sur le lien de *dépendance* entre les entités (iii) les méthodes dirigées par les *modèles* et (iv) les méthodes qui cherchent à *corriger* un mauvais alignement.

2.3.3.1 Méthodes à base de scénarios

Ces méthodes proposent de définir non pas un futur unique, mais plusieurs futurs possibles sous la forme de scénarios. Les méthodes traditionnelles de prévision stratégique ne sont pas toujours appropriées car elles reposent sur l'hypothèse que le futur peut être prévu de façon suffisamment fiable pour choisir une direction stratégique claire. Or ceci n'est pas le cas dans les environnements dits "turbulents", c'est-à-dire changeant fréquemment. Les approches par planification à base de scénarios peuvent s'avérer utiles dans la mesure où elles sont bien adaptées aux environnements complexes et incertains tels qu'on les observe dans certaines organisations [Nurcan99].

Les scénarios sont des descriptions de situations futures possibles. Leur identification suit un processus systématique, interactif et imaginatif. Le processus se termine avec l'élaboration de

trois ou quatre scénarios présentant des alternatives plausibles plutôt qu'une extrapolation du présent [Camponovo04], [Godet00].

2.3.3.2 Méthodes de dépendance

L'évolution d'une entité entraîne souvent une rupture de la relation d'alignement. Son impact sur les autres entités doit être analysé. Il existe quatre approches différentes : indépendance, interdépendance, dépendance, et double dépendance [Lämmel04] et [Etien05a]. Chaque famille est définie selon la direction du lien de dépendance entre les entités qui évoluent.

Nous considérons dans la suite de cette section que le nombre d'entités alignées se réduit à deux, aucune approche de co-évolution mettant en jeu plus de deux entités n'ayant été identifiée dans la littérature.

La co-évolution est gérée de façon indépendante (Figure 6) quand il n'existe pas de dépendance entre les processus d'ingénierie qui permettent de faire évoluer chacune des deux entités. Ce cas est typiquement celui des projets confrontés à de fortes pressions temporelles : les évolutions sont alors directement implémentées dans le système sans vérifier la cohérence avec les processus cibles. Cette approche est également utilisée dans les projets d'évolution qui impliquent des systèmes en fin de vie. Une telle approche nécessite de vérifier rétrospectivement la relation d'alignement pour s'assurer que le système (E2) est aligné avec l'autre entité (E1) [Mens04]. Si l'alignement est rompu, d'autres évolutions seront nécessaires et un nouveau cycle d'ingénierie est entrepris. Cette approche est utilisée dans les projets d'installation d'ERP où la paramétrisation d'une transaction doit être implémentée pour vérifier, grâce à des tests de non régression, qu'il n'y a pas d'impact sur les autres transactions ou modules.

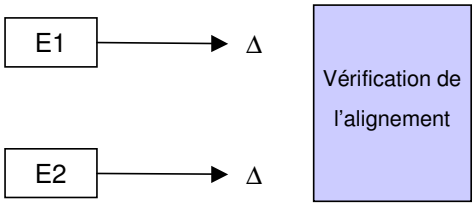


Figure 6 : Méthode d'indépendance

Dans les approches de dépendance, l'évolution de l'entité E2 se déduit de l'évolution de l'entité co-évoluant E1. Comme le montre la Figure 7, chaque évolution élémentaire de l'entité 'maître' E1 est répercutée sur l'entité 'esclave' E2 par application de règles [Krishna04a] et de mesures d'alignement [Bodhuin04] assurant ainsi l'alignement entre les deux entités en présence. Typiquement, ce type d'approche est utilisé dans les cas d'amélioration des processus d'entreprise. L'évolution du système est alors entreprise pour satisfaire les évolutions des processus. [Krishna04a] utilise ce type d'approche pour déduire les évolutions du système représenté en Z à partir d'évolutions des exigences exprimées au niveau intentionnel avec des modèles i*.

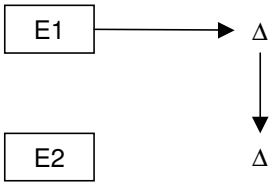


Figure 7: Méthode de dépendance

On parle de double dépendance quand chaque entité E1 ou E2 co-évoluant joue le rôle de maître c'est-à-dire qu'on est capable à partir des évolutions de chaque entité de déduire les évolutions de l'autre. Par exemple, [Kardasis98] propose des règles pour évaluer l'impact de l'évolution des processus d'entreprise sur le système et vice-versa. Une approche de double dépendance peut être considérée comme une combinaison de deux approches de dépendance simple. La Figure 8 schématise cette approche.

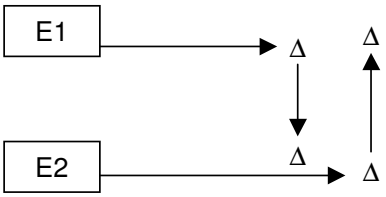


Figure 8: Méthode de double dépendance

L'approche d'interdépendance s'appuie sur l'existence d'un modèle commun. L'évolution du modèle pivot peut reposer sur des méthodes traditionnelles [Han97], sur l'existence et le respect d'invariants [Banerjee87] ou sur le fait que les deux entités sont représentées ensemble dans le même modèle. Il est ensuite nécessaire de répercuter les évolutions du pivot sur les deux entités. Un schéma de cette approche est proposé dans la Figure 9.

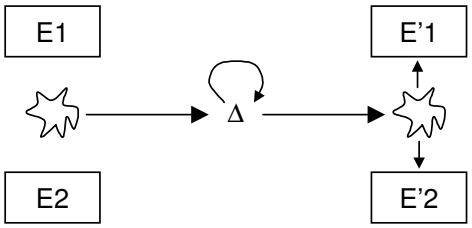


Figure 9 : Méthode d'interdépendance

2.3.3.3 Méthodes dirigées par les modèles

Le développement dirigé par les modèles (en anglais "Model Driven Architecture" ou MDA) diffère des approches traditionnelles de développement de logiciel en considérant les modèles comme le concept clé. Ces approches proposent d'organiser le(s) modèle(s) définissant un système. Pour atteindre cet objectif, l'architecture dirigée par les modèles propose un découpage des modèles selon deux préoccupations majeures : (i) l'expression des fonctionnalités d'un système, les PIMs (Platform Independant Model) ; (ii) l'expression des spécificités technologiques d'une mise en oeuvre de ces fonctionnalités, les PSMs (Platform Specific Model). Ainsi, associée aux standards technologiques, l'approche MDA permet (1) de dissocier le code métier et le code technique et (2) de mettre en œuvre un même modèle de fonctionnalités à l'aide de solutions technologiques variées. Elle permet aussi d'intégrer des applications en mettant en relation leurs modèles respectifs. Il devient alors possible de supporter l'intégration, l'interopérabilité et l'évolution des applicatifs au fur et à mesure de l'évolution des besoins et de l'apparition et la disparition des solutions technologiques [Marvie04].

Les relations entre modèles sont définies par des "transformations". Ce type d'approche a pour principal avantage de garder non seulement trace des informations de conception au sein des modèles, mais aussi du processus par lequel la conception a été affinée et implémentée. Ceci permet de représenter le changement et de le propager facilement. Les approches dirigées par les modèles semblent prometteuses pour gérer les problèmes d'évolution et

d'alignement. Cependant, la maîtrise de l'évolution dans les approches dirigées par les modèles reste dépendante de la qualité des langages de transformation ; or ceux-ci ne font qu'émerger dans le monde de la recherche [Hearnden03].

2.3.3.4 Approches de correction

Certaines approches, comme [Bodhui04] ou [Luftman00], s'appuient sur les résultats de l'évaluation de l'alignement pour définir les évolutions à mettre en œuvre. Ceci peut se faire de différentes façons suivant le moyen utilisé pour évaluer l'alignement. Il est par exemple possible de mettre en œuvre des actions correctives, c'est-à-dire d'identifier des évolutions à entreprendre sur l'une des entités afin d'augmenter la valeur d'une ou plusieurs mesures. L'évolution se fait ainsi pas à pas en évaluant l'alignement après chaque changement [Bodhuin04].

Parfois, l'évaluation produit une valeur caractérisant le niveau d'alignement atteint. Dans ce cas, la description du niveau limite peut servir de base à la mise en œuvre de l'évolution [Luftman00].

L'attribut méthode d'évolution est défini comme suit :

Méthode d'évolution : Set of {scénario, indépendance, dépendance, double dépendance, interdépendance, modèle, correction}

2.4. La vue outil

La vue outil ne comporte qu'un seul attribut « outil » qui indique si l'outil peut être utilisé pour évaluer, construire ou faire évoluer l'alignement entre plusieurs entités. Certains outils sont uniquement adaptés à l'un ou l'autre des objectifs, d'autres pouvant servir à la réalisation de plusieurs d'entre eux. Certains auteurs proposent d'utiliser des questionnaires pour évaluer l'alignement. De tels questionnaires permettent d'identifier comment les gestionnaires et les ingénieurs perçoivent l'alignement entre les stratégies d'entreprise et les stratégies TI. Ainsi, dans [Kefi03] les réponses aux questions sont choisies sur une échelle de Likert en cinq points (de 1 entièrement d'accord à 5 complètement en désaccord avec la proposition faite dans la question). Un exemple de question est : « Etes-vous d'accord avec l'affirmation suivante : Les stratégies d'entreprise et les stratégies des technologies de l'information sont alignées dans mon organisation ? ». Les questions s'appuient souvent sur les facteurs de succès et d'échec de [Luftman00], s'intéressent à la communication entre gestionnaires et ingénieurs ou la compréhension que les uns ont du domaine des autres.

Les métriques sont également utilisées [Bodhuin04] ou suggérées [Soffer04b] pour réaliser des mesures quantitatives dans le cadre de l'évaluation de l'alignement. Bodhuin propose par exemple deux métriques : la couverture et l'adéquation technologique qui correspondent respectivement au taux d'activités des processus prises en charge par le système et le taux d'adéquation entre les composants d'une activité donnée et ceux du système [Bodhuin04].

Le modèle CMM (Capability Maturity Model) est un modèle qui permet d'évaluer et de faire évoluer des processus logiciels, en proposant des actions d'amélioration. Ce modèle comporte cinq niveaux de maturité : initial, reproductible, défini, maîtrisé et optimisé. Ces niveaux constituent autant d'étapes sur le chemin menant à des processus matures, c'est-à-dire conformes à un ensemble de bonnes pratiques observées à travers le monde dans des

entreprises réputées pour bien gérer leurs processus. A partir de ce modèle largement utilisé par les organisations, Luftman a défini le Strategic Alignment Maturity Assessment [Luftman00] permettant d'identifier le niveau d'alignement entre les stratégies d'entreprise et les stratégies des technologies de l'information. Ce cadre compte cinq niveaux comme le CMM. L'identification du niveau atteint par une organisation repose sur l'évaluation de six critères : la maturité de communication, la maturité de la capacité à mesurer, la maturité à diriger, la maturité du partenariat entre gestionnaires et ingénieurs, la maturité de l'architecture et la maturité des connaissances. Les valeurs pour ces critères aux différents niveaux ont été identifiées et validées par la pratique.

Le modèle Balanced Scorecard (en français – tableau de bord) est un instrument stratégique qui comprend quatre axes dont chacun correspond à une question particulière :

- l'axe financier essaie de répondre à la question « est-ce que l'entreprise peut créer une valeur pour les actionnaires ? »
- l'axe des clients s'intéresse à la façon dont les clients perçoivent l'entreprise et à leur vision de l'entreprise pour que celle-ci soit capable d'atteindre les objectifs fixés.
- l'axe de l'apprentissage organisationnel essaie de répondre à la question « de quelle manière l'entreprise doit-elle soutenir ses capacités à s'adapter aux changements pour atteindre les objectifs fixés ? »
- l'axe des processus internes s'intéresse au processus interne que l'organisation doit développer pour mieux satisfaire les clients et les actionnaires.

Pour chacun de ces axes, des objectifs doivent être définis, et pour chacun des objectifs, des indicateurs doivent être précisés. Ces indicateurs doivent être quantifiables et complémentaires les uns des autres. Les objectifs relèvent du niveau stratégique alors que les indicateurs sont précisés au niveau opérationnel. Le couplage objectif/indicateur permet donc un alignement entre ces deux niveaux [Kaplan92].

Le modèle Balanced Scorecard a été adapté aux technologies de l'information (Balanced scorecard IT), mais pour être appliqué uniquement sur des projet TI. Ceci a provoqué l'isolement des TI par rapport aux autres fonctions de l'entreprise. Des chercheurs ont donc proposé d'intégrer le modèle Balanced Scorecard dans d'autres approches pour permettre une meilleure communication entre les gestionnaires et les ingénieurs [Hu05].

Certaines approches utilisent des règles de dérivation pour construire ou faire évoluer les entités alignées. Ces règles s'appuient sur les relations entre les différentes entités pour identifier (i) à quoi doit ressembler telle ou telle autre entité en fonction d'une autre entité [Landtsheer03] et (ii) comment vont évoluer les entités lorsqu'un premier changement initial est survenu à l'une d'entre elles [Krishna04a].

Certains auteurs proposent d'utiliser un langage unique pour gérer l'alignement entre différentes entités. Ainsi, certaines approches proposent d'imposer le même paradigme de développement à tous les artefacts du système pour construire un code aligné avec le design. Par exemple, le design et le code sont tous deux écrits avec des paradigmes orientés objet dans [Clarke99].

Cette idée est aussi exploitée par SysML [SysML] qui propose d'adapter UML à différentes disciplines impliquées dans des projets d'ingénierie des systèmes telles que l'électronique ou

la mécanique. Ceci permet de comprendre comment les différents modèles utilisés dans ces disciplines s'intègrent les uns aux autres.

[Bleistein05a] propose une approche d'ingénierie des exigences qui unifie la modélisation des stratégies d'entreprise et la modélisation des exigences du système. Cette unification permet de valider les exigences du système avec les objectifs stratégiques de l'entreprise grâce à des liens explicites au sein d'un modèle unique.

L'attribut outil se définit comme suit :

Outil : Set of {questionnaire, métriques, cadre, tableaux de bord, règles de dérivation, langage unique}

En synthèse, le cadre de référence montre l'ensemble des qualités offertes par les approches d'alignement. Ces qualités sont décrites dans la perspective de quatre vues, intitulées objet, but, méthode et outils dont les détails sont rassemblés dans la figure 10.

Vue Objet
Nombre d'entités : Entier
Nature des entités : set of {stratégie d'entreprise, stratégie des techniques de l'information, processus d'entreprise, système, exigence, environnement, architecture, logiciel, code}
Relation entre entités : enum {traçabilité, règles, liens, pas défini}
Vue But
But : set of {construire, évaluer, évoluer}
Vue Méthode
Méthode d'évaluation : Enum {quantitative, qualitative}
Méthode de construction : Enum {Top-down, bottom-up, mixte}
Méthode d'évolution : Set of {scénario, indépendance, dépendance, double dépendance, interdépendance, modèle, correction}
Vue Outil
Outil : Set of {questionnaire, métriques, cadre, tableaux de bord, règles de dérivation, langage unique}

Figure 10 : Résumé du cadre de référence

3. Positionnement de sept approches au moyen du cadre de référence

Cette section présente une évaluation de sept approches d'alignement :

- une approche d'évaluation et d'évolution de l'alignement stratégique [Luftman00]
- une approche de dérivation de spécifications à partir de modèles orientés but [Landtsheer03] et [Landtsheer04]
- une approche d'alignement entre processus et système [Wegmann05a] et [Wegmann05b]
- une approche d'alignement entre l'architecture et le contexte d'entreprise [Wieringa03]

- une approche d'alignement orientée-besoins [Bleistein05a] et [Bleistein05b]
- une approche de co-évolution reposant sur l'évaluation d'un mauvais alignement [Bodhuin04]
- une approche de co-évolution reposant sur l'impact de changement [Krishna04a]

Ces sept approches ont été selectionnées parmi toutes celles citées dans cette section, d'une part pour leur intérêt particulier, d'autre part parce qu'elles nous ont paru utilisables en pratique, et enfin parce qu'elles forment, selon nous, un ensemble représentatif de l'état de l'art. Cette partie décrit brièvement ces approches et les situe par rapport au cadre de référence décrit ci-dessus.

3.1. Une approche d'évaluation et d'évolution de l'alignement stratégique

[Luftman00] propose un cadre pour mesurer l'alignement entre les stratégies d'entreprise et les stratégies des technologies de l'information (TI). Ce cadre reprend les fondements du modèle CMM (Capability Maturity Model) : (i) il comprend cinq niveaux constituant autant d'étapes sur le chemin de l'alignement, (ii) les niveaux ont été identifiés par les auteurs en s'appuyant sur les bonnes pratiques et (iii) les niveaux supérieurs servent d'objectifs à atteindre.

Afin d'identifier le niveau d'alignement d'une organisation à un moment donné, six critères ont été identifiés :

- le degré de maturité de communication,
- le degré de maturité de la capacité à mesurer,
- le degré de maturité à diriger,
- le degré de maturité du partenariat entre gestionnaires et ingénieurs,
- le degré de maturité de l'architecture et la maturité des connaissances.

Pour chacun de ces critères des attributs sont définis, ainsi que différentes valeurs possibles pour ces attributs. Ces valeurs sont nommées caractéristiques. Elles permettent une évaluation selon une échelle de Likert en cinq échelons (1 = pas d'alignement à 5 = fort alignement) avec un échelon par niveau. Le niveau d'alignement est déterminé en fonction des caractéristiques obtenues pour chacun des attributs des différents critères. Le Tableau 2 décrit partiellement les caractéristiques du niveau 1. Les autres critères ainsi que les autres niveaux sont définis de la même façon.

Niveau	Critère	Attribut	Caractéristique
Niveau 1 : Processus initial, ad-hoc	Communication	Compréhension du domaine managérial par le domaine technique	Minimum
		Compréhension du domaine technique par le domaine managérial	Minimum
		Apprentissage inter/intre organisationel	Occasionnel, ad-hoc
		Rigidité du protocol	Commande et contrôle
		Partage de la connaissance	Ad-hoc
		Liaison(s) etendue efficacité	Aucune ou ad-Hoc
	Mesure	Métriques TI	Technique ; non reliée au domaine managérial
		Métriques de gestion	Ad-hoc ; non reliée aux TI
		Métriques d'équilibre	Ad-hoc non liées
		Accord du niveau service	Présent de façon sporadique
		Références	Pas pratiqué de façon générale
		Revues/estimations formelles	Aucune
		Amélioration continue	Aucune
	Direction	...	...
	Partenariat	...	...
	Architecture	...	...
	Connaissances	...	...

Tableau 2 : Description du niveau 1

L'approche proposée pour atteindre et maintenir l'alignement s'appuie sur (i) la compréhension de la maturité de l'alignement, (ii) la maximisation des facteurs de succès et (iii) la minimisation des facteurs d'échec. L'auteur propose un processus en six étapes :

1. Définir les buts et former une équipe. Celle-ci doit être constituée de gestionnaires et d'ingénieurs provenant des différentes activités commerciales de l'entreprise (par exemple : marketing, finance, ingénierie, recherche et développement). L'équipe doit évaluer la maturité de l'alignement entre les stratégies d'entreprise et les stratégies des TI.

2. Comprendre le lien entre business et TI. L'équipe évalue chacun des six critères avec l'objectif de converger vers une unique caractéristique pour chacun des attributs.

3. Analyser et prioriser les écarts. Les différentes opinions émises par les participants indiquent des opportunités d'alignement différentes. Cette étape a pour but de comprendre les activités nécessaires pour améliorer l'alignement. Pour chacun des critères, les écarts entre la situation actuelle de l'organisation et la situation que l'équipe se fixe comme objectif doivent être priorisés. Le niveau de maturité supérieur sert de ligne directrice pour identifier les actions à entreprendre.

4. Spécifier les actions. Connaître le niveau de maturité de l'alignement aide à identifier les actions à entreprendre pour améliorer l'alignement. Cette étape a pour but d'affecter des tâches à chacun des écarts identifiés à l'étape précédente en définissant précisément, les documents à délivrer, les ressources, les risques, les mesures permettant de vérifier que l'écart a été comblé.

5. Choisir et évaluer les critères de succès. Cette étape nécessite de revoir les buts et rediscuter régulièrement sur les critères de mesure identifiés pour évaluer l'implémentation des projets.

6. Maintenir l'alignement. Cette étape est, selon Luftman, la plus difficile. Pour maintenir les bénéfices des TI, un "comportement d'alignement" doit être développé et cultivé.

Il est à noter que dans cette approche, les évolutions ne visent pas à modifier l'une ou l'autre des stratégies, mais à atteindre un plus haut niveau de maturité de l'alignement entre les stratégies. Ce ne sont donc pas les entités qui évoluent, mais la relation d'alignement elle-même.

L'approche d'évaluation et d'évolution de l'alignement stratégique de Luftman [Luftman00] se positionne de la façon suivante par rapport au cadre de référence :

Vue Objet
Nombre d'entités : 2
Nature des entités : {stratégie d'entreprise, stratégie des techniques de l'information}
Relation entre entités : pas définie
Vue But
But : {évaluer, évoluer}
Vue Méthode
Méthode d'évaluation : {qualitative}
Méthode d'évolution : {correction}
Vue Outil
Outil : {cadre}

3.2. Une approche de dérivation de spécifications à partir de modèles orientés but

Landtsheer *et al*. proposent de dériver des spécifications tabulaires reposant sur des événements à partir de modèles d'exigences orientés but [Landtsheer03] et [Landtsheer04]. Les premières sont définies avec le langage SCR (une spécification SCR définit la machine à travers un ensemble de tableaux) alors que les secondes sont représentées avec le modèle KAOS [Lamsweerde01]. De telles dérivations permettent d'enrichir les modèles de buts, mais aussi et surtout d'aligner les besoins et les spécifications du système qui les opérationnalisent.

La dérivation des tableaux SCR à partir de modèles KAOS utilise des règles. La mise en œuvre de ces règles sert de base au processus de dérivation. Parmi ces règles, on trouve par exemple, la transformation d'une sémantique « d'élagage » à une sémantique « générative ». Le langage KAOS a une sémantique d'élagage : tout changement est possible à l'exception de ceux qui sont interdits. Au contraire, SCR a une sémantique générative : chaque changement est interdit sauf ceux qui sont explicitement nécessaires aux spécifications. Lorsqu'une nouvelle opération est ajoutée au modèle source, la procédure de conversion doit être appliquée au nouveau modèle pour produire les nouveaux ensembles de tableaux. La différence sémantique est résolue en supposant que le modèle source capture tous les comportements acceptables et seulement ceux-ci … Six autres règles permettent de dériver les spécifications SCR à partir de modèles KAOS, par exemple pour palier au fait que SCR ne considère que deux agents alors que KAOS peut en modéliser plusieurs, ou qu'une machine SCR est déterministe alors que les agents KAOS sont non déterministes.

Le processus de dérivation consiste en une série d'étapes ayant pour but de supprimer les différences sémantiques, structurelles et syntaxiques entre KAOS et SCR.

L'approche de dérivation de spécifications à partir de modèles orientés but [Landtsheer03] et [Landtsheer04] se positionne de la façon suivante par rapport au cadre de référence :

Vue Objet
Nombre d'entités : 2
Nature des entités : {système, exigence}
Relation entre entités : typologie de règles
Vue But
But : {construire}
Vue Méthode
Méthode de construction : Top-down
Vue Outil
Outil : {règles de dérivation}

3.3. Une approche d'alignement entre processus et système

L'approche SEAM ("Systemic Enterprise Architecture Method") s'inscrit dans un contexte d'évolution. Elle a pour but de construire une situation future dans laquelle le marché, l'entreprise elle-même et son système sont alignés [Wegmann05b], (la relation d'alignement dans la situation initiale n'est pas précisée). La méthode SEAM s'intéresse donc non seulement à l'alignement entre le système et l'entreprise mais aussi d'un point de vue plus managérial et stratégique à l'alignement entre l'entreprise et son environnement, le marché.

Dans SEAM, l'entreprise est représentée par un modèle hiérarchique. Chaque niveau organisationnel représente une réalité partielle de l'entreprise. Chacun de ces niveaux contient des systèmes, c'est-à-dire des ensembles d'entités qui collaborent. Un système peut être un système informatique, un département, une entreprise, un réseau d'entreprises ou même un marché. Un modèle d'entreprise SEAM compte habituellement trois niveaux : (1) le niveau business représentant l'entreprise et ses partenaires, (2) le niveau opérationnel correspondant à certaines unités organisationnelles et (3) le niveau des technologies de l'information décrivant les employés et le système informatique. D'autres niveaux peuvent être ajoutés pour décrire par exemple le marché ou l'architecture du système.

L'interaction des entités au sein d'un même ensemble ou d'ensembles différents peut être décrite à différents niveaux de détail. Ces niveaux s'appellent des niveaux fonctionnels.

Dans SEAM, l'alignement se définit de la façon suivante :

- Alignement d'ensembles d'entités de niveaux organisationnels différents : deux représentations d'un ensemble d'entités à deux niveaux organisationnels adjacents sont alignés s'il est possible d'identifier le comportement décrit au niveau organisationnel le plus élevé dans le comportement décrit au niveau organisationnel le plus bas.

- Alignement d'ensembles d'entités de niveaux fonctionnels différents (au même niveau organisationnel) : deux représentations d'un système (i.e. ensemble d'entités) à deux niveaux fonctionnels différents sont alignés s'il est possible d'identifier le comportement décrit au niveau fonctionnel le plus haut dans le comportement décrit au niveau fonctionnel le plus bas.

- Alignement du business et des technologies de l'information : l'alignement du business et des TI requiert l'alignement d'ensembles d'entités de niveaux organisationnels différents et l'alignement d'ensembles d'entités de niveaux fonctionnels différents.

L'application de la méthode SEAM commence par la description du contexte incluant les différents participants de la collaboration. Par exemple, pour une application e-business, les participants sont l'organisation, les clients, les fournisseurs et la banque qui fournit le service de paiement par carte de crédit. L'un des rôles est décrit en détail ce qui permet de spécifier un autre niveau organisationnel qui doit être aligné avec le précédent. De même, si plusieurs niveaux fonctionnels sont décrits au même niveau organisationnel, il est important de s'assurer qu'ils sont alignés et ainsi de suite jusqu'à atteindre le niveau de l'implémentation. De cette façon, les relations entre les modèles sont explicites et les modèles sont alignés à la fois au sein d'un niveau organisationnel mais aussi entre différents niveaux organisationnels.

La méthode SEAM a aussi la particularité d'utiliser la même notation quel que soit le niveau d'abstraction. La différence entre les niveaux réside dans les heuristiques utilisées pour raisonner sur le contenu des diagrammes.

Vue Objet

Nombre d'entités : 3

Nature des entités : {processus d'entreprise, système, environnement}

Relation entre entités : typologie de liens

Vue But

But : {construire}

Vue Méthode

Méthode de construction : Top-down

Vue Outil

Outil : {règles de dérivation, langage unique}

3.4. Une approche d'alignement entre l'architecture et le contexte d'entreprise

[Wieringa03] suggère qu'il n'y a pas un unique problème d'alignement, mais trois, entre le monde social, le monde linguistique et le monde physique. Le monde physique est le monde des entités qui « ont un poids, consomment de l'énergie et produisent de la chaleur et du bruit ». Le monde social est le monde des processus d'entreprise, des besoins, de la valeur ajoutée, de l'argent, des normes. Une partie du monde social est le monde linguistique de la manipulation des symboles ; c'est le monde du logiciel et des documents.

Les auteurs définissent un cadre afin de construire une situation où ces trois mondes et les différents éléments qui les composent sont alignés. La Figure 11 montre ce cadre. Il liste des descriptions des différents éléments qui servent traditionnellement à modéliser des processus d'entreprise ou à concevoir des systèmes d'information. Il est important de noter que toutes ces descriptions ne doivent pas forcément être réalisées pour chacun des systèmes. Au contraire, chaque projet particulier produit une partie de ces descriptions. Ce cadre pose le paradoxe suivant :

- pour réaliser une description d'une entité d'un niveau donné, les descriptions des niveaux supérieurs sont nécessaires.
- pour affiner une description d'un niveau donné, les descriptions du niveau inférieur sont nécessaires.

Ce paradoxe est résolu (1) en réalisant ensemble les descriptions d'une colonne du cadre et (2) en développant un affinement d'une description uniquement après avoir identifié les descriptions de services de niveau inférieur nécessaire à cet affinement.

Cette approche est donc mixte : si globalement la démarche est top-down, localement, elle est bottom-up pour permettre un affinement nécessaires à la poursuite de l'approche.

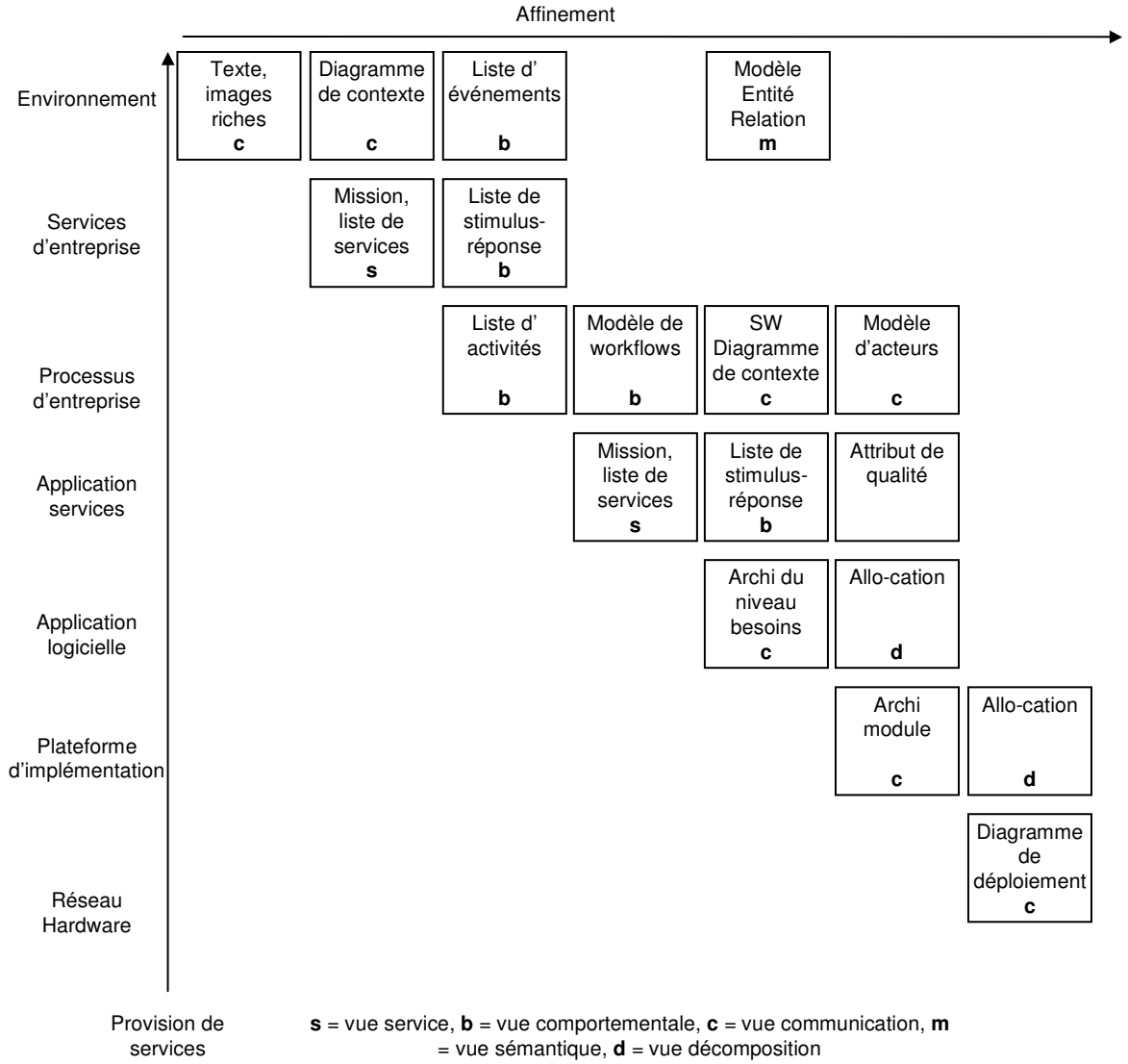


Figure 11 : Cadre permettant la conception d'un alignement entre les trois mondes

Il est important de noter que cette approche a pour but de construire un alignement, ce qui peut se faire par la modification des entités existantes. Ce sont bien des problèmes de création de l'alignement et non d'évolution ou de maintien de cette relation face aux changements qui est traitée par cette approche car l'alignement n'existe pas ou est inconnu dans la situation de départ.

Ce cadre aide à résoudre les problèmes liés à l'alignement en fournissant des indications simples de conception. Parmi ces indications, on peut par exemple citer :

- décomposition fonctionnelle : pour chaque service à délivrer, définir un composant.
- décomposition orientée communication : pour chaque communication avec une entité externe, définir un composant. Cette directive a trois variantes : (i) orientée système, pour chaque système communicant avec l'entité, définir un composant qui gère cette communication ; (ii) orientée acteur, pour chaque acteur communicant avec l'entité, définir un composant qui gère cette communication ; et (iii) orientée événement, pour chaque événement auquel l'entité doit répondre, définir un composant.
- décomposition orientée comportement : pour chaque processus de l'environnement qui est contrôlé, définir un composant système.
- décomposition orientée sujet : pour chaque partie du monde pour laquelle des données doivent être maintenues, définir un composant système.

Cette approche se positionne donc par rapport à notre cadre de référence de la façon suivante:

Vue Objet

Nombre d'entités : 5

Nature des entités : {processus d'entreprise, système, environnement, logiciel, code}

Relation entre entités : typologie de règles

Vue But

But : {construire}

Vue Méthode

Méthode de construction : mixte

Vue Outil

Outil : {règles de dérivation}

3.5. Une approche d'alignement orientée-besoins

[Bleistein05a] et [Bleistein05b] cherchent à aligner la stratégie d'entreprise au système. Pour cela, ils proposent une démarche d'ingénierie des exigences qui unifie dans un même modèle (1) les objectifs stratégiques de l'organisation et (2) les activités et les processus par lesquels ces objectifs sont réalisés. Cette approche repose sur la modélisation de stratégie d'entreprise par des buts et la définition du contexte physique au moyen des *problem frame* définis par Jackson [Jackson01].

Les *problem frames* captent, structurent et classent des problèmes liés au développement de logiciel selon un cadre de diagramme de problème (*problem diagram*). Un diagramme de problème est constitué de deux parties : une partie besoins et un diagramme de contexte de domaine car, selon Jackson, un besoin ne peut être compris que dans le contexte dans lequel il existe. Ainsi, les besoins et les domaines de contexte sont liés par des liens de référence ou de contrainte.

Les besoins du niveau stratégique sont trop abstraits pour concevoir ou implémenter une solution système. Les besoins sont donc affinés grâce au concept de progression entre problèmes. Ce concept s'appuie sur la notion de projection qui fait référence à la capacité de

décrire un contexte de domaine selon plusieurs points de vue, niveaux d'abstraction et degrés de détail.

La Figure 12 illustre une progression de diagrammes de problèmes. Les ovales représentent les besoins RA, RB, RC, RD, RM qui font respectivement référence au diagramme de contexte DA, DB, DC, DD, M où M est la machine et correspond aux logiciel, matériel, données, et ressources réseau Le contexte de domaine DA correspond au niveau de l'entreprise. RA correspond aux stratégies d'entreprise. Le contexte de domaine DA est affiné. Une analyse de DA et RA permet d'identifier un besoin RB qui se réfère à DB tout en satisfaisant RA.

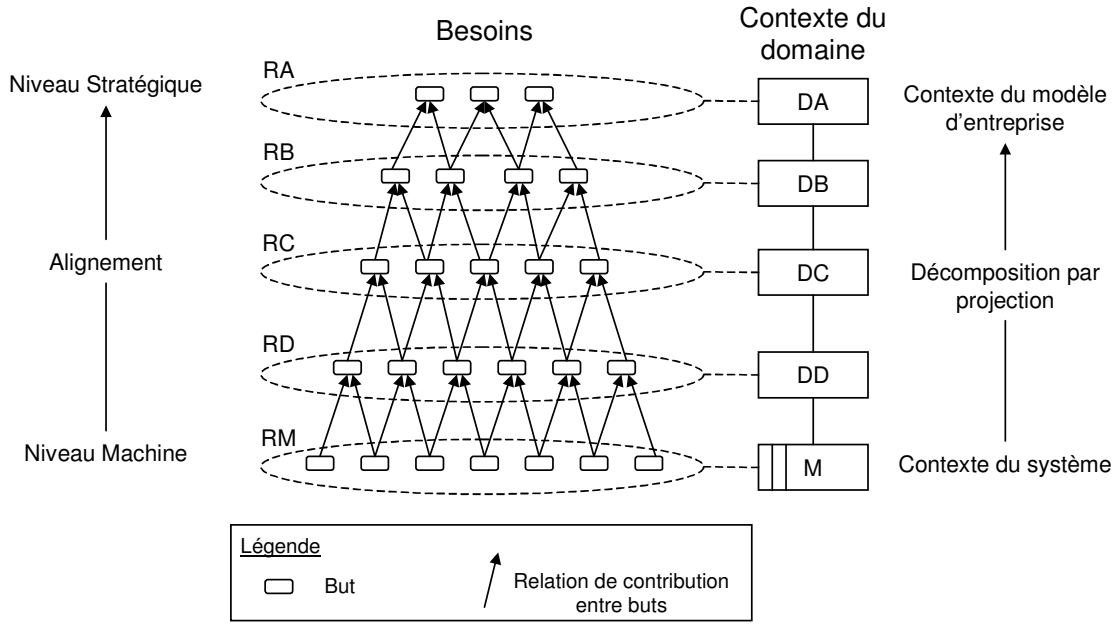


Figure 12 : Intégration d'un modèle de but et d'une progression de problèmes

Afin d'établir des liens explicites et directs entre besoins et diagrammes de problèmes, les auteurs adoptent une modélisation orientée but pour représenter les besoins. Les buts peuvent être formulés à différents niveaux d'abstraction d'un haut niveau stratégique à un bas niveau technique. A la Figure 12, à chaque niveau d'abstraction, les besoins sont décrits comme une portion plus large d'un modèle de but. Les sous-buts sont des projections de leur but père et la satisfaction des sous-buts assure la satisfaction des buts pères.

Les auteurs utilisent le model BRG (Business Rules Group) [Kolber00] pour organiser les stratégies d'entreprises. Ce modèle est un cadre conceptuel constitué de deux concepts majeurs :

- les finalités qui sont les choses que l'entreprise souhaite atteindre comme (i) une vision, qui correspond à une image globale de ce que l'entreprise veut être ou devenir ; (ii) un but, i.e. un état de l'entreprise qui indique ce qui doit être satisfait pour effectivement atteindre la vision ; ou (iii) un objectif, c'est-à-dire une cible mesurable que l'entreprise cherche à atteindre pour satisfaire un but.

- les moyens qui sont les choses que l'entreprise utilise pour atteindre ces finalités comme (i) une mission qui est la première activité réalisée pour atteindre une vision ; (ii) une stratégie, c'est-à-dire un composant de la mission qui permet d'atteindre un but ; ou (iii) une tactique, qui aide à implémenter une stratégie et permet de réaliser un objectif.

Ce modèle ne propose pas de langage spécifique de représentation. Pour palier à ce problème, Bleistein *et al*. [Bleistein05a] proposent de faire correspondre le modèle BRG et i* permettant ainsi de rendre opérationnel ce cadre conceptuel. Ceci permet également d'enrichir le modèle i* de règles sémantiques pour construire des modèles de buts et de guidage pour l'analyse organisationnelle.

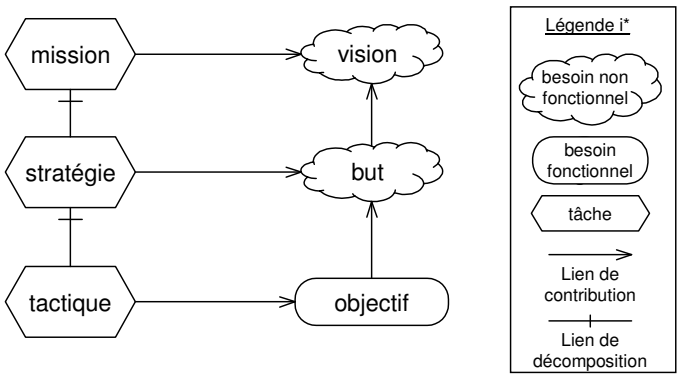


Figure 13 : Correspondances entre le modèle BRG et la notation i*

La Figure 13 illustre les correspondances entre le modèle BRG et le modèle i*. Les concepts de mission, stratégie et tactiques sont chacun équivalents au concept de tâche du modèle i* représenté par un hexagone. Les concepts de vision et de but du modèle BRG peuvent être considérés comme des besoins non fonctionnels représentés par des nuages. Enfin, un objectif est un besoin fonctionnel car tous deux sont concrets et mesurables. Les liens entre tâches sont des liens de décomposition. Les autres liens sont des liens de contribution.

Le modèle BRG, est habituellement séparé des modèles de besoins ou des spécifications du système mais sert de guide pour organiser d'autres types de modélisation ou de documentation. Selon les auteurs, la mise en relation d'un modèle organisationnel et d'un modèle de représentation du système engendre souvent une perte d'information ainsi qu'une rupture de l'alignement. L'application de i* au modèle BRG permet de résoudre ce problème en unifiant le modèle de stratégie d'entreprise et le modèle des besoins. Les buts de la Figure 12 sont en fait représentés en utilisant la notation i* par application de la sémantique du modèle BRG.

Cette approche permet de construire un système aligné avec la stratégie d'entreprise et les processus d'entreprise.

Vue Objet

Nombre d'entités : 2

Nature des entités : {stratégie d'entreprise, système}

Relation entre entités : typologie de liens

Vue But

But : {construire}

Vue Méthode

Méthode de construction : Top-down

Vue Outil

Outil : {règles de dérivation, langage unique}

3.6. Une approche de co-évolution reposant sur l'évaluation d'un mauvais alignement

Cette approche repose sur l'idée que la modification d'un objet peut avoir un impact sur d'autres objets [Bodhuin04]. Ces impacts peuvent d'une part rompre la relation d'alignement entre le système et les processus d'entreprise et d'autre part servir de support aux actions pour rétablir l'alignement. Les auteurs proposent une stratégie pour (i) détecter une rupture de l'alignement entre les processus d'entreprise et des systèmes qui les supportent et (ii) identifier les objets à changer pour rétablir l'alignement. La Figure 14 illustre la stratégie d'alignement proposée.

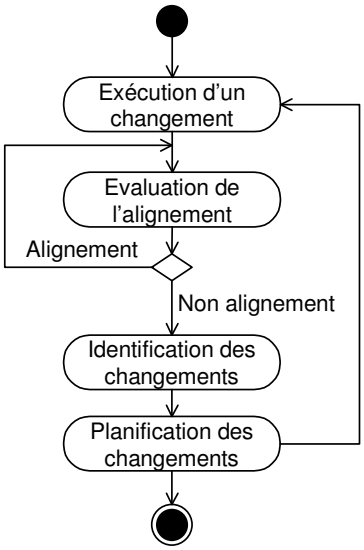


Figure 14 : Stratégie d'alignement représentée avec un diagramme d'activités UML

La phase d'exécution d'un changement consiste à mettre en œuvre les changements pour (i) faire évoluer le système ou les processus ou (ii) corriger une rupture de l'alignement.

La phase d'évaluation de l'alignement a pour but d'identifier les ruptures d'alignement résultant des différents changements réalisés. Pour cela, les auteurs ont identifié un ensemble de paramètres et leur ont assigné des seuils. Il est ainsi possible de mesurer l'alignement et de détecter une rupture si la valeur obtenue pour le paramètre se situe en dessous de ce seuil. Le taux de couverture technologique qui correspond au pourcentage d'activités des processus d'entreprise adéquatement supportées par le système est un exemple de paramètre.

La phase d'identification des changements n'a lieu que si la phase précédente a détecté une rupture de la relation d'alignement. Dans ce cas seulement des actions correctives doivent être entreprises pour rétablir l'alignement. Cette phase préconise une analyse d'impact pour déterminer les répercussions d'un changement sur d'autres éléments. Une liste de liens de dépendance entre éléments des processus ou du système ou uniquement entre des éléments du système a été identifiée, comprenant, par exemple, les liens :

- *utilise (processus, système)* pour préciser qu'un processus peut être supporté par un ou plusieurs systèmes ; ou

- *dépend_de (activité1, activité2)* qui spécifie qu'une activité dépend d'une autre par exemple dans le cas où l'exécution de la deuxième activité ne peut pas avoir lieu tant que la première n'a pas été achevée.

Un certain nombre de types de modifications sont définis par ailleurs pour indiquer la façon dont un objet peut être modifié. Par exemple, une activité peut être modifiée soit en changeant la façon dont elle est exécutée, soit en changeant ses modalités d'interactions avec d'autres activités. Enfin, des règles de propagation ont été spécifiées. Elles permettent de décrire quand et comment propager des modifications entre objets connectés. Ainsi, par exemple, si la façon d'exécuter une activité est modifiée, le changement peut avoir un impact sur les composants du système qui la supportent.

La dernière phase de planification des changements correspond au choix des innovations à réaliser en utilisant, par exemple, des règles de priorisation.

Cette approche se positionne par rapport au cadre de la façon suivante :

Vue Objet
Nombre d'entités : 2
Nature des entités : {processus d'entreprise, système}
Relation entre entités : typologie de liens
Vue But
But : {évaluer, évoluer}
Vue Méthode
Méthode d'évaluation : quantitative
Méthode d'évolution : {correction}
Vue Outil
Outil : {métriques, règles de dérivation}

3.7. Une approche de co-évolution reposant sur l'impact de changement

Les auteurs proposent d'établir une correspondance entre des exigences modèlisées avec i* et des spécifications du système écrites en Z [Krishna04a] et [Krishna04b]. Les auteurs modélisent les contextes, les intentions et la logique organisationnelles avec i* alors que les spécifications des fonctionnalités et la conception du système utilisent la notation formelle Z.

La relation d'alignement entre les besoins modélisés en i* et les spécifications Z du système se base sur des correspondances entre concepts [Vilkomir04], [Krishna04a] et [Krishna04b]. Il est ainsi possible d'établir une correspondance, par exemple entre un acteur et une tâche ou un lien de décomposition de tâches et des éléments de la spécification écrite en langage Z.

Ces correspondances permettent de construire des spécifications Z alignées avec le contexte organisationnel modélisé avec i*. Elles sont également utiles pour maintenir la relation d'alignement face aux changements.

Les auteurs s'intéressent également à la co-évolution des exigences et du système. Pour cela, ils définissent des règles pour répercuter les changements d'un modèle i*, sur les

spécifications Z. Seize types de changement d'un modèle i* ont été identifiés : l'ajout et la suppression des liens de dépendances, des tâches, des buts, des ressources, des besoins non-fonctionnels, des liens de contribution, des liens de décomposition de tâches et des acteurs. Par exemple, l'ajout d'une tâche entraîne, entre autres conséquences, (i) la création d'un nouvel élément dans la spécification Z, (ii) l'introduction du nom de la tâche dans les schémas de buts, ressources ou besoins non-fonctionnels liés à cette tâche, (iii) l'ajout du nom de la tâche dans l'élément de schéma Z correspondant à l'acteur.

La Figure 15 schématise l'approche de co-évolution décrite dans [Krishna04a]. La spécification Z est obtenue par application des règles de correspondance entre les concepts i* et ceux de Z. La spécification Z étendue est obtenue par affinement du schéma structurel à partir d'informations complémentaires qui ne sont pas inclues dans un modèle i* mais acquises par de plus amples analyses.

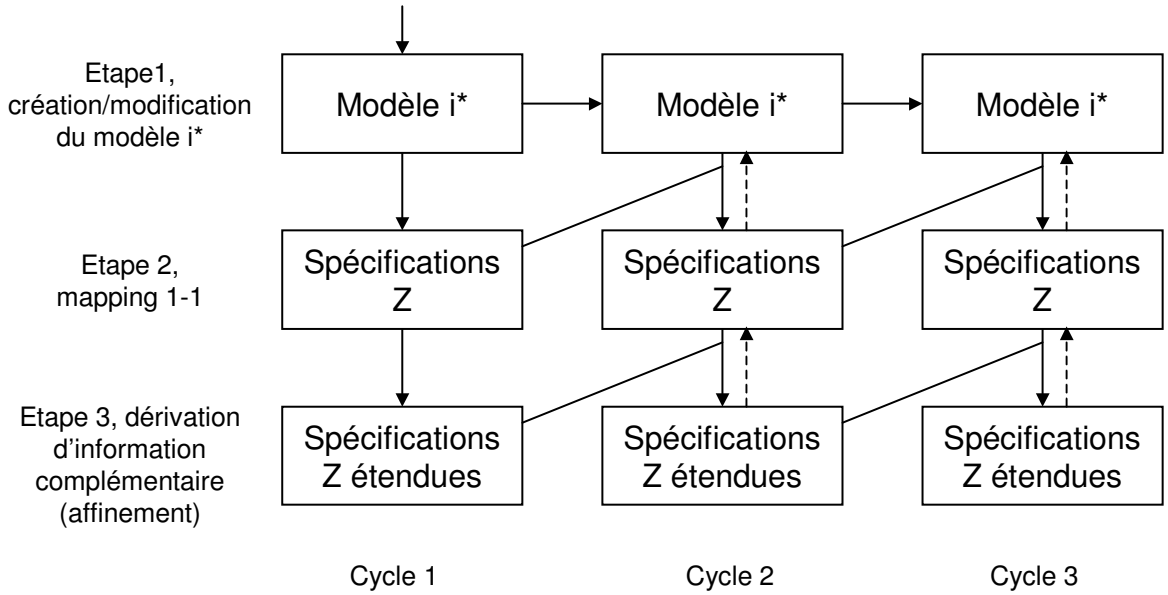


Figure 15 : Co-évolution de modèles i* et de spécifications Z

Les auteurs soulignent qu'une correspondance inverse entre une collection de spécifications Z et un modèle i* est possible. La répercussion des changements peut donc se faire de Z vers i*, aucune indication concrète n'est cependant offerte.

Cette approche peut donc se positionner dans notre cadre de référence de la manière suivante :

Vue Objet

Nombre d'entités : 2

Nature des entités : {système, exigence}

Relation entre entités : typologie de liens

Vue But

But : {construire, évoluer}

Vue Méthode

Méthode de construction : Top-down

Méthode d'évolution : {double dépendance}

Vue Outil

Outil : {règles de dérivation}

4. Résumé de l'évaluation

Le cadre de référence que nous proposons permet d'étudier différents aspects de l'alignement. Il est constitué de quatre vues qui correspondent à quatre perspectives selon lesquelles on peut étudier l'alignement : le quoi, le pourquoi, le comment et le moyen. Dans chacune de ces vues sont définis des attributs qui précisent les caractéristiques de l'alignement. Nous avons illustré le cadre en étudiant et en positionnant sept approches représentatives de l'état de l'art. Le Tableau 3 synthétise les résultats de cette analyse.

	[Luftman00]	[Landtsheer]	[Wegmann05]	[Wieringa03]	[Bleistein05]	[Bodhuin04]	[Krishna04a]
Nombre d'entités	2	2	3	5	2	2	2
Nature des entités	stratégie d'entreprise, stratégie des TI	système, exigence	processus d'entreprise, système, environnement	processus d'entreprise, code, système, logiciel, environnement	stratégie d'entreprise, système	processus d'entreprise, système	système, exigence
Relation entre entités	pas défini	règles	liens	règles	liens	liens	liens
But	évaluer, évoluer	construire	construire	construire	construire	évaluer, évoluer	construire, évoluer
Méthode d'évaluation	qualitative	-	-	-	-	quantitative	-
Méthode de construction	-	Top-down	Top-down	Mixte	Top-down	-	Top-down
Méthode d'évolution	correction	-	-	-	-	correction	double dépendance
Outil	cadre	règles de dérivation	règles de dérivation, langage unique	règles de dérivation	règles de dérivation, langage unique	métriques, règles de dérivation	règles de dérivation

Tableau 3 : Résumé de l'évaluation des sept approches

Comme le montre le Tableau 3, l'alignement met en jeu le plus souvent deux entités mais davantage peuvent être impliquées comme dans [Wegman05a] et [Wieringa03].

Ces entités sont très variées d'une approche à l'autre : [Krishna04a] et [Landtsheer04] s'intéressent à l'alignement du système et des exigences, [Bodhuin04] à l'alignement du système et des processus d'entreprise, [Bleistein05] à l'alignement du système et des stratégies d'entreprise et [Luftman00] à l'alignement des stratégies d'entreprise et des stratégies des technologies de l'information. [Wegmann05a] étudie l'alignement entre le système, les processus et l'environnement. [Wieringa03] considère l'alignement entre les processus d'entreprise, le système, l'environnement, le logiciel et le code.

Le but des approches d'alignement varie sans prédominance particulière de la construction [Bleistein05], [Landtsheer04], [Wegmann05], [Wieringa03] de l'évaluation [Bodhuin04], [Luftman00] ou de l'évolution d'un alignement [Bodhuin04], [Krishna04a] et [Luftman00].

La majorité des approches détaillées propose une méthode pour construire un système aligné avec d'autres entités. Ces approches sont majoritairement "Top-down" et utilisent des règles de dérivations pour passer de niveaux d'abstraction élevés aux niveaux plus bas [Bleistein05], [Landtsheer04], [Wegmann05]. Les règles de dérivation sont parfois utilisées conjointement avec un modèle unique comme dans [Bleistein05] et [Wegmann05].

Deux approches s'intéressent à l'évaluation de l'alignement, mais seul [Bodhuin04] propose une évaluation objective. Dans la littérature, l'approche de Bodhuin *et al.* est d'ailleurs l'une des seules qui propose un système pour mesurer l'alignement, alors que de nombreux auteurs [Soffer04b] [Giaglis99] soulignent l'importance de cette problématique dans la

compréhension de l'alignement. [Bodhuin04] ne spécifie cependant que deux métriques, ce qui peut paraître insuffisant et laisse entrevoir l'utilité d'une approche traitant plus complètement cet aspect.

Concernant l'évolution de l'alignement, deux approches s'appuient sur l'évaluation de l'alignement pour : mettre en œuvre des actions correctives [Bodhuin04] et [Luftman00] proposer des axes d'amélioration pour atteindre le niveau supérieur de maturité d'alignement..

CHAPITRE 3 : MESURER L'ALIGNEMENT

1. Introduction

Comme nous l'avons vu aux deux chapitres précédents, la notion d'alignement n'est pas clairement définie. Les définitions proposées sont souvent informelles, vagues et incomplètes, ce qui rend difficile tout raisonnement autour de ce concept. Il est par exemple difficile d'identifier de façon objective un non alignement entre les processus d'entreprise et le système qui les supporte et d'en comprendre les causes.

Ce chapitre propose tout d'abord une définition précise de l'alignement qui se veut objective. Nous proposons de définir le concept d'alignement non pas directement entre les entités elles-mêmes (système et processus d'entreprise) mais entre des modèles qui représentent ces entités.

Nous définissons ensuite dix métriques entre éléments de méta-modèles, qui permettent de mesurer l'alignement entre les processus d'entreprise et le système, d'évaluer le degré d'alignement et d'apprécier les nuances entre parfaitement aligné et pas du tout aligné.

Afin d'éviter de redéfinir des métriques pour chaque projet ou pour chaque couple de méta-modèles, nous définissons des métriques d'alignement génériques entre des méta-modèles génériques. Nous proposons également un processus permettant de construire, à partir de ces métriques génériques, les métriques spécifiques pour tout couple constitué d'un méta-modèle spécifique de processus et d'un méta-modèle spécifique de système.

Ce chapitre est organisé de la façon suivante : la section 2 définit le concept d'alignement ; la section 3 propose des métriques pour évaluer le degré d'alignement, la section 4 présente et illustre le processus de génération de métriques spécifiques, la section 5 est consacrée à l'introduction de deux notions, le seuil et le poids, dans l'expression des métriques. La section 6 est une conclusion de ce chapitre.

2. Définition de l'alignement

La définition de l'alignement que nous proposons repose sur une analyse de la littérature.

Nos travaux s'inscrivent dans la lignée des recherches entreprises au Centre de Recherche en Informatique sur la modélisation et la méta-modélisation [Deneckère01], [Schwer99] ainsi que celles de l'OMG (Object Management Group) sur UML [UML] et le MOF [MOF]. A la suite de ces travaux, il nous paraît pertinent de définir la notion d'alignement entre des modèles qui représentent, respectivement, le système et le processus. Cette façon de procéder permet de se concentrer sur les concepts et non sur les instances. Dans la mesure où l'utilisation de modèles est largement répandue dans le monde industriel, cette approche, bien que fondée sur des abstractions, nous semble tout à fait adaptée pour résoudre des problèmes concrets.

Dans le reste de ce chapitre, nous utilisons les niveaux d'abstraction de modèle et de méta-modèle tels qu'ils sont définis par l'OMG. Ainsi, un modèle est une aggrégation de méta-données qui permettent de décrire les données du niveau instance et les liens qui existent entre elles. Un méta-modèle décrit la structure et la sémantique des méta-données ; il rassemble donc les méta-méta-données [MOF]. Un modèle est une instance de méta-modèle.

Nous proposons de définir la notion d'alignement entre les processus d'entreprise et le système comme une correspondance entre des éléments du modèle de processus et des éléments du modèle du système.

Une définition formelle de la notion d'alignement suppose d'expliciter les types de liens qui peuvent exister entre les différents éléments de ces modèles : à partir de typologies de liens proposées dans la littérature, nous proposons de définir, dans une première sous-section, deux types de liens : *correspond* et *représente*. Ces deux liens nous permettent, dans une seconde sous-section de proposer une définition formelle de la notion d'alignement.

2.1. Définition des liens *correspond* et *représente*

Les deux types de liens que nous définissons s'appuient sur des types identifiés dans la littérature. La première partie de cette section décrit les différents liens qui nous ont servi de support à la définition des liens *correspond* et *représente*. Les deux parties suivantes définissent et décrivent chacun de ces deux liens : le type de lien *correspond* exprime une égalité entre des concepts similaires de méta-modèles différents. Le type de lien *représente* veut dire qu'un concept d'un modèle a un impact sur un concept de l'autre modèle. Enfin, dans une quatrième partie, nous essayons de montrer comment s'articulent les liens *correspond* et *représente*.

2.1.1 Aperçu de différentes typologies de liens utilisées dans la littérature

Plusieurs auteurs se sont intéressés à la définition de liens entre éléments de méta-modèle. Nous en avons retenu quatre :

1. [Ralyté04] définit de façon générique les liens qui peuvent exister entre des éléments appartenant à des méta-modèles différents ;
2. [Polh96] décrit au sein d'un méta-modèle les types de dépendances qui peuvent exister entre différents éléments ;
3. [Bunge77] spécifie formellement deux notions permettant d'établir des liens de dépendance ;
4. [Bodhuin04] propose cinq liens de dépendance entre éléments du méta-modèle de système et du méta-modèle de processus.

Chacun des quatre paragraphes suivants précisent ces différents liens.

Ralyté [Ralyté04] définit le concept de *modelElement* comme le lien entre un élément et le méta-modèle auquel il appartient. Ce concept permet de modéliser la relation entre des

éléments de différents méta-modèles. Cette relation peut être soit une *association*, soit une *composition*, soit un lien d'*héritage*. Dans notre cas, dans la mesure où les deux méta-modèles permettent de représenter respectivement les processus d'entreprise et le système, il apparaît clairement que les liens entre les concepts des deux méta-modèles ne peuvent pas être des liens de composition ou d'héritage. Cependant cela ne signifie pas que les liens entre les concepts des deux méta-modèles sont des liens d'association dans le sens le plus courant du terme. En effet, la relation d'association telle qu'elle est définie dans [Ralyté04] est générale, elle peut être spécialisée.

Pohl [Pohl96] définit une typologie de dépendances pouvant exister entre des concepts, (par exemple agent, produit, état, étape de processus, besoin…). Dix-huit types de dépendances sont identifiés et classés en cinq catégories : *condition*, *contenu*, *document*, *évolution* et *abstraction*. Parmi ces types de dépendance, deux nous paraissent pertinents pour la définition du concept d'alignement : le lien '*similaire'* qui permet d'exprimer une similarité entre objets et le lien '*repose sur*' qui spécifie qu'un objet est influencé par la définition d'un autre objet. Les autres liens décrivent des relations de contraintes, de conflits, d'abstraction, de documentation pour tracer un objet d'information… Ils ne nous semblent pas utiles pour exprimer une correspondance entre les concepts des deux méta-modèles en vue de définir ou de mesurer l'alignement.

Bunge [Bunge77] définit la notion de *couplage* et la notion de *similarité*. Ces deux notions sont spécifiées pour les éléments de l'ontologie de Bunge. L'élément central de cette ontologie est celui de *chose* dont le monde est fait. Deux choses sont couplées si l'existence d'une chose affecte l'histoire d'une autre, c'est-à-dire l'ordre des états traversés par cette chose. La similarité entre deux choses correspond à l'intersection des ensembles des propriétés des deux choses. Cette notion ne relie pas deux choses similaires mais permet d'exprimer le degré de similarité entre ces deux choses. Ces liens sont définis par Bunge à un niveau instance mais ont été utilisés par Chidamber *et al.* à un niveau type pour servir de base à la définition de critères de mesure pour la modélisation orientée objet [Chidamber94].

Bodhuin [Bodhuin04] propose cinq types de dépendances entre composants du système et activités des processus. Ainsi, il définit les liens *inclut* et *composé_de* afin de spécifier respectivement les activités qui constituent un processus et les composants qui composent un système logiciel. Deux liens *dépend_de* sont définis permettant d'identifier un lien de dépendance (i) entre activités quand, par exemple, une première activité fournit de l'information à une deuxième ; cette dernière ne pouvant démarrer si la première n'est pas complétée et (ii) entre composants de systèmes, quand un composant utilise un autre composant, ou s'il existe un lien d'héritage ou de composition entre eux. Enfin, Bodhuin définit le lien *utilise* qui permet de spécifier qu'un processus d'entreprise est supporté par un ou plusieurs systèmes.

Deux types de lien émergent de ces typologies : (i) ceux qui permettent d'exprimer une similarité entre concepts et (i) ceux spécifiant une dépendance entre concepts. A partir de ces deux types de liens, nous avons défini les liens *correspond* et *représente* que nous utilisons pour définir le concept d'alignement.

2.1.2 Lien *correspond*

Il n'est pas rare que deux méta-modèles permettant de modéliser des entités différentes aient un certain nombre de concepts communs. Ainsi, par exemple, un méta-modèle de système comporte des classes d'objets ayant des attributs et des états, ces derniers changeant au cours du temps. Ces deux concepts peuvent aussi être des éléments d'un méta-modèle de processus. Il est ainsi possible de définir des correspondances entre des concepts de même nature mais appartenant à des méta-modèles différents. Le type de lien *correspond* exprime une égalité entre des concepts similaires de méta-modèles différents.

Le lien *correspond* (i) est à la fois proche du lien 'similaire' de Pohl et de la notion de similarité de Bunge et (ii) est une spécialisation du lien d'Association défini dans [Ralyté04]. En effet, comme le lien 'similaire' de Pohl, le lien *correspond* permet de relier deux concepts similaires. Nous utilisons la notion de similarité de Bunge qui exprime le degré de similarité entre deux choses pour pouvoir dire formellement si deux concepts se *correspondent*.

Soit MM_s un méta-modèle de système et MM_p un méta-modèle de processus d'entreprise, on dit qu'un concept X ($X = \{x_1, x_2, \ldots x_n\}$) du méta-modèle MM_s *correspond* ($\mathcal{M}$) (map en anglais) à un concept Y ($Y = \{y_1, y_2, \ldots y_n\}$) du méta-modèle MM_p si quelque soit i entier naturel, il existe des isomorphismes h_i tels que $x_i = h_i(y_i)$ (qu'on écrit par simplicité $x_i \cong y_i$). L'utilisation d'un isomorphisme permet (1) de surmonter les problèmes liés au fait que x_i et y_i peuvent appartenir à des domaines différents et (2) de définir une relation symétrique grâce au caractère bijectif de l'isomorphisme.

$$X \mathcal{M} Y \Leftrightarrow x_i \cong y_i, \forall i \in [\![1, n]\!]$$

Cette relation $\mathcal{M}$ est symétrique. Ainsi, si un concept X *correspond* à un concept Y, alors Y *correspond* à X et réciproquement.

Si le modèle de système M_s instancie le méta-modèle de système MM_s et si le modèle de processus M_p instancie le méta-modèle de processus MM_p, il est possible de définir un lien *correspond* entre un concept de M_s et un concept de M_p. Ainsi, on dit qu'un concept c_s ($c_s = \{c_{s1}, c_{s2}, \ldots c_{sn}\}$) du modèle de système M_s *correspond* au concept c_p ($c_p = \{c_{p1}, c_{p2}, \ldots c_{pn}\}$) du modèle de processus M_p si :

1. c_s est une instance de C_s concept du méta-modèle de système MM_s, c_p est une instance de C_p concept du méta-modèle de processus MM_p et C_s *correspond* à C_p et
2. $c_{si} \cong c_{pi}$ pour tout i entier naturel compris entre 1 et n.

2.1.3 Lien *représente*

Le type de lien *représente* spécifie qu'un concept d'un méta-modèle a un impact sur un concept d'un autre méta-modèle.

Le lien *représente* est une spécialisation du lien d'Association entre éléments de méta-modèles différents défini dans [Ralyté04]. Ce lien s'appuie sur la notion de couplage de Bunge, sur les liens 'repose sur' de Pohl, 'dépend_de' et 'utilise' de Bodhuin.

Nous ne nous intéressons qu'au lien représente entre un méta-modèle de système MM_s et un méta-modèle de processus d'entreprise MM_p. On considère qu'un concept X du méta-modèle MM_s *représente* ($\Re$) un concept Y du méta-modèle MM_p si l'existence du premier affecte le comportement, la valeur ou l'existence du deuxième (ce qu'on notera $X \triangleright Y$).

$$X \Re Y \Leftrightarrow X \triangleright Y.$$

De la même façon que pour le lien *correspond*, il est possible de définir le lien *représente* entre deux concepts de modèle de système et de processus. Un concept c_s d'un modèle de système M_s *représente* un concept c_p d'un modèle de processus M_p si :

- c_s est une instance de C_s concept du méta-modèle de système MM_s, c_p est une instance de C_p concept du méta-modèle de processus MM_p et C_s *représente* C_p et
- c_s affecte le comportement, la valeur ou l'existence de c_p.

2.1.4 Articulation entre *correspond* et *représente*

Le lien *correspond* exprime une relation forte entre deux concepts en imposant l'égalité de leurs propriétés (à un isomorphisme près). En revanche, le lien *représente* permet de préciser une relation moins restrictive entre deux concepts car seul le fait que l'un affecte le comportement, la valeur ou l'existence du deuxième est pris en compte.

Ainsi, un lien *correspond* ne peut exister qu'entre deux concepts de même nature alors que deux concepts de nature différente peuvent être liés par un lien *représente*.

Il découle des définitions des deux liens que si deux concepts sont liés par le lien *correspond*, alors ils sont également liés l'un à l'autre par un lien *représente*. L'inverse n'est pas vrai.

2.2. Définition de la relation d'alignement

Les liens *correspond* et *représente* permettent d'établir une correspondance entre les concepts de deux méta-modèles différents. Afin de définir la notion d'alignement entre un système et des processus d'entreprise, il est nécessaire de préciser (i) les liens *correspond* qui existent entre des éléments de même nature du méta-modèle du système et du méta-modèle du processus et (ii) les liens *représente* qui existent entre des éléments de ces deux méta-modèles.

Toutefois, il n'apparaît pas pertinent de préciser tous les liens *représente* : il convient d'exclure les cas pour lesquels il existe déjà un lien *correspond*. En effet, si on gardait tous les liens *représente*, il se présenterait des cas de figure où l'information ne serait pas gérée de la même façon dans le système et les processus. Intuitivement, l'alignement ne peut être parfait dans ce cas de figure.

Supposons, par exemple, que l'on dispose de certains objets des processus *représentés* par des propriétés dans le système. Un objet peut changer d'états mais pas une propriété. Si la définition de l'alignement englobe les liens *représente* entre un objet des processus et une propriété du système, alors il est possible que l'alignement parfait soit atteint alors que l'information n'est pas gérée de la même façon dans les processus et dans le système. Il y a donc une contradiction. Eliminer les cas où il existe déjà un lien *correspond* permet de résoudre cette contradiction.

Nous pouvons ainsi proposer la définition suivante :

Définition littéraire :

On appelle relation d'alignement entre un modèle de système et un modèle de processus l'ensemble :

- des liens *correspond* entre un élément du modèle de système et un élément du modèle de processus et
- des liens *représente* entre un élément du modèle de système et un élément du modèle de processus ; tels qu'il n'existe pas de lien *correspond* faisant intervenir au moins l'un des concepts du méta-modèle de système ou du méta-modèle de processus que ces éléments instancient.

Soit

- $\mathcal{L}$, un lien ;
- M_s, un modèle de système ;
- M_p, un modèle de processus ;
- a, un élément de M_s ;
- b, un élément de M_p ;

Il existe un lien *correspond* si *a correspond* à *b* ($a \mathcal{M} b$) et un lien *représente* si *a représente b* ($a \Re b$).

On considére que a instancie le concept c_a du méta-modèle de système ($a = \Im(c_a)$) et que b instancie le concept c_b du méta-modèle de processus ($b = \Im(c_b)$). On ne s'intéresse au lien *représente* entre a et b que si c_a ne *correspond* pas à c_b et s'il n'existe pas de concept c_i appartenant au méta-modèle de processus MM_p *correspondant* à c_a ni de concept c_j appartenant au méta-modèle de système MM_s *correspondant* à c_b.

Ainsi, on considère que a et b sont liés ($a \mathcal{L} b$) si *a correspond* à *b* ou *a représente b* avec la restriction précisée si dessus. Ce qui peut s'écrire :

$$a \mathcal{L} b = a \mathcal{M} b \vee [a \Re b \wedge [a = \Im(c_a) \wedge b = \Im(c_b) \wedge \neg (c_a \mathcal{M} c_b) \wedge (\nexists c_i \in MM_p, c_i \mathcal{M} c_a) \wedge (\nexists c_j \in MM_s, c_b \mathcal{M} c_j)]]$$

Définition formelle :

La relation d'alignement $\mathcal{A}$ eentre un modèle de processus d'entreprise et un modèle de système est l'ensemble des liens $\mathcal{L}$, tel que quelque soit b un élément du modèle de processus, il existe a un élément du modèle de système qui est lié à b par $\mathcal{L}$. Ce qui s'écrit :

$$\mathcal{A} = \{\mathcal{L} \mid \forall\, b \in M_p, \exists\, a \in M_s \wedge a\ \mathcal{L}\ b\}$$

Cet ensemble correspond à l'ensemble maximal, c'est-à-dire l'alignement parfait. On considère que l'alignement n'est plus parfait dès que l'un des liens entre ces concepts n'existe plus. Dans ce cas, un concept de M_p n'est pas lié à un concept du système ni par un lien *correspond* ni par un lien *représente*. Ce concept n'est alors pas pris en charge par le système.

On peut mesurer le degré d'alignement en comparant (i) l'ensemble des liens qui existent entre les éléments du modèle de processus et ceux du modèle de système avec (ii) cet ensemble maximal. Si les deux ensembles sont les mêmes l'alignement est parfait, sinon, il n'est que partiel.

On réduit par exemple les liens entre les concepts du diagramme d'activités UML pour représenter les processus et ceux du méta-modèle Entité Relation pour modéliser le système au seul lien *correspond* entre les classes des objets apparaissant dans le diagramme d'activités et des entités types E/R. L'alignement entre le modèle de système et le modèle de processus est l'ensemble des liens *correspond* tel que quelque soit *c* la classe UML, il existe *e* une entité-type E/R telque *e correspond* à *c*. S'il existe une classe *c'* qui n'est pas liée à une Entité Type *e* cela signifie que l'information n'est pas gérée de la même façon dans les processus et le système : l'alignement est rompu.

3. Proposition de métriques pour évaluer le degré d'alignement

Dans cette section, nous proposons des métriques permettant d'évaluer le degré d'alignement entre un modèle de processus d'entreprise et un modèle de système. Les métriques sont définies entre les éléments de ces modèles. La qualité des métriques dépend donc de la pertinence des modèles et de la confiance que l'on peut avoir en eux. Nous sommes conscients que de tels modèles n'existent pas toujours dans les entreprises. Cependant, il semble que l'utilisation de modèles soit répandue dans le monde du management. L'efficacité des métriques est sujet au fait que les modèles représentent la réalité de façon fiable ou pas. Il est néanmoins toujours possible d'utiliser des méthodes de "process mining" ou d'analyse des deltas [VanderAalst04] pour définir des modèles correspondant à ce qui est réellement utilisé.

Les métriques que nous proposons sont structurées au sein d'un cadre, en fonction des concepts qu'elles utilisent [Etien05d]. Ce cadre a été adapté du cadre de Cavano et McCall initialement défini pour des métriques permettant d'évaluer la qualité d'un logiciel.

Afin d'éviter de définir des métriques pour chaque couple de méta-modèles spécifiques, nous proposons de les décrire à un niveau générique, entre deux méta-modèles. Ces deux méta-modèles sont définis à partir de l'ontologie de Bunge Wand et Weber en ce qui concerne le système et l'ontologie de Soffer et Wand pour les processus apportant ainsi une base théorique à la définition des métriques. Un processus de génération permet ensuite, à partir des métriques génériques, de définir les métriques spécifiques. Ce processus est décrit à la section 4.

Le reste de la section est organisée comme suit : la section 3.1 présente le cadre permettant d'organiser la production de métriques ; la section 3.2 introduit les trois niveaux d'abstraction (générique, spécifique et projet) auxquels sont définies les métriques et leur application ; les sections 3.3 et 3.4 définissent le niveau générique en détaillant respectivement les deux méta-modèles et les métriques.

3.1. Critères et métriques d'alignement

A partir des principaux concepts fréquemment utilisés dans les méta-modèles de processus et méta-modèles du système, nous avons défini des critères pour mesurer l'alignement et nous leur avons associé des métriques. Dans ce but, nous avons défini un cadre de mesure en nous appuyant sur celui défini par Cavano et McCall [Cavano88].

3.1.1 Le cadre de mesure de la qualité du logiciel de Cavano et McCall

Cavano et McCall [Cavano88] ont développé un cadre hiérarchique pour mesurer la qualité des logiciels. Ce cadre repose sur trois concepts clé : les *facteurs*, les *critères* et les *métriques*. Les facteurs sont des caractéristiques qui peuvent être appréciées d'un point de vue externe. Ils sont orientés vers la prise de décisions (exhaustivité, flexibilité, adaptabilité). Les critères de qualité sont les caractéristiques du produit. Ils correspondent au point de vue interne, technique (complétude, simplicité, précision). Les métriques permettent de mesurer un critère. Selon le IEEE Standard Glossary of Software Engineering terminology [IEEE SGSET91], la métrique de qualité peut se définir comme « une fonction qui prend pour argument des données du logiciel et qui renvoie une valeur numérique unique. Cette valeur permet de mesurer à quel point le logiciel possède un attribut de qualité donné ».

Le cadre de Cavano et McCall est composé de onze facteurs et d'une trentaine de critères. C'est une référence dans le domaine de la mesure de qualité. Il a inspiré de nombreux autres modèles hiérarchiques comme le modèle de Boehm [Boehm78], le standard ISO / IEC 9126 [ISO/IEC 9126] ou le modèle de Dromey [Dromey95]. Le cadre de McCall est également utilisé par des industriels comme par exemple, aux Etats-Unis dans tous les grands projets militaires, publics ou liés à l'espace [Fitzpatrick96].

3.1.2 Le cadre de mesure de l'alignement

De la même façon, notre cadre de mesure d'alignement est organisé autour des notions de *facteurs*, *critères* et *métriques*. Nous avons défini quatre facteurs selon lesquels la relation d'alignement peut être mesurée. Chaque facteur a des critères qui lui sont associés. Ces derniers sont, à leur tour, associés à des métriques qui permettent de mesurer le degré

d'alignement entre le modèle de processus d'entreprise et le modèle du système qui les supporte. Comme le montre le Tableau 4, dix critères et dix métriques ont été identifiés.

Facteurs	Critères	Metriques
Alignement Intentionnel	Taux de support	Ratio d'activités prises en charge par le système
	Satisfaction des buts	Ratio des buts satisfaits par le système
	Présence des acteurs	Ratio d'acteurs existant dans le système
	Présence des ressources	Ratio de ressources existant dans le système
Alignement Informationnel	Complétude de l'information	Ratio d'objets des P & du S qui se correspondent
	Exactitude de l'information	Ratio d'états des P & du S qui se correspondent
Alignement Fonctionnel	Complétude de l'activité	Ratio d'objets d'une activité correspondant à un objet du système
	Exactitude de l'activité	Ratio d'états d'objets d'une activité correspondant à un état du système
Alignement Dynamique	Fiabilité du système	Ratio de transitions d'état implémentées
	Réalisme dynamique	Ratio de chemins implémentés

Tableau 4 : Métriques pour mesurer la relation d'alignement

Le *facteur intentionnel* et ses critères associés évaluent le degré selon lequel les activités des processus d'entreprise sont supportées par le système (*Taux de support*). Il mesure aussi à quel point le système permet de satisfaire les buts du business (*Satisfaction des buts*) ou de supporter les acteurs et les ressources (*Présence des acteurs* et *Présence des ressources*).

Le *facteur informationnel* complète le facteur précédent en proposant une analyse approfondie de la façon dont l'information gérée au sein du processus est supportée par le système. Pour être parfaitement aligné avec les processus d'entreprise, le système doit manipuler tous les objets définis dans le modèle de processus et supporter tous les états de ces objets. Afin de permettre une telle évaluation deux critères ont été définis, la *Complétude de l'information* et l'*Exactitude de l'information*.

Le *facteur fonctionnel* étudie en détail chaque activité des processus et évalue pour chacune d'entre elles le taux d'objets utilisés par les processus et le taux de leurs états qui sont respectivement supportés par le système. De telles évaluations se font à l'aide de deux critères, la *Complétude de l'activité* et l'*Exactitude de l'activité*.

Le *facteur dynamique* a pour but d'évaluer le caractère dynamique des processus d'entreprise à travers l'étude des transitions d'états et des chemins. Ce facteur a deux critères qui lui sont associés, le critère de *Fiabilité du système* et celui de *Réalisme dynamique*.

3.1.3 Notations

Toutes les métriques sont construites de la même façon :

1. elles utilisent (i) les concepts du méta-modèle de processus et du méta-modèle de système et (ii) les liens *correspond* ou *représente*.

2. ce sont des fonctions prenant en paramètre deux modèles, un modèle de processus d'entreprise et un modèle du système. Les métriques du facteur fonctionnel prennent en paramètre un troisième paramètre, l'activité que l'on étudie et pour laquelle on calcule ces métriques.

3. elles sont définies comme Nombre d'éléments de A sur Nombre d'éléments de B, de façon formelle card(A)/card(B) où :

   - B est un ensemble d'éléments présents dans le modèle de processus ;
   - A est un sous-ensemble de B. A est l'ensemble des éléments de B qui sont supportés par le système, c'est-à-dire tel que, pour chaque élément de B, il existe

un élément du modèle du système qui lui soit lié par un lien *correspond* ou *représente* (suivant la métrique).
- le mot clé card spécifie qu'on calcule le nombre d'éléments de l'ensemble étudié.

Ainsi, le critère *Présence des acteurs* évalue le degré selon lequel les acteurs présents dans les processus d'entreprise sont gérés par le système. Dans la métrique associée :

B est l'ensemble des acteurs présents dans le modèle de processus et
A est l'ensemble des acteurs de B pour lesquels il existe un objet dans le modèle de système qui peut être lié à cet acteur par un lien *correspond*. Cet objet doit se trouver à l'interface entre le système et son environnement et pouvoir déclencher des actions sur d'autres objets.

Par exemple, dans le cas de gestion de réservations de chambres d'hôtel, nous considérons que le client est un acteur. Cet acteur sera pris en charge par le système s'il existe un objet dans le système qui peut être relié à l'acteur par un lien *correspond*. Dans le système, cet objet doit pouvoir déclencher des actions sur d'autres objets. Si le concept de client est considéré dans le système comme une propriété de la réservation, l'acteur client du modèle de processus n'est pas géré par le système.

La métrique associée au critère *Présence des acteurs* s'écrit : $card(Ac_p^m) / card(Ac_p)$.

En effet, de façon pratique, les deux ensembles ne sont pas nommés A et B. Nous avons essayé de leur donner un nom plus explicite. L'ensemble B a un nom qui contient une ou deux lettres correspondant à la (ou les deux) première(s) lettre(s) de l'élément du méta-modèle de processus qu'on étudie, et un p pour **p**rocessus en indice. Afin d'éviter d'avoir deux ensembles d'éléments qui portent le même nom, on utilise parfois la première lettre de l'élément en anglais ou les deux premières lettres de l'élément si les termes anglais et français commencent par la même lettre. Dans notre exemple, l'ensemble des acteurs présents dans le modèle de processus est nommé Ac_p.

L'ensemble A est désigné comme B avec en exposant la lettre r ou m pour montrer (i) que A est un sous-ensemble de B et (ii) que ces éléments sont reliés à des éléments du modèle du système par un lien ***r**eprésente* ou *correspond* (**m**ap en anglais). Les ensembles des éléments du système nécessaires à la définition précise de l'ensemble A utilisent la même convention d'écriture. Ils ont en indice la lettre s pour **s**ystème et non un p. Dans notre exemple, l'ensemble des acteurs du système *correspondant* à un objet de l'environnement qui déclenche une transition d'état sur un autre objet est nommé Ac_p^m.

Afin d'éviter de redéfinir ces métriques pour chaque couple de méta-modèles spécifiques, nous proposons de les spécifier à un niveau générique. La section suivante permet d'identifier le lien entre les différents niveaux ainsi que leurs différents composants.

3.2. Définition des métriques spécifiques

Nous nous proposons d'évaluer le degré d'alignement entre un modèle de processus d'entreprise et un modèle de système. Il est donc nécessaire de définir des métriques spécifiques pour ces deux modèles, c'est-à-dire entre les éléments de leurs méta-modèles.

Cependant, afin d'aider les ingénieurs à définir les métriques spécifiques associées au méta-modèle de processus et au méta-modèle de système, nous définissons des métriques génériques. Celles-ci sont spécifiées entre deux méta-modèles génériques. Ces méta-modèles génériques servent de méta-modèles de référence de la même façon que dans [Rosemann02]. La Figure 16 schématise notre approche de définition de métriques spécifiques.

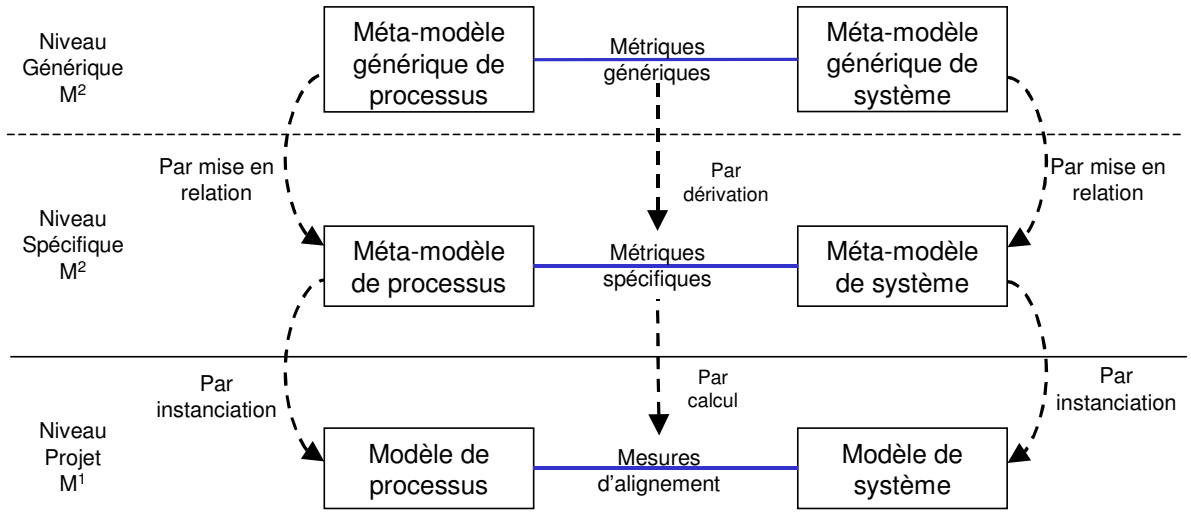


Figure 16 : Un système de métriques d'alignement à trois niveaux différents

Notre approche adopte la terminologie de l'OMG. Les trois niveaux représentés à la Figure 16 correspondent aux deux niveaux intermédiaires de l'architecture en couches [OMG03].

Le *niveau Projet* nommé niveau modèle dans [MOF] ou M^1 dans [Bezivin01] correspond aux méta-données qui permettent de décrire les données du niveau instance (non représenté sur la figure). Ces méta-données sont agrégées sous forme de modèle. A ce niveau, on trouve en particulier un modèle de processus et un modèle de système entre lesquels nous cherchons à évaluer, à un instant donné, l'alignement.

Le *niveau Spécifique* nommé niveau méta-modèle dans [MOF] ou M^2 dans [Bezivin01] contient la description (c'est-à-dire les méta-méta-données) de la structure et de la sémantique des méta-données. Ce niveau correspond au méta-modèle de processus et au méta-modèle de système qui indiquent le type des éléments utilisés dans le modèle de processus et dans le modèle de système. Ces modèles sont des instances de leur méta-modèle respectif. Les méta-modèles i* [Yu97], diagramme d'activités UML [UML], WIDE [Casati96], Entité-Relation [Chen76] … sont à ce niveau. L'évaluation de l'alignement entre les modèles de processus et de système du niveau projet est obtenue en appliquant les métriques spécifiques définies entre le méta-modèle de processus et le méta-modèle du système.

Le *niveau Générique* se situe au même niveau que le niveau spécifique dans l'architecture en couches de l'OMG. Néanmoins, les méta-modèles définis à ce niveau servent de référence pour représenter un système ou un processus d'entreprise. Nous avons défini deux méta-modèles génériques :

1. le méta-modèle BPRAM (**B**usiness **P**rocess **R**epresentation for **A**lignment **M**easure) qui permet de représenter de façon générique les concepts intervenant dans la modélisation des processus d'entreprise. Ce méta-modèle est défini à partir de l'ontologie de Soffer et Wand (SW) [Soffer04c] ;

2. le méta-modèle SRAM (**S**ystem **R**epresentation for **A**lignment **M**easure) qui permet de représenter de façon générique les concepts permettant de modéliser tout système. Ce méta-modèle est défini à partir de l'ontologie de Bunge, Wand et Weber (BWW) [Wand93].

Ces méta-modèles génériques identifient les concepts génériques nécessaires à la définition de métriques génériques pour évaluer l'alignement. Ils permettent, par mise en relation, de rendre explicites les éléments et la structure des méta-modèles spécifiques nécessaires à l'évaluation de l'alignement.

Ces méta-modèles génériques ne sont pas universels (comme le MOF), c'est-à-dire qu'ils ne représentent pas un système ou des processus de façon exhaustive. Au contraire, ces méta-modèles proposent une description partielle pour un objectif particulier, celui d'évaluer l'alignement entre un modèle de processus et un modèle de système. Certains aspects sont pris en compte alors que d'autres sont omis. Ils sont le résultat de l'application d'un filtre [Breton02].

La génération d'un système de métriques d'alignement spécifiques repose sur la mise en relation des concepts des méta-modèles intervenant dans les différentes métriques génériques. [Rosemann02] utilise ce principe de mise en relation entre méta-modèles. Le méta-modèle BWW sert, par exemple, de référence pour comparer deux méta-modèles, celui d'ARIS [Scheer98] et celui du diagramme d'activités UML [UML].

Les concepts du méta-modèle générique de processus sont, par exemple, mis en relation avec ceux de i* [Yu97], si les processus d'entreprise sont spécifiés avec ce méta-modèle. De la même façon, les éléments du méta-modèle spécifique du système (par exemple les diagrammes de classes et de transitions d'états UML [UML]) sont mis en relation avec ceux du méta-modèle générique du système. Les métriques spécifiques d'alignement sont obtenues par dérivation des métriques génériques d'alignement. Les mesures d'alignement sont calculées par application des métriques spécifiques.

Nous pensons qu'il existe un grand nombre d'avantages à utiliser une approche en niveaux d'abstraction et à définir les méta-modèles au niveau générique à partir d'adaptations de l'ontologie de Bunge :

1. les métriques génériques reposent sur une base théorique fournie par l'ontologie de Bunge ;
2. les métriques génériques servent de guide pour définir les métriques spécifiques : ces dernières étant une spécialisation des premières ;
3. le processus pour construire les métriques spécifiques est simple et facile à mettre en œuvre. Il génère moins d'erreurs que des constructions ad hoc et, enfin,
4. les ensembles spécifiques de métriques d'alignement sont cohérents entre eux car ils sont générés à partir du même moule.

Les deux sous-sections suivantes présentent le niveau générique en décrivant respectivement les deux méta-modèles et les métriques génériques. La section 4 détaille et illustre le processus de génération de métriques spécifiques.

3.3. Présentation des deux méta-modèles génériques

Cette section décrit le méta-modèle générique du système et le méta-modèle générique d'entreprise qui sont respectivement définis à partir de l'ontologie de Bunge, Wand et Weber [Weber92] et de l'ontologie de Soffer et Wand [Soffer04c]. Nous commençons par introduire le concept d'ontologie.

3.3.1 La notion d'ontologie

Dans cette section, nous nous intéressons à la notion d'ontologie, ainsi qu'à son rôle en ingénierie des systèmes d'information.

3.3.1.1 Qu'est-ce qu'une ontologie ?

Le terme ontologie est issu du domaine de la philosophie. Il signifie selon le Petit Robert [Robert00] : « partie de la métaphysique qui s'applique à l'être en tant qu'être, indépendamment de ses déterminations particulières ». L'ontologie est la branche de la philosophie qui s'intéresse (i) à la structure de l'être, cherchant à en définir l'essence, et (ii) à son comportement, c'est-à-dire aux modes d'existence de cet être.

A partir de cette notion, on a défini le concept dénombrable d'ontologie. On ne parle alors plus de l'ontologie, mais d'une ou plusieurs ontologies.

En informatique, il n'existe pas une définition unique du terme ontologie. Nous adoptons la définition relativement simple de Schulze-Kremer [SchulzeKremer97] : une ontologie est une « description concise et non ambiguë des principales entités d'un domaine d'application et des potentielles relations qui peuvent exister entre elles ».

Des ontologies peuvent être proposées à différents niveaux d'abstraction [Wand04]. Au niveau le plus général, une ontologie rassemble les concepts fondamentaux nécessaires à la description de n'importe quel phénomène du monde. Au niveau intermédiaire, une ontologie manipule les concepts nécessaires pour décrire des types particuliers de phénomènes qui existent dans certain domaine, comme l'architecture, le droit… Aux niveaux les plus bas, les ontologies manipulent les concepts nécessaires à des mondes spécifiques comme par exemple, le monde perçu par une entreprise dans un contexte particulier. C'est à ces deux derniers niveaux que les ontologies utilisées en informatique sont associées.

Les théories de l'ontologie sont pertinentes pour le domaine des systèmes d'information, car, selon Wand et Weber, l'essence d'un système d'information est de fournir une représentation fidèle d'un monde qu'une personne ou un groupe de personnes perçoit. Le système d'information sera alors aussi proche de la réalité que l'ontologie elle-même. Une ontologie fournit un ensemble de concepts pour modéliser des systèmes et raisonner sur leurs caractéristiques.

L'utilisation d'une ontologie permet (i) d'apporter une base théorique et (ii) d'être suffisamment général et indépendant des technologies ou méta-modèles utilisés.

3.3.1.2 Des ontologies dans le domaine des systèmes d'information

Dans le domaine des systèmes d'information, les ontologies permettent d'identifier les éléments de base du monde réel que les systèmes d'information doivent être capables de modéliser. Cependant, en partant du principe que les systèmes d'information sont des éléments du monde réel, Wand et Weber utilisent des ontologies pour modéliser les systèmes d'information eux-mêmes et vérifier si les méta-modèles permettant leur conception sont complets (c'est-à-dire fournissent tous les concepts nécessaires à la modélisation de système d'information).

Pour ces raisons, Wand et Weber ont adapté l'ontologie de Bunge [Bunge77], [Bunge79] qui repose sur le concept de *chose* [Wand92], [Wand93]. Ce fut la première adaptation de l'ontologie de Bunge qui a servi de base à d'autres ontologies comme celle définie par Soffer et Wand [Soffer04c]. Ces deux auteurs ont défini une ontologie des processus d'entreprise comme fondement théorique pour la conception et la modélisation des processus d'entreprise.

3.3.1.3 Utilisation de méta-modèles

Nous avons choisi de ne pas utiliser directement l'ontologie de Bunge Wand et Weber (BWW) et l'ontologie de Soffer et Wand (SW) pour deux raisons :

1. les concepts de ces deux ontologies sont définis au niveau instance. Le concept élémentaire de l'ontologie de Bunge comme des ontologies BWW et SW est celui de *chose* qui correspond à un élément du monde réel. Un exemple de chose serait le client C632.

2. les concepts de ces deux ontologies sont définis en utilisant la théorie des ensembles. Dans de nombreux travaux de recherche, des chercheurs ont essayé de simplifier et de clarifier les explications de ces concepts en les définissant en langage naturel. Cependant, les concepts restent malgré tout difficiles à comprendre et nécessitent une connaissance en profondeur de ces ontologies.

Il nous semble pertinent de décrire les métriques au niveau type et non au niveau instance. En effet, pour évaluer l'alignement, il nous semble plus intéressant de vérifier que le concept de client présent dans le modèle de processus est aussi présent dans le modèle de système plutôt que de constater que le client C632 existe dans les deux bases de données. L'utilisation des types est répandue et est devenue une norme en particulier avec les travaux de l'OMG.

Rosemann et Green [Rosemann02] ont proposé une définition formelle des concepts clé de l'ontologie BWW en les décrivant par un méta-modèle. De cette façon, la définition est plus familière aux professionnels des SI, plus spécifique que le langage naturel mais plus simple à comprendre qu'un langage reposant sur la théorie des ensembles. L'utilisation d'un méta-modèle permet d'expliquer clairement les concepts et leurs liens. Un tel méta-modèle permet également de mieux structurer et analyser les ontologies. Les deux méta-modèles génériques que nous définissons s'inspirent des travaux développés par l'équipe de Rosemann dans le cadre du projet ARC [ARC].

Nous ne reprenons pas le méta-modèle BWW développé dans ce projet car celui-ci est trop complexe pour l'usage que nous voulons en faire. Nous construisons deux méta-modèles à partir des ontologies BWW et SW. Ces deux méta-modèles ne cherchent pas à modéliser chacune de ces deux ontologies comme le fait [Rosemann02]. Ils ont pour but de mettre en lumière les concepts qui sont importants afin de nous permettre de mesurer l'alignement entre

le modèle d'entreprise et le modèle du système. Ils n'utilisent donc que la partie des concepts de ces deux ontologies qui nous paraissent pertinents pour ce propos.

L'utilisation des ontologies comme base de ces méta-modèles apporte (i) une certaine légitimité dans la mesure où elles permettent de vérifier qu'un méta-modèle est complet et (ii) un fondement théorique.

3.3.2 Le méta-modèle SRAM et le méta-modèle BPRAM

Nous définissons dans cette section deux méta-modèles :

- le méta-modèle BPRAM (**B**usiness **P**rocess **R**epresentation for **A**lignment **M**easure) qui permet de représenter de façon générique différentes sortes de processus d'entreprise ;
- le méta-modèle SRAM (**S**ystem **R**epresentation for **A**lignment **M**easure) qui permet de représenter de façon générique des systèmes d'information.

Pour faciliter la lecture, on utilise, dans la suite de cette thèse, les expressions 'éléments BPRAM' et 'éléments SRAM' pour parler des éléments de ces deux méta-modèles.

Le concept de base de ces deux méta-modèles reste celui de *chose*. Cependant, nous le définissons au niveau type. Nous considérons donc qu'une chose est un ensemble d'éléments ayant des propriétés communes. Le concept de client est un exemple de chose. Une solution alternative aurait été de conserver 'chose' au niveau instance, ce qui suppose :

1. qu'une classe de choses est un ensemble de choses possédant des propriétés communes
2. qu'une classe de choses est une chose.

Ce procédé est décrit à la Figure 17. Il aurait dû être utilisé pour chacun des concepts de l'ontologie. Mais, nous avons jugé que ce procédé aurait inutilement alourdi les deux méta-modèles.

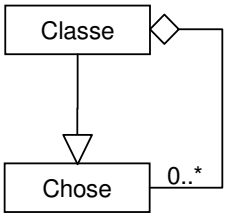


Figure 17 : Représentation des concepts de classe et de chose

Nous avons donc préféré définir chacun des concepts à un niveau type.

3.3.2.1 Méta-modèle SRAM

Le méta-modèle SRAM permet de mettre en évidence les différents éléments nécessaires à la représentation d'un système en vue de mesurer l'alignement entre une instance de ce modèle du système et une instance du modèle de processus.

Nous définissons le méta-modèle SRAM à partir des concepts de l'ontologie de Bunge, Wand et Weber [Wand92], [Wand93]. Ce méta-modèle permet : (1) une représentation statique du système grâce en particulier aux concepts de chose et de propriété et (2) une représentation

dynamique au travers des concepts d'état, d'événement et de transformation qui permettent de préciser comment les choses évoluent au cours du temps. La Figure 18 présente le méta-modèle SRAM en utilisant la notation UML.

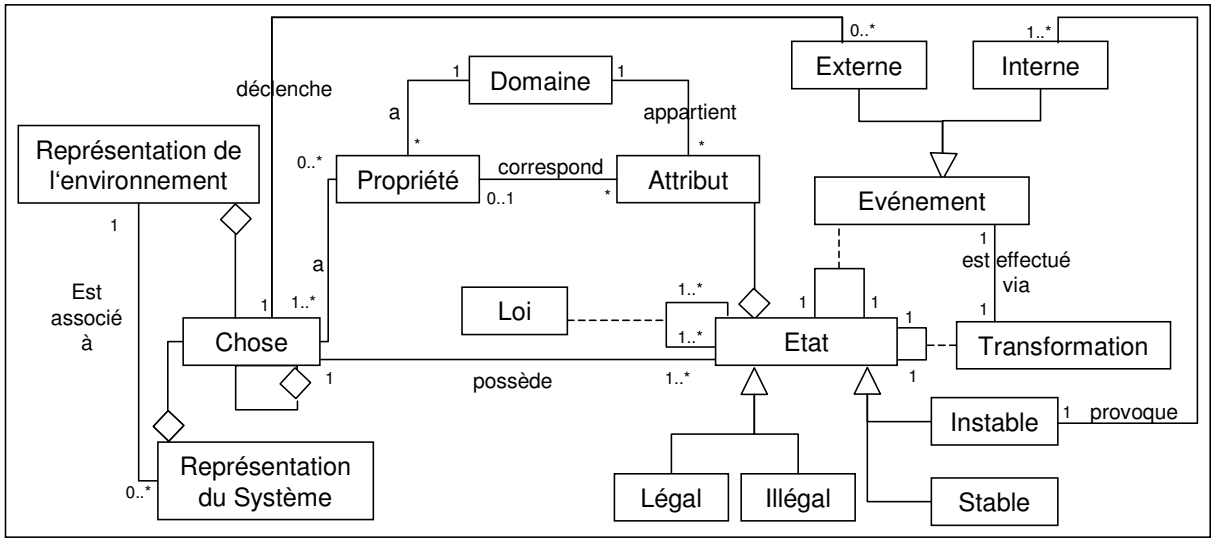


Figure 18 : Méta-modèle SRAM représenté avec la notation UML

Une *chose* est un concept qui permet de définir, à un niveau type, les propriétés ou le comportement d'un ensemble d'éléments de même nature. Une chose peut être simple ou composite si elle est composée d'autres choses. Les choses possèdent des *propriétés* qui peuvent être (i) intrinsèques si elles sont propres à une chose, comme par exemple la hauteur, (ii) mutuelles si elles appartiennent à plusieurs choses, comme une personne qui travaille dans une entreprise, ou (iii) émergentes si elles appartiennent à une chose composée et non à l'un des composants. Les propriétés sont modélisées par une fonction *d'attribut* qui permet de faire correspondre une valeur à une chose.

A un instant *t*, l'ensemble des valeurs des attributs de la chose définit son *état*. Une *loi d'état* permet de restreindre les valeurs des propriétés d'une chose à un sous-ensemble qui est considéré comme possible d'après des règles naturelles ou humaines. La loi qui, par exemple spécifie que l'âge d'une personne n'excède pas 150 ans est une loi naturelle, c'est un fait observable dans la nature. Ainsi, si la valeur de la propriété âge de la chose personne est égale à 152, cette valeur ne satisfait pas la loi. La chose est dans un état illégal. De la même façon, il existe des lois humaines, c'est-à-dire définies par les hommes. Par exemple, la réduction appliquée sur les achats d'une personne ne peut excéder le taux de remise fixé en fonction de son statut ou de celui de l'organisation à laquelle elle appartient. Ainsi, la réduction accordée à Monsieur X membre de l'entreprise Y ne peut dépasser 40% car une loi spécifie que le rabais attribué aux employés de l'entreprise Y ne peut excéder le taux de 40%. Au delà, la chose taux de remise est dans un état illégal.

Un état d'une chose est considéré comme *stable* si (i) il peut durer dans le temps et (ii) a besoin de l'action d'une autre chose pour changer. Un état *instable* est un état qui doit évoluer. Un changement d'état est appelé un *événement* ; il est effectué via une *transformation*. Plus précisément, un événement est une paire d'états <s, s'> où s et s' sont deux états tels qu'il existe une transformation g telle que s' = g(s). L'état d'une chose peut entraîner un changement d'état d'une autre chose. On dit que deux choses sont couplées si

l'histoire (c'est-à-dire l'ensemble des états qui varient en fonction du temps) d'une des choses dépend de l'histoire de l'autre.

De cette façon, il est possible de définir le concept de *représentation du système* comme un ensemble de choses couplées à au moins une autre chose de l'ensemble. La *représentation de l'environnement* correspond aux choses qui sont extérieures à la représentation du système, mais qui sont couplées avec au moins une des choses de cette représentation. Un *événement externe* est un événement qui est déclenché par une chose sur une autre chose et donc par extension, par une chose de la représentation de l'environnement sur une chose de la représentation du système. Par opposition, un *événement interne* correspond à un événement qui survient sur une chose, un sous-système ou le système en vertu d'une transformation au sein même de la chose du sous-système ou du système.

Dans un système de gestion de commande, les concepts de client, compte client, commande, matières premières et fournisseur sont des choses. Les concepts de client et de fournisseur sont des éléments de la représentation de l'environnement alors que toutes les autres choses font partie de la représentation du système. Lorsqu'un client fait une commande, cela correspond à un événement externe puisqu'une chose de la représentation de l'environnement, le client, déclenche un événement sur la chose commande de la représentation du système. En effet, un changement d'état correspondant à la création de la commande survient sur cette chose. Le système se trouve alors dans un état instable. Un ou plusieurs événements internes doivent avoir lieu afin de restaurer la stabilité du système. Ainsi, en conséquence de la création de commande, le nombre de matières premières doit diminuer et le compte client est mis à jour. Des transformations permettent de réaliser ces événements.

3.3.2.2 Méta-modèle BPRAM

Le méta-modèle BPRAM permet de représenter les processus au niveau générique. Il est défini à partir de l'ontologie de Soffer et Wand [Soffer04c] qui est une adaptation de l'ontologie de Bunge, Wand et Weber et utilise, en partie, les mêmes concepts que cette ontologie. Le méta-modèle BPRAM a un grand nombre de concepts identiques au méta-modèle BPRAM, comme les choses, les propriétés, les états, les lois, les transformations ou les événements. Certains concepts ont néanmoins été ajoutés ou précisés afin de pouvoir représenter les spécificités propres aux processus d'entreprise. La Figure 19 présente le méta-modèle BPRAM en utilisant la notation UML.

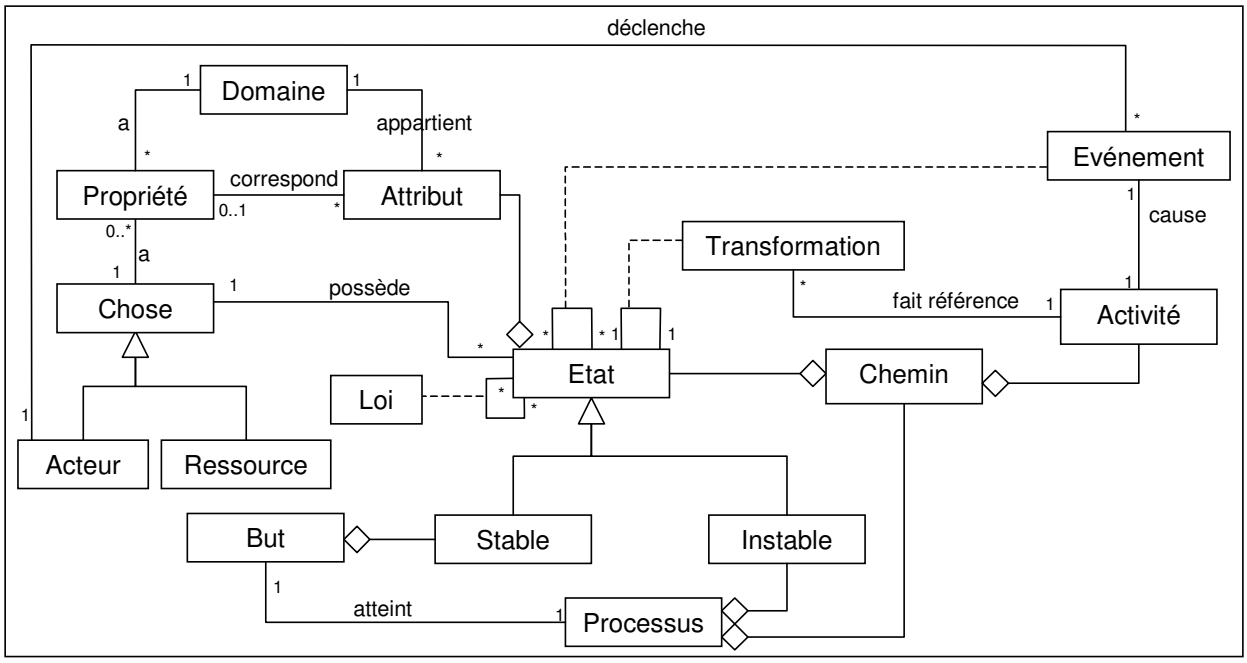


Figure 19 : Méta-modèle BPRAM représenté avec la notation UML

Dans le méta-modèle BPRAM, le concept de chose est défini au niveau type de la même façon que dans le méta-modèle SRAM. Une chose a des propriétés qui prennent leurs valeurs dans un domaine et possède des états. Les concepts de client ou de commande sont des exemples de choses.

En plus des concepts identiques au méta-modèle SRAM tels que les choses, les états, les propriétés, les transformations, etc., le méta-modèle BPRAM permet de représenter les spécificités des processus d'entreprise. Ainsi, un *chemin* est un ensemble d'états tel qu'il existe des transformations permettant de passer d'un état à l'état suivant. Un *processus* est un ensemble de chemins, mais c'est aussi une séquence d'états instables aboutissant à un ensemble d'états stables appelés aussi *but*. Les *activités* sont des transitions d'état. Elles sont causées par des *événements* et associées à des *transformations*. Les *acteurs* sont des choses à l'interface des processus d'entreprise et de leur environnement qui déclenchent des événements externes. Les *ressources* sont des choses qui ne provoquent aucun changement d'état. La partition entre acteur et ressource n'est pas totale ; certaines choses ne sont ni des acteurs ni des ressources.

Si on s'intéresse, par exemple, au processus relatif à la gestion de commande, les concepts de client, compte client, commande, matières premières et fournisseur peuvent être considérés comme des choses. Les concepts de client et de fournisseur sont des acteurs. En effet, le client déclenche un événement sur la chose commande lorsqu'il fait une commande. Le fournisseur déclenche un événement sur la chose matières premières quand il fait une livraison. Si on s'intéresse au processus de gestion de commande de la demande par le client jusqu'à la livraison de la marchandise chez le client, le but du processus est que la commande soit livrée et que le compte client soit débité. En termes d'états, le but peut s'écrire comme suit : (commande statut = livrée) AND (compte client = débité). On peut avoir différentes façons d'atteindre ce but suivant que la marchandise est en stock ou pas. Différents chemins peuvent donc être définis. L'un de ces chemins peut, par exemple, être composé des activités suivantes : création de la commande, préparation de la marchandise, prévenir le client que la

marchandise est disponible, paiement de la commande, livraison au client. Chacune de ces activités entraîne des changements d'état sur les différentes choses identifiées. Ces changements sont effectués par des transformations.

Ces deux méta-modèles permettent d'identifier les concepts nécessaires, à notre sens, pour définir des métriques d'alignement.

3.4. Définition des métriques génériques

Dans cette section, nous présentons les dix métriques génériques en utilisant : (i) les concepts des méta-modèles SRAM et BPRAM pour représenter le système et le processus et (ii) les liens *représente* et *correspond*.

Le Tableau 5 présente de façon synthétique les dix métriques génériques en utilisant le cadre de mesure d'alignement. Les concepts des méta-modèles SRAM et BPRAM sont soulignés, le lien type est écrit en italique. Ainsi, par exemple la métrique associée au critère *Taux de support* peut s'écrire au niveau générique comme :

Nombre d'activités *représentées* chacune par un événement du système / Nombre d'activités.

Activité est un concept du méta-modèle BPRAM, événement un élément du méta-modèle SRAM et *représentée* signifie qu'il existe un lien *représente* entre ces deux concepts.

Facteurs	Critère	Métriques	Description
Alignement Intentionnel	Taux de Support	Taux d'activités supportées	Nombre d'activités *représentées* chacune par un événement du système / Nombre *d'activités*
	Satisfaction des buts	Taux des buts satisfaits par le système	Nombre de buts pour lesquels chaque état *correspond* à un état du système / Nombre de buts
	Présence des Acteurs	Taux d'acteurs existant dans le système	Nombre d'acteurs des processus *correspondant* chacun à une chose du système déclenchant un événement / Nombre d'acteurs des processus
	Présence des ressources	Taux de ressources existant dans le système	Nombre de ressources *correspondant* chacune à une chose du système / Nombre de ressources
Alignement Informationnel	Complétude de l'information	Taux de choses des P & du S qui se correspondent	Nombre de choses des processus *correspondant* chacune à une chose du système / Nombre de choses des processus
	Exactitude de l'information	Taux d'états des P & du S qui se correspondent	Nombre d'états de choses des processus *correspondant* chacun à un état d'une chose du système / Nombre d'états de choses des processus
Alignement Fonctionnel	Complétude de l'activité	Taux de choses des P & du S qui se correspondent	Nombre de choses d'une activité *correspondant* chacune à une chose du système / Nombre de choses des processus
	Exactitude de l'activité	Taux d'états des P & du S qui se correspondent	Nombre d'états de choses d'une activité *correspondant* chacun à un état d'une chose du système / Nombre d'états de choses des processus
Alignement Dynamique	Fiabilité du système	Taux de lois implémentées	Nombre de transformations du modèle de processus qui *correspondent* chacune à une transformation du modèle de système / Nombre de transformations du modèle de processus
	Réalisme dynamique	Taux de chemins	Nombre de chemins pour lesquels chaque état intervenant dans le chemin *correspond* à un état d'une chose du système tels que la succession de ces états dans le système soit possible / Nombre de chemins

Tableau 5 : Définition des métriques au niveau générique

Les définitions formelles de ces métriques font intervenir un certain nombre d'ensembles que nous précisons dans le Tableau 6. Deux fonctions sont utilisées :

- type() : qui s'applique à un élément et permet d'en préciser le type ;

- a() : qui s'aplique à un élément et permet de préciser que l'élément passé en paramètre est possèdé par le premier élément. Ainsi c.a(e) signifie que c possède e.

Définition	Commentaire
$A_p = \{a \in M_{proc} \mid a.type(t) = Activité\}$	ensemble des activités du modèle de processus M_{proc}
$Ev_s = \{ev \in M_{syst} \mid ev.type(t) = Evénement\}$	ensemble des événements du modèle de système M_{syst}
$B_p = \{b \in M_{proc} \mid b.type(t) = But\}$	ensemble des buts présents dans le modèle de processus M_{proc}
$C_p = \{c \in M_{proc} \mid c.type(t) = Chose\}$	ensemble des choses du modèle de processus d'entreprise M_{proc}
$C_s = \{c \in M_{syst} \mid c.type(t) = Chose\}$	ensemble des choses du modèle de système M_{syst}
$E_p = \{e \in M_{proc} \mid e.type(t) = Etat\}$	ensemble des états des choses du modèle de processus M_{proc}
$E_s = \{e \in M_{syst} \mid e.type(t) = Etat\}$	ensemble des états des choses du modèle de système M_{syst}
$E_{s,c} = \{e \in M_{syst} \mid e.type(t) = Etat \wedge c.a(e) \wedge c.type(t) = Chose\}$	ensemble des états de la chose c du modèle de système M_{syst}. $E_{s,c}$ est un sous-ensemble de E_s

Tableau 6 : Présentation des ensembles intervenant dans la définition des métriques génériques

Dans la suite de cette section, nous présentons chacune des dix métriques en détail.

3.4.1 Critères et métriques liés au facteur intentionnel

L'objectif du *facteur intentionnel* est de mesurer le degré de satisfaction des processus d'entreprise par le système. Nous avons identifié quatre critères que nous décrivons ainsi que les métriques qui leur sont associées : le *Taux de support*, la *Satisfaction des buts*, la *Présence des acteurs* et la *Présence des ressources*.

3.4.1.1 Taux de support

Nous construisons le *Taux de support* en nous inspirant du 'Technological coverage' de Bodhuin [Bodhuin04]. Ce critère mesure le pourcentage des activités d'entreprise gérées par le système. De la même façon, le *Taux de support* permet d'exprimer le degré selon lequel les activités des processus sont implémentées dans le système.

Les rares approches de littérature qui s'intéressent à l'évaluation de l'alignement se focalisent sur la façon dont les activités sont gérées par le système. Intuitivement, c'est la première façon d'aborder l'alignement : il est donc important de savoir quelle est la proportion d'activités gérées par le système.

Une activité de processus est prise en charge par le système si elle est *représentée* par un événement du système.

Comme nous l'avons vu, le concept d'activité est spécifique aux modèles de processus. Dans le méta-modèle BPRAM, une activité est une transition d'état causée par un événement et associée à une transformation. Ce concept n'a pas d'équivalent dans le méta-modèle SRAM.

Dans le méta-modèle SRAM, le concept d'événement est la paire d'états <s, s'> où s et s' sont deux états tels qu'il existe une transformation g pour laquelle s' = g(s).

Le concept d'activité du méta-modèle BPRAM et le concept d'événement du méta-modèle SRAM sont très proches, ils sont tous les deux matérialisés par une transformation. Ainsi, on peut considérer qu'un événement affecte l'existence d'une activité.

Par exemple, dans le contexte de gestion de réservations de chambres d'hôtel, l'événement 'paiement de la réservation' change l'état de la chose réservation et affecte l'activité de paiement.

La Figure 20 montre les parties des méta-modèles BPRAM (en haut) et SRAM (en bas) intervenant dans la définition de la métrique *Taux de support*. La flèche épaisse en pointillé permet de mettre en évidence les liens *correspond* ou *représente*.

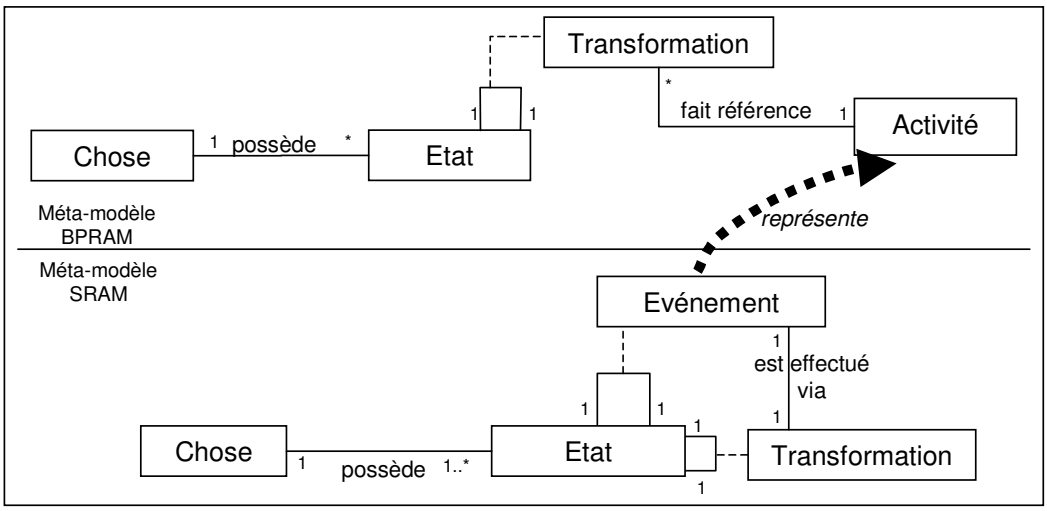


Figure 20 : Parties des méta-modèles BPRAM et SRAM nécessaires à la définition du Taux de support

Ayant compris le lien entre activité et événement, nous pouvons définir la métrique associée au *Taux de support*. Rappelons qu'une métrique est une fonction qui prend comme paramètre un modèle de processus et un modèle de système. La métrique Ts associée au *Taux de support* calcule la proportion d'activités du modèle de processus qui sont reliées par un lien *représente* à un événement du modèle de système.

De façon plus formelle, soit A_p^r l'ensemble des activités de M_{proc} représentées par un événement de M_{syst}. A_p^r est un sous-ensemble de A_p, qui ne comprend que les activités *a* de A_p, pour lesquelles il existe un événement *ev* du modèle du système relié à *a* par un lien *représente*. $A_p^r = \{a, a \in A_p \mid \exists ev \in Ev_s \wedge ev \Re a\}$

La métrique associée à ce critère est :

$$Ts\ (M_{proc}, M_{syst}) = card(A_p^r) / card\ (A_p)$$

L'interprétation de cette métrique est la suivante : plus le *Taux de support* est élevé, plus les activités sont gérées par le SI. A l'inverse, un faible *Taux de support* indique que de nombreuses activités sont gérées manuellement. Afin d'augmenter ce taux, il est nécessaire d'automatiser certaines activités et donc d'introduire dans le système de nouvelles choses, de nouvelles propriétés, de nouvelles lois ou de nouveaux événements pour supporter ces activités.

3.4.1.2 Satisfaction des buts

Le critère de *Satisfaction des buts* et la métrique qui lui est associée visent à vérifier que le système met en œuvre des fonctionnalités qui permettent de satisfaire les buts de l'entreprise. Ce critère mesure donc la proportion des buts d'entreprise qui sont supportés par le système.

Dans le méta-modèle BPRAM, un but est un ensemble d'états stables. Il est légitime de penser qu'un but est supporté par le système si chacun des états qui le définissent existe dans le système, c'est-à-dire qu'il existe un lien *correspond* entre chacun des états du but et un état d'une chose du système (Figure 21).

Ainsi, par exemple, si dans un processus de réservations de chambres d'hôtel, l'un des buts correspond au paiement de la réservation, alors l'état 'payé' de la chose 'réservation' ainsi que l'état 'réservée' de la chose chambre doivent exister dans le modèle du système pour que ce but soit supporté. Dans le cas contraire, ce but ne peut être atteint.

Pour évaluer le critère de satisfaction des buts, nous comparons le nombre de buts du modèle de processus M_{proc} pour lesquels chaque état *correspond* à un état d'une chose du modèle du système M_{syst} et le nombre total de buts dans le modèle d'entreprise. On dit qu'un état e_1 d'une chose c_1 du modèle M_{proc} *correspond* à un état e_2 d'une chose c_2 du modèle M_{syst} si : (i) ces deux choses c_1 et c_2 se *correspondent*, c'est-à-dire que l'ensemble des propriétés de c_1 est isomorphe à l'ensemble des propriétés de c_2 et (ii) l'ensemble des valeurs associées aux propriétés de c_1 est identique à l'ensemble des valeurs associées aux propriétés de c_2.

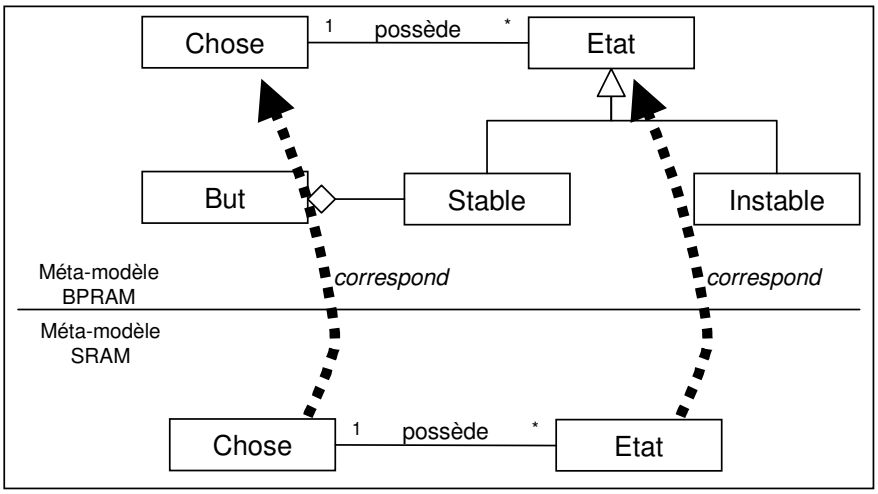


Figure 21 : Parties des méta-modèles BPRAM et SRAM nécessaires à la définition de la Satisfaction des buts

Considérons B_p, l'ensemble des buts présents dans le modèle de processus d'entreprise. L'ensemble B_p^m des buts supportés par le système est un sous ensemble de B_p où l'on ne s'intéresse qu'aux buts tels que quelque soit l'état d'une chose du modèle de processus qui compose ce but, il existe un état d'une chose dans le système qui lui *correspond*. Ceci s'écrit de la façon suivante : $B_p^m = \{b \in B_p \mid \forall\, e, e \in E_p \wedge e \in b \Rightarrow \exists\, e' \in E_s \wedge e \mathcal{M} e'\}$

La métrique Satisfaction des buts s'écrit comme suit :

$$Sb(M_{proc}, M_{syst}) = card(B_p^m) / card(B_p)$$

On peut interpréter la valeur de cette métrique ainsi : un faible taux de *Satisfaction des buts* implique qu'un grand nombre de buts ne peuvent par être atteints par le système. Il est alors nécessaire (i) d'étudier pour chaque but, les états des choses qui le composent et (ii) de s'assurer que ceux-ci sont bien implémentés dans le système. Selon les cas, il peut s'avérer nécessaire d'ajouter des états ou des choses dans le système ou encore supprimer, dans le modèle de processus, certains états qui ne sont pas indispensables à leur réalisation.

Une application de gestion de commandes peut avoir pour but :

$$b = (\text{produit.état} = \text{'reçu'}) \wedge (\text{facture.état} = \text{'envoyée'}) \wedge (\text{commande.état} = \text{'payée'})$$

Ceci signifie que le produit est reçu par le client, que la facture est envoyée et la commande payée. Ces trois états doivent exister dans le système pour que ce but soit considéré comme pris en charge par le système. Or l'état produit.état = 'reçu'de la chose 'produit' n'existe pas dans le système. L'état final pour le produit est produit.état = 'envoyé'. Le but n'est donc pas géré par le système.

Cette distinction entre le système et les processus d'entreprise trouve son origine dans le fait que la concurrence entre les différents sites d'achat en ligne est grande. L'entreprise veut se prévaloir d'un service complet jusqu'à réception de la marchandise. Pour que ce but soit géré par le système, soit l'état produit.état = 'reçu' est introduit dans le système (avec toutes les transformations nécessaires pour atteindre cet état), soit cet état est supprimé des processus d'entreprise dans la mesure où le produit est envoyé par colis suivi du type Colissimo et l'entreprise n'a en fait pas une grande marge de manœuvre sur la réalisation de l'état produit.état = 'reçu'.

3.4.1.3 Présence des acteurs

Les processus d'entreprise mettent en jeu des acteurs qui déclenchent des actions sur d'autres choses. Mesurer la proportion d'acteurs présents dans les processus d'entreprise qui sont modélisés dans le système permet de vérifier que dans le système et dans les processus, les changements d'états sont déclenchés par les mêmes actions.

Selon le méta-modèle BPRAM, un acteur est une chose à l'interface des processus et de leur environnement qui, par une action, déclenche une transition d'états d'une autre chose. La Figure 22 présente les éléments des méta-modèles BPRAM et SRAM nécessaires à la définition de la métrique *Présence des acteurs*.

Dans le méta-modèle BPRAM, acteur du premier degré (c'est-à-dire ayant une interaction directe avec le système) et acteur du second degré (ayant une interaction indirecte) ne sont pas différenciés. Ces deux types d'acteurs étant partie prenante de l'interface, nous pensons qu'il est important qu'ils soient tous deux définis dans la représentation de l'environnement du système sinon les mêmes événements ne sont pas déclenchés par les mêmes actions, selon qu'on considère le processus ou le système.

Par exemple, un client qui passe une commande, dans le cas d'une application de e-commerce sur Internet, est un acteur. En effet, le fait de passer une commande provoque la création d'une commande et donc un changement d'état de la chose commande.

On considère qu'un acteur du modèle de processus M_{proc} est supporté par le système s'il existe une chose de la représentation de l'environnement qui est liée à cet acteur par un lien *correspond* et qui déclenche un événement sur une autre chose dans le modèle du système.

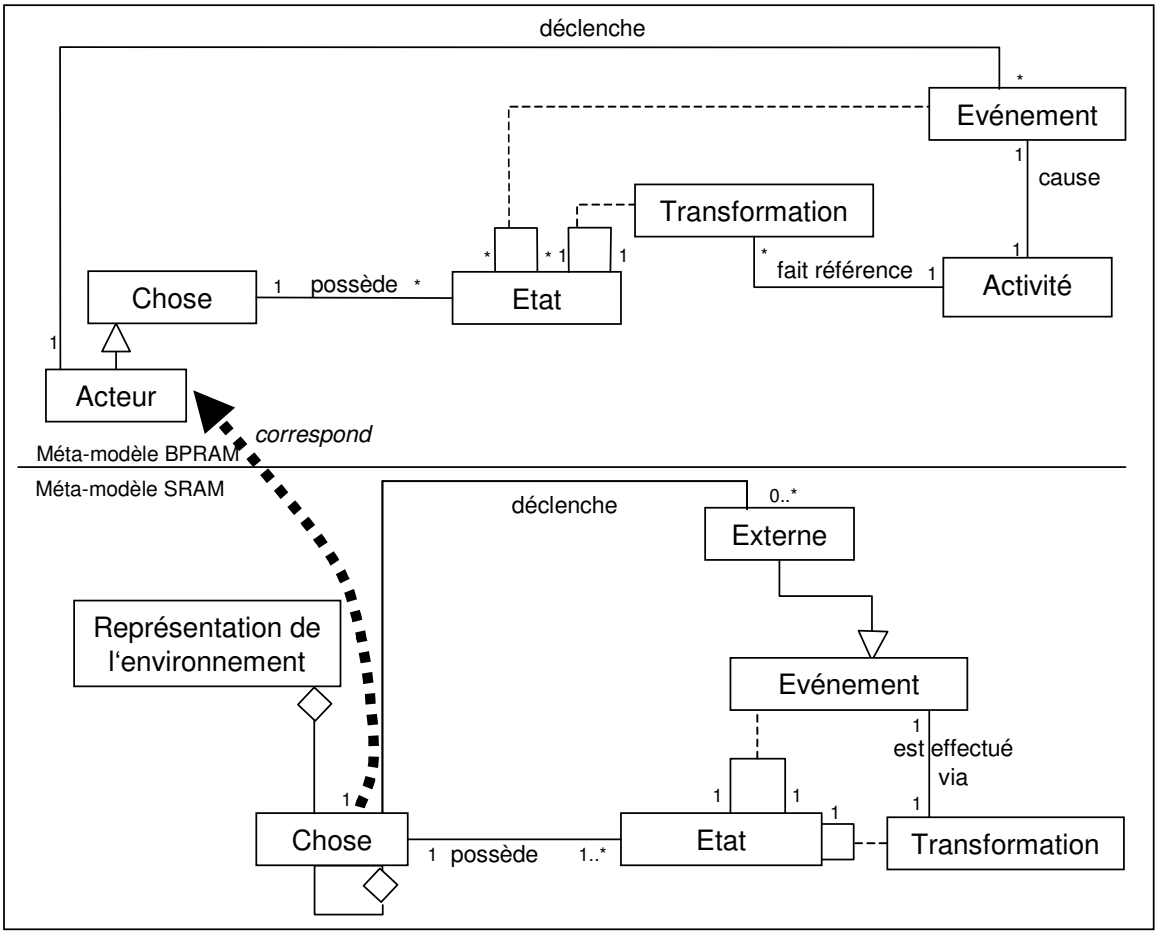


Figure 22 : Parties des méta-modèles BPRAM et SRAM nécessaires à la définition de la Présence des acteurs

Le critère *Présence des acteurs* mesure le taux d'acteurs présents dans les modèles de processus qui existent dans le système.

De façon plus formelle, si on note Ac_p l'ensemble des acteurs présents dans le modèle de processus M_{proc} et Ac_p^m l'ensemble des acteurs de M_{proc} supportés par le système. Ce dernier contient chaque éléments *ac*, pour lequel il existe une chose *c* de l'environnement du système qui est liée par un lien *correspond* à cet acteur et qui déclenche (δ) un événement sur une chose c' de M_{syst}. La métrique associée au critère *présence des acteurs* s'écrit comme le quotient de la cardinalité de Ac_p^m sur la cardinalité de Ac_p.

Soit :

- $Ac_p = \{ac \in M_{proc} \mid ac.type(t) = Acteur\}$
- $Ce_s = \{c \in Re_s \mid c.type(t) = Chose\}$ où Re_s est la représentation de l'environnement du système
- $Ac_p^m = \{ac, ac \in Ac_p \mid \exists\, c \in Ce_s \wedge \exists\, c' \in C_s \wedge \exists\, (e, e') \in E_{s,c'}^2 \wedge c\ \delta <e, e'> \wedge ac\ \mathcal{M}\ c\}$

$$Pa(M_{proc}, M_{syst}) = card(Ac_p^{m}) / card(Ac_p)$$

Une faible valeur de cette métrique signifie qu'un grand nombre d'acteurs présents dans le modèle de processus ne sont pas supportés par le système. En conséquence, certaines transitions d'état ne résultent pas de la même cause dans les processus et dans le système. On peut alors soit (i) supprimer des acteurs des processus d'entreprise qui ne sont pas présents dans le système, s'ils ne jouent pas un rôle significatif ou (ii) introduire de nouveaux acteurs dans le système.

3.4.1.4 Présence des ressources.

Dans la manipulation des processus d'entreprise, on utilise couramment des *ressources*, c'est-à-dire des objets informationnels dont les états ne sont pas modifiés par le processus, mais qui sont indispensables au bon déroulement du processus.

Le critère *Présence des ressources* évalue la proportion de ressources utilisées par les processus d'entreprise qui sont supportées par le système.

Dans le méta-modèle BPRAM, une ressource est une chose qui ne provoque pas d'action.

Une ressource du modèle de processus M_{proc} est gérée par le système si elle *correspond* à une chose du modèle du système M_{syst} qui ne provoque pas de changement d'état sur une autre chose (Figure 23). Par exemple, dans le cas d'un projet d'implémentation de distributeurs automatiques de billets, une carte de crédit est une ressource : elle ne change pas de numéro, ni de titulaire, mais elle est indispensable pour permettre à son propriétaire de retirer de l'argent dans un distributeur. Afin de mémoriser certaines propriétés de la carte tel que le numéro de carte de crédit et la date de validité, il est nécessaire d'avoir dans le système une chose qui correspond à la ressource carte de crédit.

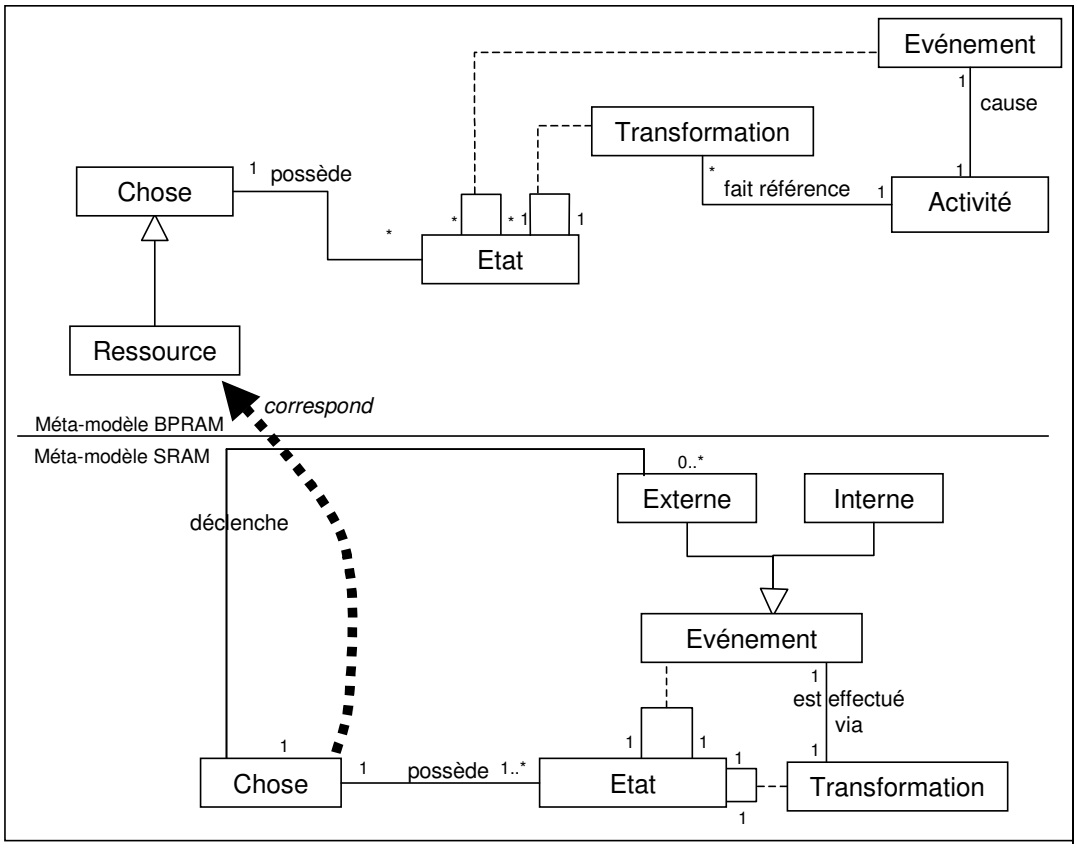


Figure 23 : Parties des méta-modèles BPRAM et SRAM nécessaires à la définition de la Présence des ressources

La métrique (Pr) associée au critère de présence des ressources se présente sous la forme d'un rapport. Au dénominateur, R_p est l'ensemble des ressources utilisées dans le modèle de processus d'entreprise M_{proc}. Au numérateur, R_p^m est un ensemble des ressources du modèle de processus, *correspond* à une chose du système qui ne provoque pas de changement d'état sur une autre chose du système. R_p^m est donc l'ensemble des éléments *r* de R_p tels que pour chacun de ces éléments, il existe, dans l'ensemble C_s des choses du modèle du système M_{syst}, une chose *c* (i) qui soit liée à *r* par un lien *correspond* et (ii) qui ne subisse pas de changement d'état provoqué par une autre chose *c'*.

Formellement, soit :

$$Pr(M_{proc}, M_{syst}) = card(R_p^m) / card\ (R_p)$$

Avec :

- $R_p = \{r \in M_{proc} \mid r.type(t) = Ressource\}$
- $R_p^m = \{r, r \in R_p \mid \exists\ c \in C_s \wedge r\ \mathcal{M}\ c\ \wedge \forall\ c' \in C_s \wedge \forall\ (e, e') \in E_{s,c'}^2 \wedge \neg\ (c\ \delta <e,e'>)]\ \}$

Une faible valeur pour cette métrique signifie qu'un grand nombre de ressources du processus d'entreprise ne sont pas prises en charge par le système. Certaines informations utiles à la réalisation des processus ne sont donc pas disponibles dans le système. Il est alors nécessaire d'introduire dans le système des choses qui peuvent être liées aux différentes ressources non supportées par un lien *correspond*. Il se peut également que la chose existe déjà dans le système mais qu'elle ne puisse pas être reliée à la ressource du modèle de processus par un

lien *correspond* : cela veut dire que les propriétés de la chose et de la ressource ne sont pas identiques. Dans ce cas, une modification des propriétés de la ressource ou de la chose pourra suffire et permettre que la ressource soit représentée dans le système.

3.4.2 Critères et métriques liés au facteur informationnel

Le *facteur informationnel* évalue l'alignement en s'intéressant à la façon dont l'information manipulée par les processus d'entreprise est gérée par le système. Nous définissons deux critères, *Complétude de l'information* et *Exactitude de l'information* que nous présentons dans les deux sections suivantes ainsi que leur métrique associée.

3.4.2.1 Complétude de l'information

L'alignement du système et des processus d'entreprise repose sur l'existence d'une forte correspondance entre l'information gérée par les processus et celle manipulée par le système. Le critère de *Complétude de l'information* calcule le taux de choses présentes dans le modèle de processus qui sont spécifiées dans le système.

Les méta-modèles BPRAM et SRAM manipulent tous deux l'information au moyen de choses. Nous cherchons donc à mesurer la proportion de choses présentes dans le modèle de processus qui *correspondent* à des choses du modèle du système par rapport au nombre total de choses présentes dans le modèle de processus. Si cette proportion atteint l'unité, cela signifie que l'information est complètement gérée par le système. C'est de là que vient le terme *Complétude de l'information* que nous utilisons pour nommer ce critère.

Dans le cas de la gestion des réservations de chambres d'hôtel, les concepts de chambre, d'hôtel, de station, de disponibilité, de réservation ou de demande peuvent être considérés comme des choses du modèle de processus.

Le critère *Complétude de l'information* calcule la proportion des choses du modèle de processus M_{proc} qui existent en tant que chose dans le modèle du système M_{syst}. Dans notre exemple, si, dans le système, la chambre est une propriété de la chose hôtel, la chose chambre du modèle de processus n'est pas gérée par le système. Les propriétés et les états de la chose chambre ne peuvent pas être modifiés.

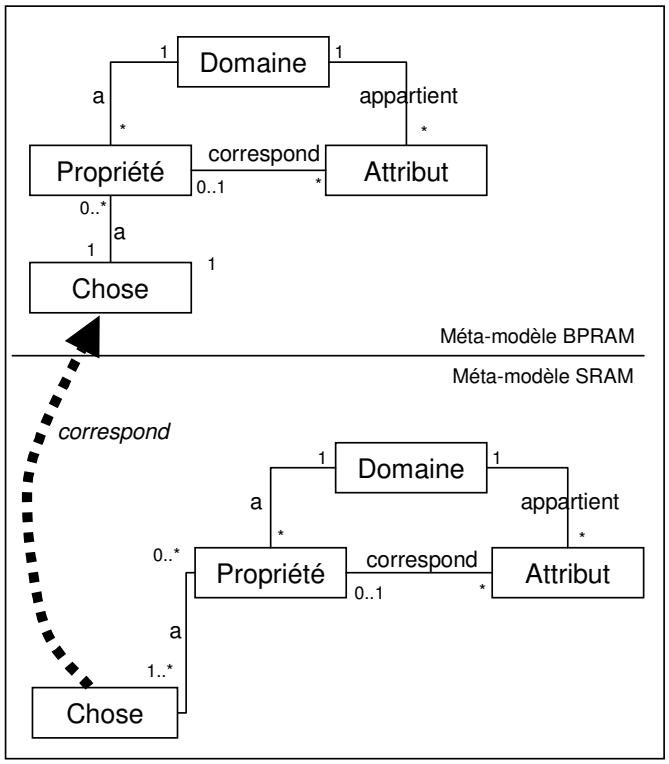


Figure 24 : Parties des méta-modèles BPRAM et SRAM nécessaires à la définition de la Complétude de l'information

Formellement, soit C_p, l'ensemble des choses du modèle de processus d'entreprise M_{proc}. L'ensemble C_p^m des choses du modèle de processus d'entreprise supportées par le système est un sous-ensemble de C_p où pour chaque chose *c* il existe une chose *c'* du modèle du système M_{syst} qui peut être liée à *c* par un lien *correspond*. $C_p^m = \{c, c \in C_p \mid \exists\, c' \in C_s \wedge c \mathcal{M} c'\}$

La métrique (Ci) associée au critère *Complétude de l'information* se définit comme suit :

$$Ci\ (M_{proc}, M_{syst}) = card\ (C_p^m) / card\ (C_p)$$

Une valeur de cette métrique égale à 1 signifie que chaque chose du modèle de processus *correspond* à une chose du modèle de système. En d'autres termes, le système d'information gère toute l'information manipulée par les processus d'entreprise. Cette valeur peut diminuer, par exemple si de nouvelles choses sont introduites dans les processus d'entreprise, sans que cela soit répercuté dans le système. Dans le cas de la gestion des réservations de chambres d'hôtel, supposons que l'on décide de considérer le concept de chambre comme une chose à part entière (et non plus seulement comme une propriété de l'hôtel) pour permettre au client de choisir sa chambre. Si cette modification n'est pas répercutée au niveau du système, la valeur de la *Complétude de l'information* baisse. Pour retrouver une "bonne" complétude, une solution consiste à modifier le système afin qu'il manipule des choses correspondant aux nouvelles choses ou supprimer des processus d'entreprise ces choses s'il s'avère qu'elles ont peu d'importance. Il peut également être possible de réifier une propriété d'une chose pour en faire une autre chose, car l'information peut être présente dans le système mais pas sous forme de chose (comme c'est le cas dans notre exemple pour la chambre).

3.4.2.2 Exactitude de l'information

Il est bien sûr important que les processus d'entreprise et le système manipulent la même information, mais cela ne suffit pas. Intuitivement, il semble que l'information doive être gérée de la « même façon » dans les processus d'entreprise et dans le système. La « même façon » signifie que le cycle de vie des choses doit pouvoir être le même pour les choses du modèle de processus et celles du modèle du système qui leur *correspondent*. Or le cycle de vie d'une chose est défini à partir des états de cette chose. Le critère d'*Exactitude de l'information* mesure la proportion des états de choses du modèle de processus supportés par le système.

Les concepts d'état et de chose sont présents aussi bien dans le méta-modèle BPRAM que dans le méta-modèle SRAM (comme le montre la Figure 25). Ainsi, pour gérer l'information de la même façon dans les processus d'entreprise et dans le système il est nécessaire que les états des choses du modèle de processus *correspondent* à des états de choses du modèle du système. On considère que deux états se *correspondent* si les choses auxquelles ils sont associés se *correspondent* et si les valeurs qui les définissent sont les mêmes.

Ainsi, par exemple, dans le cas de gestion de commande, il est important que les différents états d'une commande à savoir : 'initiée', 'en attente', 'complète', 'payée' et 'envoyée' soient reconnus par le système. Ceci permet de savoir comment une commande évolue au cours du temps et quels états elle traverse. Des événements et des activités dépendent de ces états. Si l'état 'complète' n'existe pas dans le système, il se peut que la commande ne soit pas délivrée, le paiement ne soit pas demandé et le stock ne soit pas géré ou tout simplement que la commande arrive incomplète chez le client.

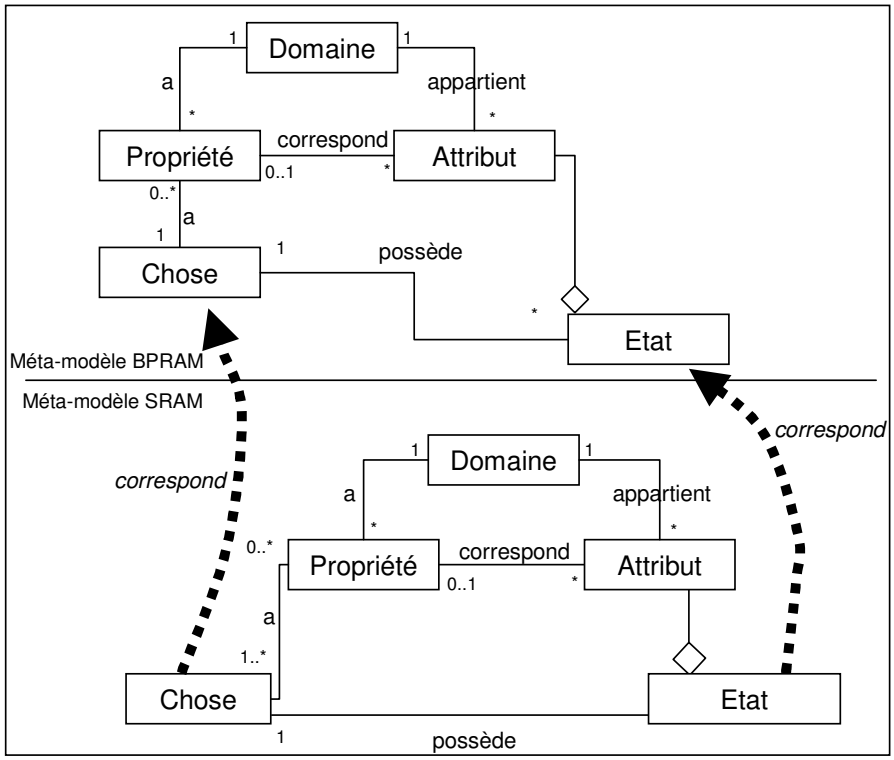


Figure 25 : Parties des méta-modèles BPRAM et SRAM nécessaires à la définition de l'Exactitude de l'information

En considérant que E_p représente l'ensemble des états des choses présentes dans le modèle de processus M_{proc} et E_p^m l'ensemble des états des choses présentes dans le modèle de processus dont chaque élément *correspond* à un état de choses du modèle du système M_{syst}. E_p^m est le sous-ensemble de E_p ne contenant que les états des choses *e* présents dans M_{proc} pour lesquels il existe un état de choses *e'* présent dans le modèle du système M_{syst} qui lui *correspond*.

Nous définissons la métrique (Ei) associée au critère *Exactitude de l'information* comme suit :

$$Ei\ (M_{proc}, M_{syst}) = card(E_p^m) / card\ (E_p)$$

avec : $E_p^m = \{e, e \in E_p \mid \exists\ e' \in E_s \wedge e\ \mathcal{M}\ e'\}$.

Compte tenu de la définition du lien *correspond* entre états, il apparaît clairement qu'une faible valeur du critère *Complétude de l'information* entraîne une faible valeur du critère *Exactitude de l'information*. En effet, si une chose du modèle de processus n'est pas supportée par le système, ses états ne le sont pas non plus. Par exemple, dans le cas de la gestion de réservations de chambres d'hôtel, si la chose chambre est présente dans les processus d'entreprise, mais n'est pas supportée par le système, ses états 'libre', 'réservée', 'en travaux' ne peuvent pas être supportés par le système. Cependant le contraire n'est pas vrai. Des choses du modèle de processus peuvent être liées à des choses du modèle du système par un lien correspond sans qu'il en soit de même pour leurs états. Dans notre exemple, il se peut que la chambre existe en tant que chose dans le modèle du système, mais que l'état 'en travaux' ne soit pas supporté par le système. Une telle situation détectée par la mesure de *l'Exactitude de l'information* doit être corrigée car sinon des cycles de vie, des événements d'entreprise, des lois de transition, ou des chemins des processus ne pourront pas être mis en œuvre. Dans notre exemple, la fermeture pour travaux d'une partie ou de la totalité de l'hôtel n'est pas gérée par le système, il est alors difficile pour les utilisateurs du système mais aussi pour le gérant d'hôtel de connaître le nombre de chambres disponibles à la location.

3.4.3 Critères et métriques liés au facteur fonctionnel

Le *facteur fonctionnel* permet d'évaluer le degré selon lequel chaque activité présente dans le modèle de processus est supportée par le système. Cette évaluation repose sur le rôle des choses et de leurs états dans les activités du modèle de processus. Le facteur fonctionnel a deux critères qui lui sont associés, la *Complétude de l'activité* et l'*Exactitude* de *l'activité*. Ces deux critères s'appliquent individuellement à chaque activité présente dans le modèle de processus. Les métriques associées à ces deux facteurs ont donc chacune trois paramètres : un modèle de processus, un modèle du système et une activité de ce modèle de processus. Il est possible d'étudier toutes les activités ou seulement celles qui sont les plus utilisées ou les plus rentables afin d'analyser les éventuels problèmes d'alignement au sein de ces activités.

3.4.3.1 Complétude de l'activité

Le critère de *Complétude de l'activité* permet d'étudier les détails de chacune des activités des processus d'entreprise. Il exprime le degré selon lequel l'information utilisée pour une activité donnée est présente dans le modèle du système. Ce critère mesure la proportion des choses intervenant dans cette activité qui sont supportées par le système.

La métrique (Ac) associée au critère *Complétude de l'activité* est le quotient du nombre de choses d'une activité du modèle de processus M_{proc} *correspondant* chacune à une chose du modèle de système M_{syst} sur le nombre total de choses de cette activité (Figure 26). Dans l'exemple de la gestion de réservations de chambres d'hôtel, si on s'intéresse à l'activité d'allocation de chambre, le critère *Complétude de l'activité* évalue dans quelle proportion les choses Chambre, Hôtel, Réservation, Client, Demande et Disponibilité ont des choses *correspondantes* dans le modèle de système.

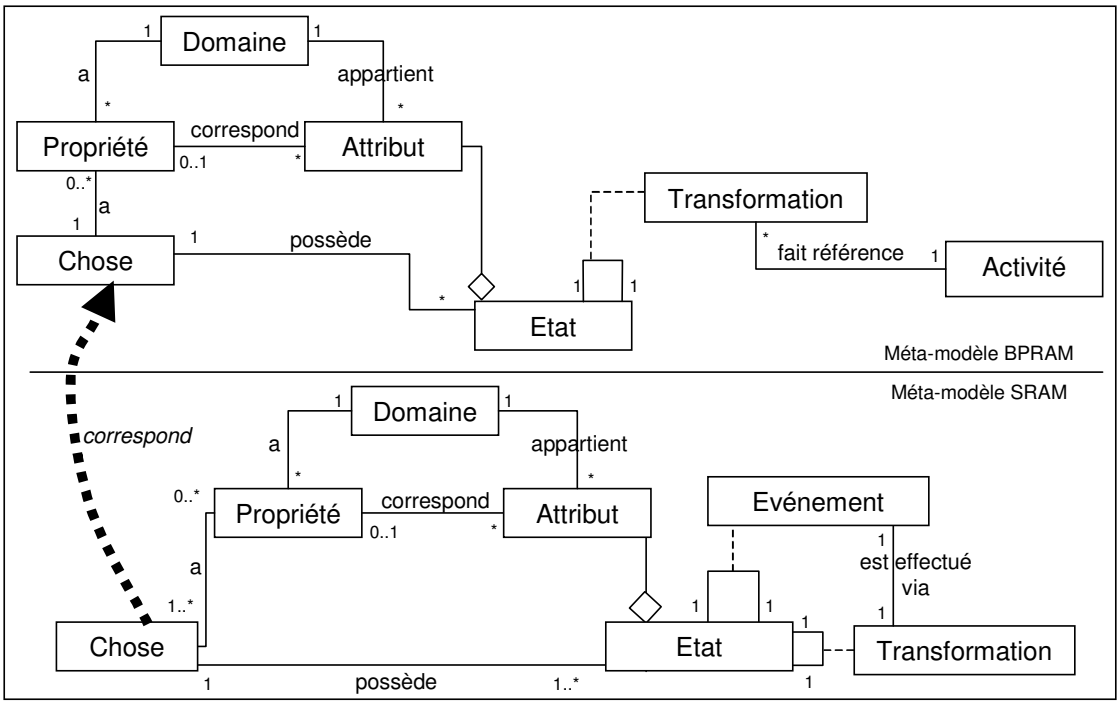


Figure 26 : Parties des méta-modèles BPRAM et SRAM nécessaires à la définition de la Complétude de l'activité

De façon formelle, soit C_a^m l'ensemble des choses intervenant dans l'activité *a* du modèle de processus M_{proc} telles que chacune *correspond* à une chose du modèle de système M_{syst}. Cet ensemble est un sous ensemble de l'ensemble C_a des choses de l'activité *a* tel que pour chaque chose *c*, il existe une chose du modèle du système *c'* qui *correspond* à *c*.

Si :

- $C_a = \{c \in a \mid c.type(t) = Chose \wedge a.type(t') = Activité\}$
- $C_a^m = \{c, c \in C_a \mid \exists\, c' \in C_s \wedge c\ \mathcal{M}\ c'\}$

Alors la métrique associée à ce critère s'écrit :

$$Ac(a, M_{proc}, M_{syst}) = card(C_a^m) / card(C_a)$$

Une faible valeur de cette métrique signifie que l'activité étudiée est mal prise en charge par le système et plus précisément que l'information nécessaire à cette activité n'est pas correctement gérée au niveau du système. Pour augmenter la valeur de ce critère, il est nécessaire d'introduire dans le système des choses qui *correspondent* aux choses de l'activité qui ne sont pas supportées. Il se peut que ces choses existent déjà dans le modèle de système

sous d'autres aspects par exemple celui de propriété. Dans ce cas, il est, par exemple, possible de réifier des propriétés de certaines choses afin que plus de choses de l'activité soient implémentées.

3.4.3.2 Exactitude de l'activité

Comme pour le critère précédent, on s'intéresse ici au détail de l'activité. Le critère *Exactitude de l'activité* analyse, pour une activité donnée, dans quelle proportion l'information manipulée par les processus est gérée de façon similaire dans le système. Ce critère calcule au sein d'une activité la proportion des états des choses de l'activité qui sont implémentés dans le système.

La métrique (Ae) associée au critère d'*Exactutude de l'activité* est le quotient du nombre des états de choses présentes dans cette activité qui *correspondent* à un état d'une chose du modèle du système sur le nombre total des états des choses présentes dans cette activité. La Figure 27 montre les différents éléments des méta-modèles BPRAM et SRAM intervenant dans la définition de l'*Exactitude de l'activité*.

Dans le cas de la gestion des réservations des chambres d'hôtel, l'activité d'allocation de chambre permet de modifier l'état de la Chambre de 'libre' à 'réservée', l'état de la Demande de 'en attente' à 'satisfaite', de créer une Réservation avec un état 'en cours', de supprimer une Disponibilité pour un Client 'en cours' et un Hôtel 'ouvert'.

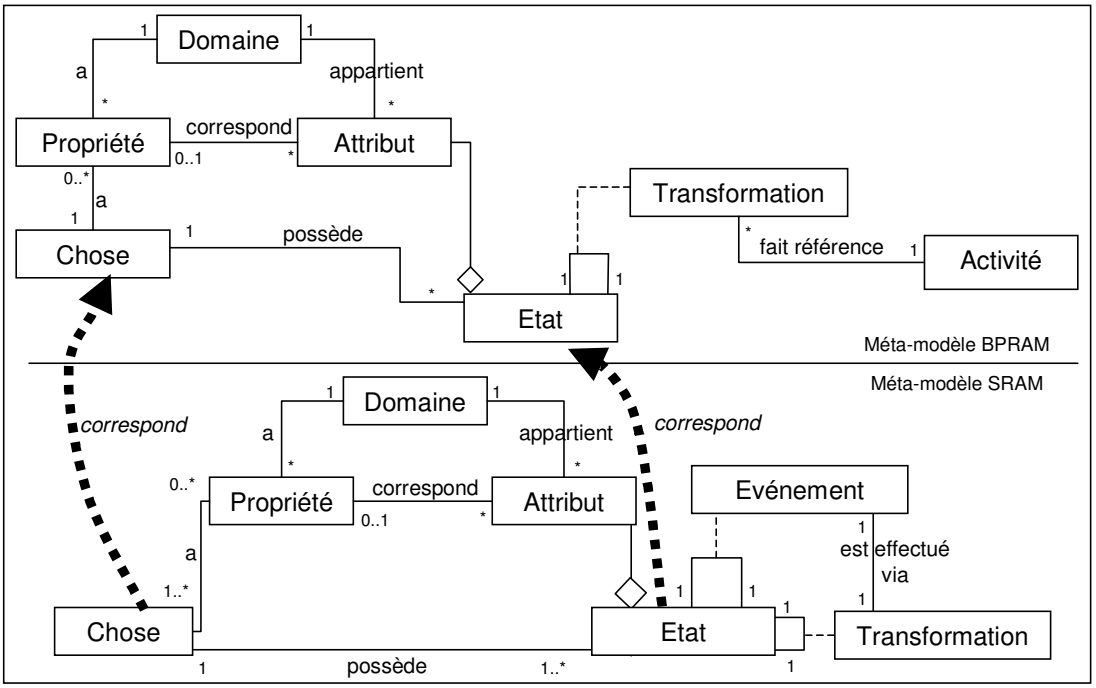


Figure 27 : Parties des méta-modèles BPRAM et SRAM nécessaires à la définition de l'Exactitude de l'activité

Formellement, E_a^m l'ensemble des états des choses intervenant dans l'activité *a* supportés par le système est un sous-ensemble de E_a, l'ensemble des états des choses intervenant dans l'activité *a*. Dans E_a^m seuls sont pris en considération les états *e* de E_a pour lesquels il existe un état *e'* d'une chose du modèle du système tel que ces deux états *e* et *e'* se *correspondent*.

Soit :

- $E_a = \{e \in a \mid e.type(t) = Etat \wedge a.type(t') = Activité\}$
- $E_a^m = \{e, e \in E_a \mid \exists e' \in E_s \wedge e \mathcal{M} e'\}$

La métrique associée à ce critère s'écrit :

$$Ae(a, M_{proc}, M_{syst}) = card(E_a^m) / card(E_a)$$

De la même façon que pour les critères de *Complétude* et d'*Exactitude de l'information*, il existe une dépendance entre les valeurs de *Complétude de l'activité* et d'*Exactitude de l'activité*. En effet, un état ne peut être supporté par le système que si la chose à laquelle il est associé est elle-même représentée dans le système.

Si une chose présente dans le modèle de processus n'est pas représentée dans le système, c'est-à-dire qu'il n'existe pas de chose dans le système qui lui *corresponde*, alors ses états ne peuvent pas être implémentés dans le système. En d'autres termes l'analyse du critère *Exactitude de l'activité* va de pair avec celle de la *Complétude de l'activité*. Si une faible valeur est observée pour chacune des métriques associées à ces critères, il est nécessaire de chercher d'abord à rétablir une forte valeur pour la *Complétude de l'activité* en introduisant dans le système des choses qui *correspondent* aux choses non gérées de l'activité. L'introduction de ces choses et de ces états devrait augmenter la valeur de la métrique associée au critère *Exactitude de l'activité*. Si cette valeur est néanmoins jugée trop faible, il est nécessaire d'introduire dans le système des états de choses qui *correspondent* aux états de choses de l'activité qui ne sont pas supportés.

3.4.4 Critères et métriques associés au facteur dynamique

Le facteur dynamique et les deux critères, *Fiabilité du système* et *Réalisme dynamique*, qui lui sont associés étudient les processus d'entreprise et le système d'un point de vue dynamique en s'intéressant aux successions d'états et d'activités.

3.4.4.1 Fiabilité du système

Les deux facteurs précédents prennent en considération les choses et / ou les états indépendamment et séparément. Le critère *fiabilité du système* considère la succession d'états de choses du modèle de processus et vérifie non seulement que chaque état est bien supporté mais aussi que leur succession au niveau du système est possible.

Le critère *Fiabilité du système* est défini pour mesurer l'alignement entre les transformations du modèle de processus d'entreprise et celles du système. Le concept de transformation du méta-modèle BPRAM permet d'établir un ordre entre deux états (Figure 28). La métrique (Fs) associée au critère de *Fiabilité du système* calcule la proportion des transformations du modèle de processus d'entreprise M_{proc} qui sont prises en charge par le système comparativement à l'ensemble des transformations du modèle de processus d'entreprise. On dira qu'une transformation est prise en charge (1) si chaque état d'une chose du modèle de processus impliqué dans la transformation *correspond* à un état d'une chose du modèle du système M_{syst} et (2) si chaque transition d'état dans M_{proc} est possible au niveau système.

Considérons l'exemple du processus d'entreprise dans lequel une commande ne peut être payée que si elle est à l'état 'complète' (le client ne paie qu'à la livraison). Il existe donc une transformation qui permet à une commande de passer de l'état 'complète' à l'état 'payée'. Le critère *Fiabilité du système* permet de vérifier que cette transformation existe dans le système. De plus, il apporte une information complémentaire à celle fournie par les critères *Complétude de l'information* et *Exactitude de l'information*puisqu'il vérifie, si les états des choses du modèle de processus sont implémentés dans le système, et si la transition entre ces différents états est possible dans le système.

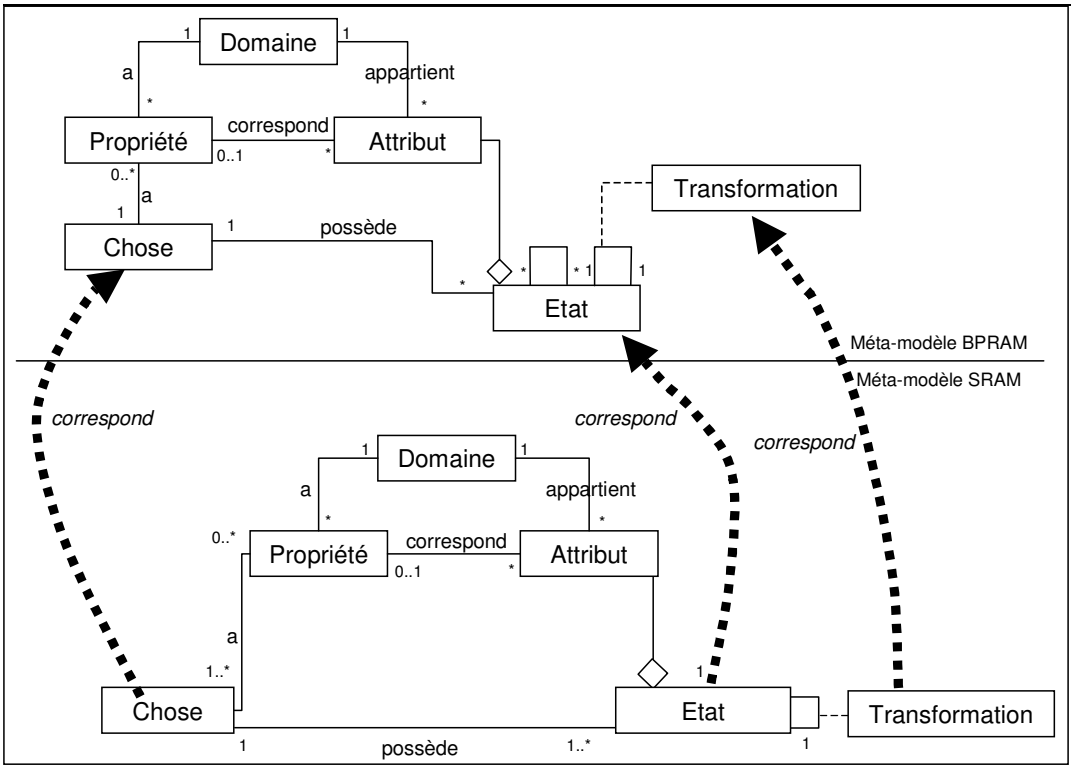


Figure 28 : Parties des méta-modèles BPRAM et SRAM nécessaires à la définition de la Fiabilité du système

Formellement, soit T_p l'ensemble des transformations du modèle de processus d'entreprise M_{proc}. On considère T_p^m le sous-ensemble de T_p formé des transformations implémentées dans le système, c'est-à-dire des transformations *tr* telles que pour chaque état e_i (respectivement e_j) intervenant dans cette transformation ($e_i \in tr$) il existe un état e'_i (respectivement e'_j) d'une chose du modèle du système M_{syst} qui lui *correspond* et telle que la transition d'états $<e_i, e_j>$ est possible entre les états du modèle du système, c'est-à-dire que la transition $<e'_i, e'_j>$ existe. La métrique associée au critère *Fiabilité du système* s'écrit de la façon suivante :

$$Fs(M_{proc}, M_{syst}) = card(T_p^m) / card\ (T_p)$$

Avec :

- $T_p = \{tr \in M_{proc} \mid tr.type(t) = Transformation\}$
- $T_p^m = \{tr, tr \in T_p \mid \forall\ (e_i, e_j) \in E_p^2, e_i \in tr, e_j \in tr, \exists\ (e'_i, e'_j) \in E_s^2 \wedge e_i\ \mathcal{M}\ e'_i \wedge e_j\ \mathcal{M}\ e'_j \wedge <e_i,e_j> \wedge <e'_i, e'_j> \}$

Cette métrique peut être interprétée de la façon suivante : une faible valeur montre que le système ne se comporte pas en miroir des processus d'entreprise, que les transitions d'états ne sont pas possibles dans le système. Afin d'accroître cette valeur il est nécessaire de s'assurer que les états des choses du modèle de processus sont convenablement implémentés dans le système. Si l'état initial ou l'état final d'une transformation n'existe pas dans le système, la transformation ne pourra pas elle-même être gérée par le système. Si, au contraire, les différents états de la transformation du modèle de processus *correspondent* à des états de choses du modèle du système, on peut augmenter la valeur de la métrique *Fiabilité du système* en permettant dans le système les transitions d'état entre les différents états en jeu.

3.4.4.2 Réalisme dynamique

De la même façon qu'on mesure l'alignement entre les transformations des processus d'entreprise et les transformations du système, il semble important et complémentaire d'évaluer si la succession des activités de l'entreprise est adéquatement prise en charge par le système. Le critère *Réalisme dynamique* mesure la proportion des chemins du modèle de processus gérés par le système.

Dans le méta-modèle BPRAM, un chemin est un ensemble fini d'états de choses du modèle de processus $\{e_1,\ldots,e_n\}$ tel qu'il existe une transformation t permettant de passer de l'état e_k à l'état e_{k+1} quelque soit k entier naturel compris entre 1 et n-1. Un chemin est considéré comme supporté si chaque état qui le constitue *correspond* à un état d'une chose du système et si la suite des états formant le chemin existe au niveau du système (Figure 29).

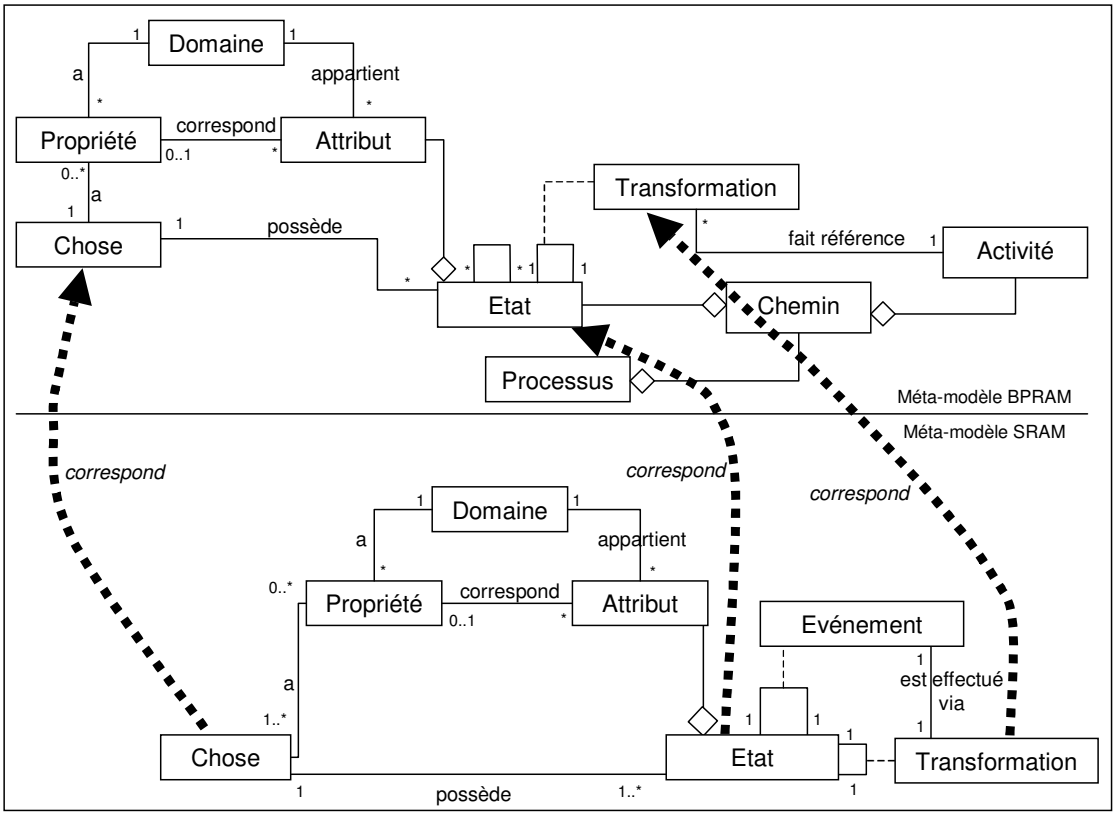


Figure 29 : Parties des méta-modèles BPRAM et SRAM nécessaires à la définition du Réalisme dynamique

Formellement soit P_p l'ensemble des chemins des processus d'entreprise et P_p^m le sous ensemble de P_p correspondant à l'ensemble des chemins gérés par le système. P_p^m est l'ensemble des chemins *p* constitués de la suite d'états $(e_1,\ldots,e_n)$ distincts deux à deux tels que pour tout k compris entre 1 et n-1, s'il existe une transformation dans le modèle de processus entre e_k et e_{k+1}, non seulement il existe deux états de choses du système e_k' et e_{k+1}' qui *correspondent* respectivement à e_k et e_{k+1}, mais aussi, la transition d'états de e_k' à e_{k+1}' est possible dans le système. La métrique associée au critère *Réalisme dynamique* s'écrit de la façon suivante.

Avec :

- $T_p = \{tr \in M_{proc} \mid tr.type(t) = Transformation\}$
- $P_p = \{p \in M_{proc} \mid p.type(t) = Chemin\}$,
- $P_p^m = \{p, p \in P_p \wedge p = \{e_1,\ldots,e_n\} \wedge \forall i,\ e_i \in E_p \wedge e_i \neq e_j \wedge \forall k \in [\![1, n-1]\!]\ (\exists t_k \in T_p \wedge e_{k+1} = t_k(e_k) \Rightarrow \exists e_k', e_{k+1}' \in E_s \wedge e_k \mathcal{M} e_k' \wedge e_{k+1} \mathcal{M} e_{k+1}' \wedge <e_k', e_{k+1}'>)\}$

$$\text{alors } Rd(M_{proc}, M_{syst}) = card(P_p^m) / card(P_p)$$

Une faible valeur de cette métrique montre que le système ne permet pas de réaliser ce qui est prévu dans le modèle de processus. Son origine peut-être due à de faibles valeurs d'autres critères comme celui de *Complétude de l'information* ou *d'Exactitude de l'information*. Si les états des modèles de processus ne sont pas présents dans le système, les chemins ne peuvent pas exister. Les réajustements de l'alignement doivent alors d'abord prendre en compte les autres critères avant d'analyser le critère de *Réalisme dynamique* en lui même. Pour augmenter cette valeur, soit le modèle d'entreprise doit être modifié en supprimant par exemple, les chemins ou les parties de chemins rarement exécutés, soit le système doit être adapté afin qu'il traduise la diversité et la dynamicité des processus.

Résumé. Nous avons présenté dix métriques définies par référence aux concepts du méta-modèle BPRAM et ceux du méta-modèle SRAM. Ces métriques montrent que l'alignement peut être évalué selon plusieurs critères complémentaires. Elles servent également de guide pour générer des métriques entre deux méta-modèles spécifiques permettant de représenter respectivement les processus d'entreprise et le système. Le processus de génération des métriques spécifiques est décrit puis illustré à la section suivante.

4. Processus de génération de métriques spécifiques

Dans cette section, dans un premier temps nous présentons comment le système générique de métriques d'alignement présenté à la section précédente permet de générer un ensemble de métriques spécifiques. Dans un deuxième temps, nous illustrons ce processus pour définir des métriques spécifiques entre le diagramme d'activités UML [UML] et le méta-modèle O* [Lee97] permettant de représenter respectivement les processus d'entreprise et le système.

4.1. Processus de génération

Le mécanisme général du processus de génération de métriques spécifiques est celui de la mise en relation entre les concepts des méta-modèles génériques BPRAM et SRAM et ceux des méta-modèles spécifiques. La Figure 30 schématise ce processus.

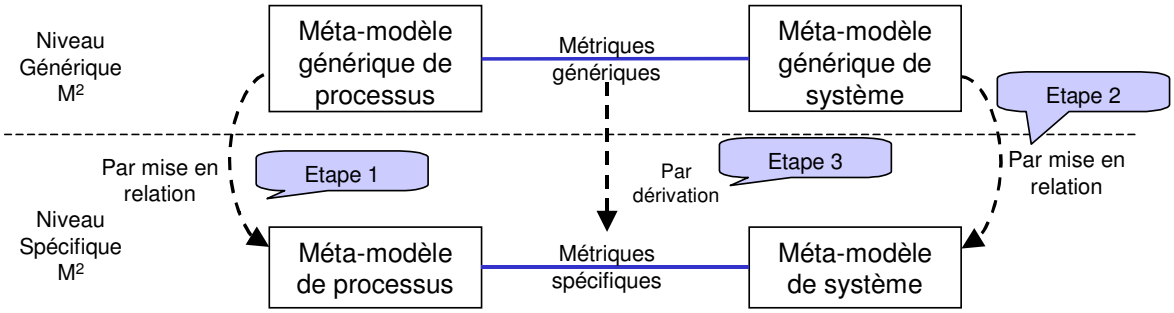


Figure 30 : Processus de génération de métriques spécifiques d'alignement

Le processus de génération de métriques spécifiques se compose de trois étapes [Etien05b] :

1. Associer les concepts du méta-modèle de processus choisi à ceux du méta-modèle BPRAM.
2. Associer les concepts du méta-modèle de système choisi à ceux du méta-modèle SRAM.
3. Adapter les métriques génériques en conséquence.

Les deux premières étapes du processus s'appuient sur la notion de mise en relation telle qu'elle est utilisée dans [Rosemann02]. Dans cet article, les auteurs cherchent à comparer deux méta-modèles spécifiques : le méta-modèle EPC (Event-driven Process Chains) [Scheer98] et le diagramme d'activité UML [UML]. Pour cela, ils utilisent comme référence le méta-modèle BWW défini à partir de l'ontologie de Bunge, Wand et Weber. En effet, plutôt que de comparer directement les deux méta-modèles, ils mettent d'abord en relation les concepts de chacun de ces deux méta-modèles avec ceux du méta-modèle BWW. Cette mise en relation permet ainsi de comparer non seulement des concepts mais aussi des structures de méta-modèles différents en s'appuyant sur les liens entre concepts (on parle alors de "*pattern matching*").

Les étapes 1 et 2 du processus de génération de métriques spécifiques ont pour but de mettre en relation les concepts des méta-modèles spécifiques avec ceux des méta-modèles génériques. Il ne s'agit pas ici de chercher à mettre en relation tous les concepts des méta-modèles spécifiques avec ceux des méta-modèles génériques. Il s'agit, au contraire, d'associer aux concepts intervenant dans les métriques génériques des concepts des méta-modèles spécifiques. De la même façon que [Rosemann02], nous nous intéressons également à la mise en relation de parties de méta-modèle.

La troisième étape consiste à adapter les métriques génériques en remplaçant chaque concept générique par le concept spécifique auquel il est associé. Les mises en relation des structures génériques et spécifiques permettent de définir les métriques spécifiques.

Le processus que nous proposons permet d'être systématique dans la production des métriques spécifiques.

La section suivante est consacrée à l'illustration du processus de génération de métriques spécifiques associées au diagramme d'activités UML et au méta-modèle O* représentant les processus d'entreprise et le système.

4.2. Illustration du processus de génération de métriques spécifiques

Pour modéliser les processus d'entreprise et le système, nous avons choisi le diagramme d'activités d'UML [UML] et le méta-modèle O* [Rolland96], [Lee97] respectivement. Le diagramme d'activités UML est souvent utilisé pour représenter les processus d'entreprise. S'autres diagrammes UML auraient pu être utilisés pour représenter le système, cependant, la gestion des événements en UML n'est pas forcément naturelle. Nous avons préféré représenter le système avec O* qui permet une description statique du système en adoptant une modélisation orientée-objet mais permet également de modéliser l'aspect dynamique en mettant l'accent sur la représentation des événements et des changements d'états. De plus, en adoptant deux méta-modèles relativement différents pour représenter le système et les processus, nous montrons que la définition des métriques spécifiques ne nécessite pas forcément des méta-modèles partageant certains concepts. Nous pensons que ce choix illustre la réalité industrielle où système et processus sont souvent modélisés avec des méta-modèles très différents.

Le reste de la section est divisée en trois parties correspondant à chacune des trois étapes du processus de génération des métriques spécifiques.

4.2.1 Etape 1 : Associer les concepts du diagramme d'activités UML à ceux du méta-modèle BPRAM

Le diagramme d'activités UML permet de représenter un processus sous la forme d'un ensemble d'activités exécutées par des acteurs. Les activités représentent les étapes particulières du processus. Elles sont associées à des transitions d'états qui permettent de modifier l'état d'un objet d'une classe. En général deux activités sont reliées entre elles par une transition. Lorsqu'une activité se termine, la transition est automatiquement enclenchée et l'activité suivante démarre. Une activité n'est donc pas déclenchée par un événement ; ce concept n'existant pas dans le diagramme d'activités UML. Plusieurs activités peuvent exprimer des alternatives. Leur réalisation dépend de conditions booléennes, mutuellement exclusives. Elles sont précédées par une décision. Deux séquences d'activités peuvent également être réalisées simultanément dans ce cas, deux synchronisations permettent respectivement de marquer le début et la fin de chacune de ces séquences.

Le diagramme d'activités UML utilise donc les concepts (i) d'activité et d'acteur pour préciser qui réalise l'activité, (ii) d'objet et d'état pour spécifier le produit des activités, (iii) de transition pour exprimer le passage d'une activité à une autre.

La Figure 31 présente un exemple de diagramme d'activités UML. Dans la colonne d'activités de gauche se trouvent toutes les activités que l'acteur 1 réalise. De façon symétrique, la colonne d'activités de droite rassemble les activités exécutées par l'acteur 2. La

réalisation des activités 2 et 3 est conditionnée par C. L'activité 2 amène l'objet2 dans l'état e". Les activités 5 et 6 s'exécutent en parallèle.

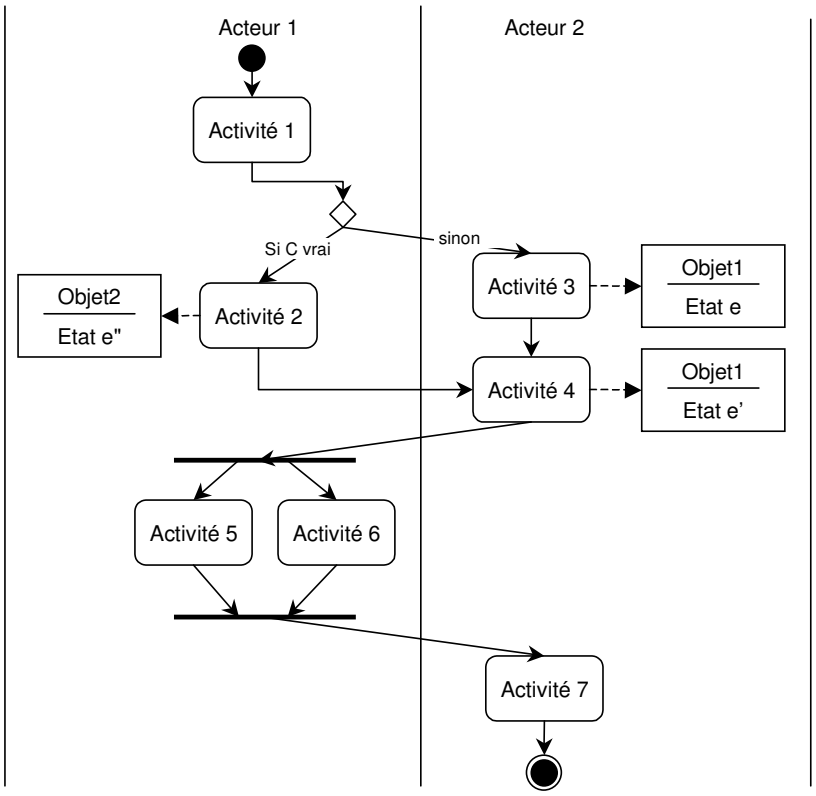


Figure 31 : Exemple de diagramme d'activités UML

Le concept de but n'est pas explicite dans le diagramme d'activités UML, alors qu'il est précisément spécifié dans le méta-modèle BPRAM. Ce concept permet de préciser explicitement les états des choses que le processus cherche à atteindre. Dans la mesure où les états des objets peuvent être précisés dans un diagramme d'activités UML, il est possible de déterminer les buts que les processus permettent d'atteindre. En effet, le but d'un processus représenté par un diagramme d'activités correspond à l'ensemble des états finaux de chaque objet, c'est-à-dire l'état dans lequel se trouve chaque objet à la fin du processus.

La Figure 32 représente les concepts du diagramme d'activités UML ainsi que les relations qui existent entre certains de ces concepts et ceux du méta-modèle BPRAM. La Figure 32 est divisée en deux parties. La partie supérieure correspond au méta-modèle BPRAM tel qu'il a été décrit précédemment. La partie inférieure présente les concepts du diagramme d'activités UML. Les flèches en gras permettent de préciser les relations entre les concepts du méta-modèle BPRAM et ceux du diagramme d'activités UML.

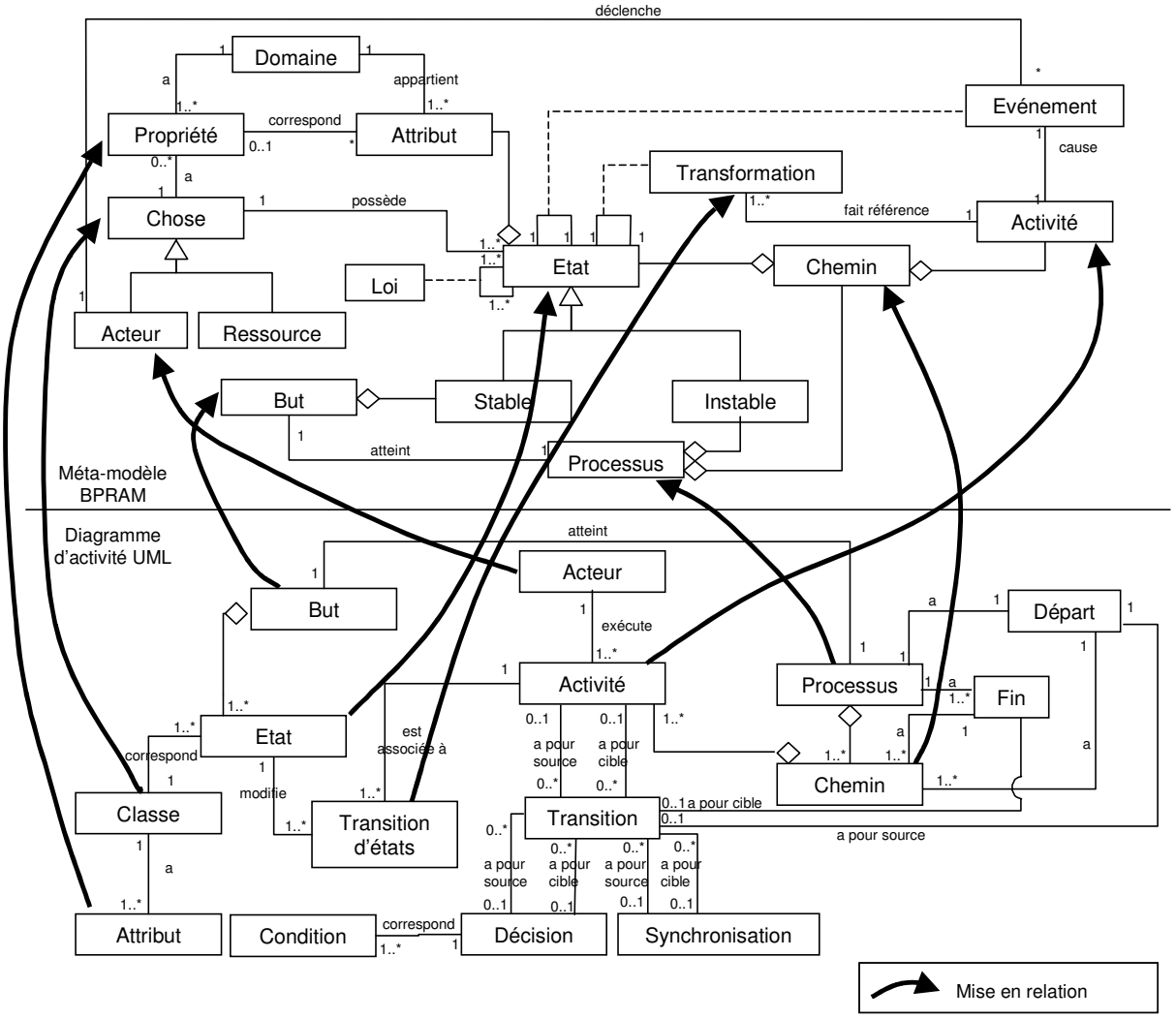


Figure 32 : Mise en relation des concepts du diagramme d'activités UML avec ceux du méta-modèle BPRAM

Les concepts de classe UML et de chose BPRAM sont tous deux définis comme un ensemble d'éléments du niveau instance ayant des propriétés communes. Ces deux concepts sont donc mis en relation. Les concepts d'attribut UML et de propriété BPRAM qui permettent de définir respectivement les caractéritiques d'une classe et d'une chose, sont également mis en relation.

Un état dans un diagramme d'activités UML permet de préciser une condition ou une situation qu'un objet satisfait durant son cycle de vie. Les transitions d'états permettent de passer d'un état à un autre ; ce sont des relations entre états. Ces deux concepts correspondent respectivement aux concepts BPRAM d'état et de transformation. En effet, une transformation est un lien entre deux états où un état est un vecteur de valeur de toutes les propriétés d'une classe.

Les concepts d'activité UML et d'activité BPRAM font référence au concept de transition d'état et de transformation respectivement. Ces deux concepts peuvent donc être mis en relation. De la même façon, les concepts de chemin et de processus UML correspondent aux concepts de chemin et de processus BPRAM. En effet, dans les deux cas, les processus sont considérés comme des ensembles de chemins, qui eux-mêmes sont perçus comme des

ensembles d'activités. Ainsi, non seulement des concepts UML et BPRAM sont mis en relation, mais aussi des parties de méta-modèles. On peut ainsi s'apercevoir que dans le méta-modèle BPRAM comme dans le diagramme d'activités UML, un chemin peut être perçu comme un ensemble d'états reliés par des transformations (respectivement des transitions d'états).

Dans le méta-modèle BPRAM, un processus atteint un ensemble d'états stables appelé but. Dans le diagramme d'activités UML, le but n'est pas explicite, mais il correspond à l'ensemble des états des objets obtenus une fois que toutes les activités ont été réalisées. Les concepts de but UML et de but BPRAM sont donc mis en relation.

Dans le diagramme d'activités UML, l'acteur exécute les activités. En d'autres termes, on peut considérer que l'acteur en UML provoque un changement d'état d'une ou plusieurs classes. Le concept d'acteur BPRAM est défini comme une chose à l'interface des processus d'entreprise et de leur environnement qui déclenche des événements externes, c'est-à-dire provoque le changement d'états d'autres choses. Les concepts d'acteurs UML et BPRAM peuvent donc être mis en relation.

En revanche, le concept BPRAM de ressource, c'est-à-dire de chose (i) utile à la réalisation d'une activité, (ii) qui ne provoque aucun changement d'état, n'existe pas dans le diagramme d'activités UML. Aucune des métriques utilisant ce concept ne peut être définie. Concrètement seule la métrique associée au critère *Présence des ressources* ne peut pas être spécifiée.

Pour représenter le système, nous avons choisi le méta-modèle O*. La section suivante décrit la deuxième étape du processus de génération des métriques spécifiques.

4.2.2 Etape 2 : Associer les concepts du méta-modèle de système choisi à ceux du méta-modèle SRAM

O* est un méta-modèle qui permet une représentation conceptuelle du système à développer, en utilisant une approche orientée objet. En O*, un *objet* est perçu comme subissant des changements provoqués par des événements. La spécification d'une classe objet dépasse donc la traditionnelle description avec des attributs et des méthodes pour inclure la description des événements. Ces derniers sont vus comme des changements d'états d'objets qui déclenchent des opérations sur d'autres objets, changent leurs états et ensuite éventuellement génèrent d'autres événements et ainsi de suite.

La Figure 33 illustre à gauche ce principe en utilisant un mode graphique. Un changement d'état de O1 déclenche l'événement EV1 qui provoque les opérations OP21 sur O2 ; OP31 sur O3 si la condition C1 est satisfaite, OP33 sinon et OP12 sur lui-même. A droite, la Figure 33 esquisse la spécification d'une classe en O*.

La spécification d'une classe compte cinq parties : (i) les propriétés de la classe, (ii) les assertions, c'est-à-dire les contraintes qui existent sur ces propriétés, (iii) les opérations, (iv) le graphe de transitions d'états des objets de la classe et (v) les événements déclenchés par la classe décrite.

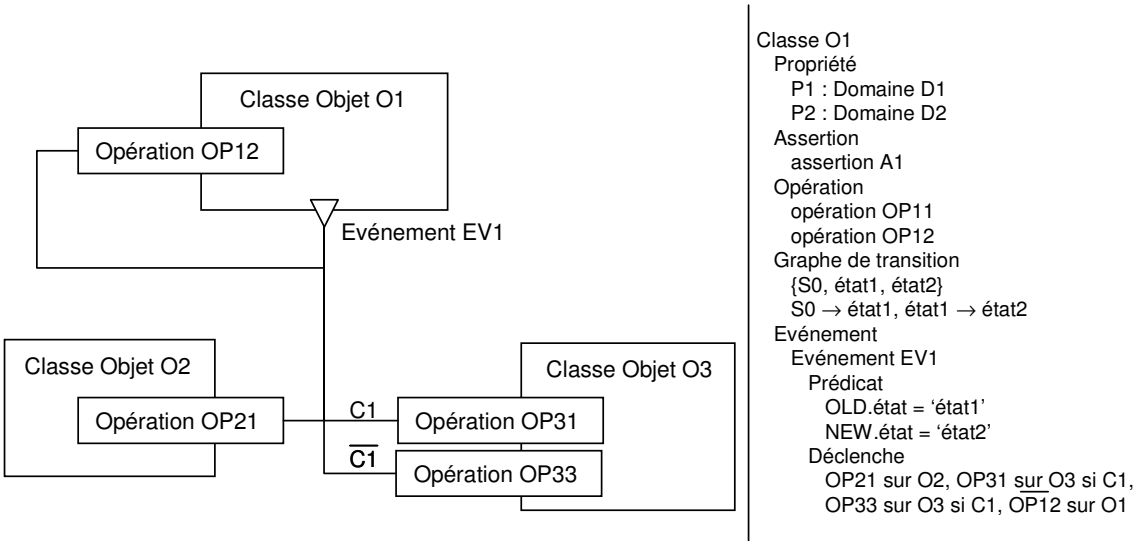


Figure 33 : Description d'une classe objet

Le méta-modèle SRAM et le méta-modèle O* permettent tous deux de décrire un système d'un point de vue statique et d'un point de vue dynamique. La partie statique fournit une description des liens durables entre les concepts (comme par exemple les liens de composition). La partie dynamique permet une description du comportement des classes en réponse à des événements issus de l'environnement ou du SI lui-même.

En O*, les deux concepts clé sont les concepts de classe objet et d'événement. Une classe objet est caractérisée par des propriétés, éventuellement des assertions (c'est-à-dire des contraintes statiques), des opérations dont l'exécution change l'état des objets de la classe, les événements que déclenche la classe objet et un graphe de transitions d'états. Un graphe de transitions d'états est un ensemble d'états reliés par des transitions. Ainsi, une transition d'états correspond à un état initial, une opération et un état final ; l'opération permettant de passer d'un état à l'autre. D'un point de vue statique, les classes objet peuvent être reliées entre elles de trois façons différentes : par un lien d'héritage, de composition ou de référence. Le concept d'événement permet de spécifier explicitement ce qui déclenche une ou plusieurs opérations regroupées au sein d'une transition dynamique. Les événements sont de trois types : (i) un événement interne est défini dans la classe des objets dont il constate un changement d'état, (ii) un événement temporel est défini dans une classe particulière nommée Horloge, qui regroupe tous les événements temporels et (iii) un événement externe est défini dans une classe de type Acteur. Il peut y avoir plusieurs classes de ce type dans un modèle, une pour chacun des agents à l'interface du système et de son environnement.

La Figure 34 est divisée en deux parties : (i) la partie supérieure correspond au méta-modèle SRAM et (ii) la partie inférieure représente le méta-modèle O* avec la notation UML.

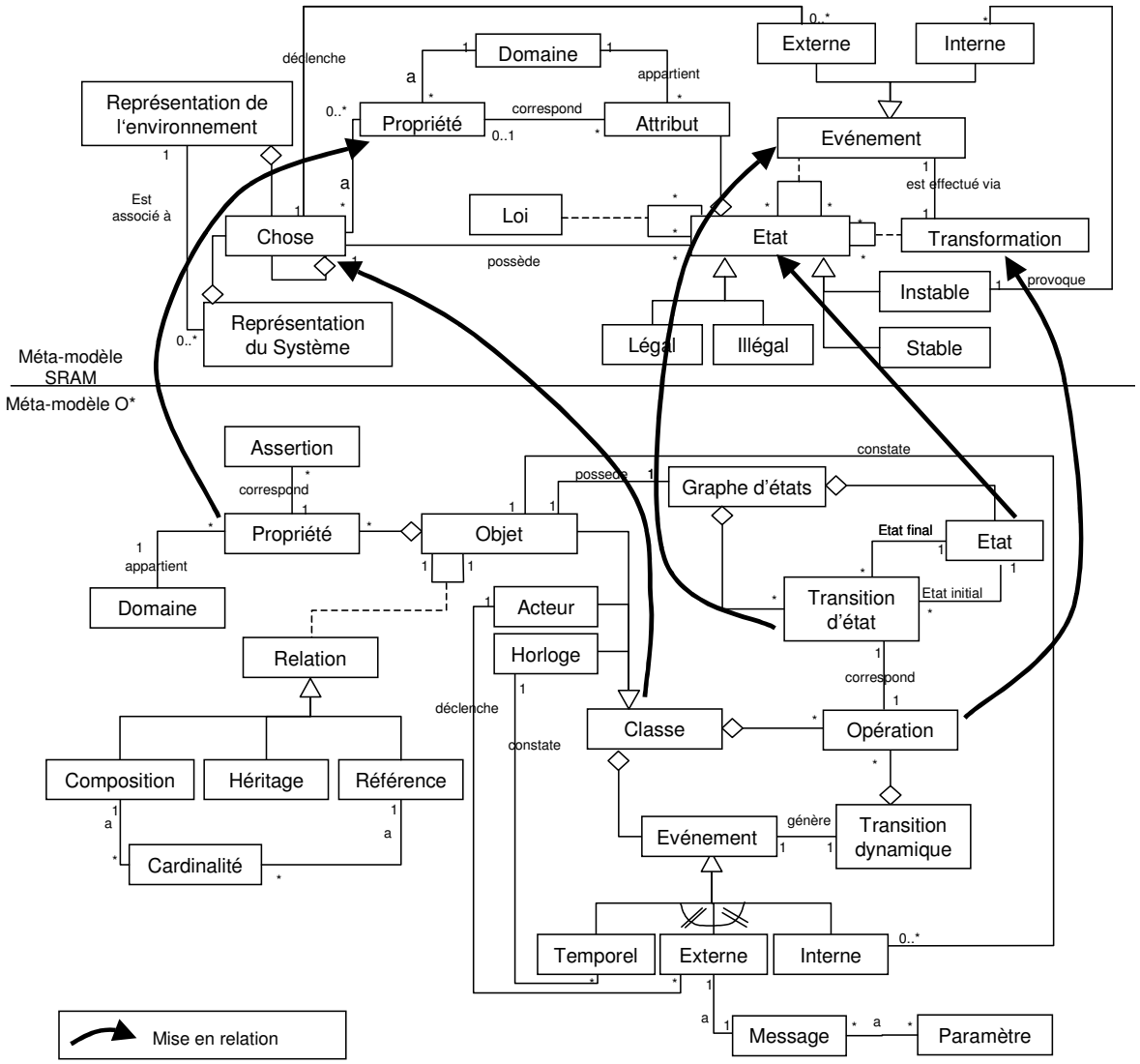


Figure 34 : Spécialisation du méta-modèle SRAM avec le méta-modèle O*

La notion de *classe objet* de O* est mise en relation avec le concept de *chose* du méta-modèle SRAM. Tous deux ont des *propriétés* et voient leurs *états* changer par l'application *d'opérations* en réponse à des événements. La notion *d'opération* O* et le concept de *transformation* du méta-modèle SRAM peuvent donc être mis en relation.

La description des événements en O* comprend la cause de la survenance d'une occurrence d'événement et son impact sur les objets. Le concept O* de transition d'états est un triplet <état initial, opération, état final>. Le concept SRAM d'événement se définit comme un couple d'états <s, s'> où s et s' sont deux états d'une même chose tel qu'il existe une transformation t et s'= t(s). C'est donc le concept O* de *transition d'états* qui est mis en relation avec le concept *d'événement* SRAM.

Au cours des deux premières étapes du processus, des concepts du diagramme d'activités UML et du méta-modèle O* ont été mis en relation respectivement avec des concepts génériques des méta-modèles BPRAM et SRAM. Ces correspondances servent de support à la génération des métriques spécifiques à partir des métriques génériques. La troisième étape du processus permet de construire les métriques spécifiques.

4.2.3 Etape 3 : génération des métriques spécifiques associées au diagramme d'activités UML et au méta-modèle O*

Les métriques génériques permettent de mesurer l'alignement entre les modèles de processus d'entreprise et le modèle du système qui les supporte. Elles prennent chacune pour référence un concept du méta-modèle BPRAM et le concept du méta-modèle SRAM auquel il est lié via un lien *correspond* ou *représente*. Les métriques spécifiques jouent le même rôle et adoptent la même structure que les métriques génériques.

Hormis pour le concept BPRAM de ressource, un concept du diagramme d'activités UML a pu être mis en relation avec chacun des concepts BPRAM intervenant dans la définition d'une ou plusieurs métriques génériques. Symétriquement, un concept du méta-modèle O* a été relié à chaque concept SRAM. Ainsi, seules neuf métriques peuvent être associées au diagramme d'activités UML et à O*.

Pour générer les métriques spécifiques à partir des métriques génériques, on relie les concepts UML et O* qui spécialisent respectivement les concepts BPRAM et SRAM intervenant dans la métrique générique par le même lien type (*correspond* ou *représente*) que celui qui existe entre les concepts génériques.

Ainsi, au niveau générique, la métrique associée au critère *Complétude de l'information* est définie comme suit :

$$Ci(M_{proc}, M_{syst}) = \frac{\text{Nombre de choses du modèle de processus } \textit{correspondant} \text{ chacune à une chose du modèle du système}}{\text{Nombre de choses du modèle de processus}}$$

Le concept BPRAM de chose est en relation avec le concept de classe UML et le concept de chose SRAM avec celui de classe objet O*.

Au niveau spécifique, la métrique associée au critère *Complétude de l'information* s'écrit :

$$Ci(M_{proc}, M_{syst}) = \frac{\text{Nombre de classes de } M_{proc} \textit{ correspondant} \text{ chacune à une classe objet du modèle O* } M_{syst}}{\text{Nombre des classes du diagramme d'activité UML } M_{proc}}$$

Les neuf autres métriques spécifiques associées au diagramme d'activités UML et au méta-modèle O* sont générées de la même façon.

Le Tableau 7 présente informellement les neuf métriques pour mesurer l'alignement entre des processus d'entreprise représentés par des diagrammes d'activités UML et des spécifications du système décrites avec O*.

Facteurs	Critères	Descriptions
Alignement Intentionnel	Taux de Support	nombre d'activités UML *représentées* chacune par une transition d'états O* / nombre d'activités
	Satisfaction des buts	nombre de buts pour lesquels chaque état *correspond* à un état d'une classe objet O* / nombre de buts
	Présence des Acteurs	nombre d'acteurs du diagramme d'activités UML qui *correspondent* chacun à une classe objet déclenchant un événement dans le modèle O* / nombre d'acteurs du schéma de processus
	Présence des ressources	Ne peut être définie.
Alignement Informationnel	Complétude de l'information	nombre de classes du diagramme d'activités UML *correspondant* chacune à une classe objet du modèle O* / nombre de classes UML
	Exactitude de l'information	nombre d'états de classe UML *correspondant* chacun à un état d'une classe objet O* / nombre d'états de classe UML
Alignement Fonctionnel	Complétude de l'activité	nombre de classes d'une activité UML *a correspondant* chacune à une classe objet du modèle O* / nombre de classes UML de l'activité *a*
	Exactitude de l'activité	nombre d'états de classe d'une activité UML *a correspondant* chacun à un état d'une classe objet O* / nombre d'états de classe de l'activité UML *a*
Alignement Dynamique	Fiabilité du système	nombre de transitions UML pour lesquelles chaque état *correspond* à un état O* et les transitions entre les états des classes UML sont possible entre états O* / nombre de transitions UML
	Réalisme dynamique	nombre de chemins pour lesquels chaque état *correspond* à un état O* et la succession de ces états O* est possible / nombre de chemins

Tableau 7 : Métriques spécifiques pour mesurer l'alignement du système et des processus décrits avec O* et UML.

De façon similaire aux métriques génériques, chaque métrique spécifique est une fonction ayant pour paramètre M_{proc}, un modèle de processus et M_{syst} un modèle de système instanciant respectivement le méta-modèle de processus d'entreprise et le méta-modèle du système choisis (ici en l'occurrence les diagrammes d'activités UML et le méta-modèle O*). Les métriques associées aux critères *Complétude de l'activité* et *Exactitude de l'activité* compte un troisième paramètre l'activité *a* qu'on étudie. Une métrique s'écrit de façon formelle comme le quotient card(A)/card(B) où :

- B est un ensemble d'éléments présents dans le modèle de processus M_{proc} décrit par des diagrammes d'activités UML ;
- A est un sous-ensemble de B. A fait référence à l'ensemble des éléments de B qui sont représentés dans le système, c'est-à-dire tel que, pour chaque élément de B, il existe un élément du modèle du système M_{syst} représenté avec le méta-modèle O* qui soit lié par un lien *correspond* ou *représente* (suivant la métrique) à l'élément de B.
- le mot clé card spécifie qu'on calcule le nombre d'éléments de l'ensemble étudié. La métrique peut donc s'écrire nombre d'éléments de A / nombre d'éléments de B.
- les ensembles A et B adoptent la même convention d'écriture qu'au niveau générique.

Ainsi, par exemple, nous avons vu que la métrique associée au critère *Complétude de l'information* se définit au niveau spécifique de la façon suivante :

nombre de classes UML *correspondant* à une classe objet O* / nombre de classes UML

- B correspond ici à l'ensemble des classes utilisées dans les diagrammes d'activités UML pour décrire les processus.
- A fait référence à l'ensemble de ces classes qui sont liées par un lien *correspond* à une classe objet du modèle O* du système.
- B et A sont nommés de façon plus expressive respectivement C_p (pour l'ensemble des classes des diagrammes d'activités UML) et C_p^m (la lettre m en exposant fait référence au lien *correspond*, map en anglais).

La métrique spécifique associée au critère *Complétude de l'information* se définit formellement de la façon suivante :

$$Ci(M_{proc}, M_{syst}) = card\ (C_p^m) / card\ (C_p)$$

Avec :

- C_p, l'ensemble des classes des diagrammes d'activités UML
- C_s, l'ensemble des classes objet du modèle du système M_{syst}
- C_p^m l'ensemble des classes des diagrammes d'activités UML qui *correspondent* à une classe objet du système. $C_p^m = \{c, c \in C_p \mid \exists\ c' \in C_s \wedge c\ \mathcal{M}\ c'\}$

De façon similaire, il est possible de définir formellement chacune des neuf métriques spécifiques associées au diagramme d'activités UML et au méta-modèle O* pour représenter respectivement les processus d'entreprise et le système. Le Tableau 8 présente les définitions formelles de ces neuf métriques spécifiques.

Critères	Métriques	Définition formelle
Taux de Support	$Ts(M_{proc}, M_{syst}) = card(A_p^r) / card(A_p)$	- A_p, l'ensemble des activités du diagramme d'activités, - Te_s l'ensemble des transitions d'états du modèle du système - A_p^r, l'ensemble des activités UML tel que pour chaque élément il existe une transition d'états du système la *représentant* ; $A_p^r = \{a, a \in A_p \mid \exists\ te \in Te_s \wedge te\ \Re\ a\}$
Satisfaction des buts	$Sb(M_{proc}, M_{syst}) = card(B_p^m) / card(B_p)$	- B_p, l'ensemble des buts - E_p, l'ensemble des états des classes présentes dans le diagramme d'activités M_{proc} - E_s, l'ensemble des états des classes objet du modèle du système M_{syst} - B_p^m l'ensemble des buts du business gérés par le système, c'est-à-dire pour lesquels chaque état intervenant *correspond* à un état dans le système. $B_p^m = \{b \in B_p \mid \forall e \in E_p \wedge e \in b \Rightarrow \exists\ e' \in E_s \wedge e\ \mathcal{M}\ e'\}$
Présence des acteurs	$Pa(M_{proc}, M_{syst}) = card(Ac_p^m) / card(Ac_p)$	- Ac_p, l'ensemble des acteurs du business - Ac_s, l'ensemble des acteurs O* - Ac_p^m, l'ensemble des acteurs du système *correspondant* chacun à une classe acteur O*. $Ac_p^m = \{ac, ac \in Ac_p \mid \exists\ ac', ac' \in Ac_s \wedge ac\ \mathcal{M}\ ac'\}$
Complétude de l'information	$Ci(M_{proc}, M_{syst}) = card\ (C_p^m) / card\ (C_p)$	- C_p, l'ensemble des classes des diagrammes d'activités UML M_{proc} - C_s, l'ensemble des classes objet du modèle du système M_{syst} - C_p^m l'ensemble des classes des diagrammes d'activités UML qui *correspondent* chacune à une classe objet du système. $C_p^m = \{c, c \in C_p \mid \exists\ c' \in C_s \wedge c\ \mathcal{M}\ c'\}$
Exactitude de l'information	$Ei(M_{proc}, M_{syst}) = card(E_p^m) / card\ (E_p)$	- E_p, l'ensemble des états des classes présentes dans le diagramme d'activités M_{proc} - E_s, l'ensemble des états des classes objet du modèle du système M_{syst} - E_p^m, l'ensemble des états des classes présentes dans le diagramme d'activités tel que chaque élément *correspond* à un état d'une classe objet du système.

		$E_p^m = \{e, e \in E_p \mid \exists\ e' \in E_s \wedge e\ \mathcal{M}\ e'\}$
Complétude de l'activité	$Ac(a, M_{proc}, M_{syst}) = card(C_a^m) / card(C_a)$	- C_a, l'ensemble des classes de l'activité *a* - C_s, l'ensemble des classes objet du système - C_a^m l'ensemble des classes intervenant dans l'activité *a* qui *correspondent* chacune à une classe objet du système. $C_a^m = \{c, c \in C_a \mid \exists\ c' \in C_s \wedge c\ \mathcal{M}\ c'\}$
Exactitude de l'activité	$Ae(a, M_{proc}, M_{syst}) = card(E_a^m) / card(E_a)$	- E_a, l'ensemble des états de classe intervenant dans l'activité *a* du diagramme d'activité M_{proc} - E_s, l'ensemble des états de classe d'objet de M_{syst} - E_a^m l'ensemble des états intervenant dans l'activité *a* tel que chaque élément *correspond* à un état du système. $E_a^m = \{e, e \in E_a \mid \exists\ e' \in E_s \wedge e\ \mathcal{M}\ e'\}$
Fiabilité du système	$Fs(M_{proc}, M_{syst}) = card(T_p^m) / card\ (T_p)$	- T_p, l'ensemble des transitions définies dans les diagrammes d'activité UML M_{proc} - E_p représente l'ensemble des états de classes du modèle UML M_{proc} - E_s représente l'ensemble des états de classes objet O* du modèle M_{syst} - T_p^m l'ensemble des transitions implémentées dans le système. $T_p^m = \{t, t \in T_p \mid \forall\ e_i, e_j \in E_p, e_i \in t, e_j \in t, \exists\ e'_i, e'_j \in E_s \wedge e_i\ \mathcal{M}\ e_i' \wedge e_j\ \mathcal{M}\ e'_j \wedge <e_i,e_j> \wedge <e'_i, e'_j> \}$
Réalisme dynamique	$Rd(M_{proc}, M_{syst}) = card(P_p^m) / card(P_p)$	- E_p représente l'ensemble des états des classes UML - E_s représente l'ensemble des états des classes Objet O* - T_p, l'ensemble des transitions définies dans les diagrammes d'activités UML - P_p, l'ensemble des chemins du diagramme d'activités P_p^m l'ensemble des chemins présents dans le système. $P_p^m = \{p, p \in P_p \wedge p = \{e_1,\ldots,e_n\} \wedge \forall\ i,\ e_i \in E_p \wedge e_i \neq e_j \wedge \exists\ t \in T_p \wedge e_{k+1} = t(e_k) \mid \forall\ k, \exists\ e_k', e_{k+1}' \in E_s \wedge e_k\ \mathcal{M}\ e_k' \wedge e_{k+1}\ \mathcal{M}\ e_{k+1}' \wedge <e_k', e_{k+1}'> \}$

Tableau 8 : Définition formelle des métriques spécifiques associées au diagramme d'activités UML et au méta-modèle O*

Le processus de génération de métriques spécifiques est terminé. Le chapitre suivant présente un exemple de calcul et d'interprétation de ces métriques pour un cas de gestion de réservation hôtelière.

5. Utilisation de seuil et de poids

Les métriques permettent d'attribuer une valeur au concept abstrait qu'est l'alignement entre les processus et le système. Ceci permet d'explorer toutes les nuances qui existent entre parfaitement et pas du tout aligné.

Afin d'aider à analyser la valeur de la mesure, il peut s'avérer pertinent de fixer un seuil en dessous duquel le mauvais alignement entre les processus et le système n'est plus tolérable. Des actions correctives doivent alors être entreprises pour que la valeur des différentes métriques repasse au dessus des différents seuils fixés.

De plus, les valeurs des différentes métriques sont calculées en considérant que chaque élément (c'est-à-dire, par exemple, chaque activité pour la métrique associée au *Taux de support* ou chaque chose pour la *Complétude de l'information*) a la même importance. Il est néanmoins possible d'attribuer un poids différent à chacun de ces éléments.

Afin de mieux apprécier les valeurs des différentes métriques, nous introduisons deux notions, les *seuils* et les *poids*. Les deux sections suivantes détaillent l'intérêt de ces deux notions.

5.1.1 Le seuil pour déterminer quand mettre en œuvre des actions correctives

Chaque métrique permet d'évaluer l'alignement entre les processus et le système selon un certain critère. Si la valeur obtenue pour la métrique égale 1, cela signifie que l'alignement selon ce critère est parfait. Cependant cette valeur est rarement atteinte. L'alignement n'est alors que partiel. Les décideurs peuvent éventuellement considérer que la situation est satisfaisante et que son amélioration ne justifie pas le coût qu'elle engendrerait.

L'introduction de *seuils* permet de fixer une valeur en dessous de laquelle l'alignement est considéré comme mauvais et nécessite la mise en œuvre d'actions correctives. Le principe de seuil permet de mieux apprécier la relation d'alignement [Bodhuin04] en ne restreignant pas l'alignement aux cas où chaque valeur des métriques égale 1. Ce seuil est fixé par les décideurs en fonction de leurs connaissances et de leur appréciation du projet.

5.1.2 Le poids pour apprécier chaque élément en fonction de son importance réelle

Les métriques que nous avons définies donnent une vision de la relation d'alignement où tous les éléments d'un même type ont la même importance. Ainsi, par exemple la métrique *Taux de support* calcule le taux d'activités des processus qui sont prises en charge par le système. Chaque activité est considérée de la même façon, qu'elle soit exécutée quotidiennement ou une fois dans tout le cycle de vie des processus, qu'elle soit la plus rentable ou non… Afin de considérer chacune des activités différemment par rapport aux autres, nous proposons d'introduire des *poids* dans la définition des métriques.

De cette façon, la métrique spécifique Taux de support peut s'écrire :

$$Ts_p = \sum \alpha^r_i \ / \sum \alpha_j$$

Avec

- α_j est le poids associé à l'activité a_j (α_i est une valeur positive),
- α^r_i est le poids associé à l'activité a^r_i (α^r_i est positif),
- a_i une activité appartenant à A_p, l'ensemble des activités,
- a^r_i une activité appartenant à $A_p{}^r$, l'ensemble des activités UML tel que pour chacune d'entre elles, il existe un événement du système la *représentant* ;

$A_p{}^r = \{a, a \in A_p \mid \exists ev \in Ev_s \wedge ev \Re a\}$

Les activités supportées par le système étant des activités des processus, pour chaque activité supportée α^r_i est égal à α_i. Certaines activités n'étant pas prises en charge, leurs poids α_i ne se retrouvent donc pas au numérateur de cette métrique pondérée. De cette façon, on obtient bien, comme pour les métriques définies précédemment une valeur comprise entre 0 et 1.

Les α_i peuvent être définis de multiples façons. Par exemple, il est possible de considérer les α_i comme la fréquence d'exécution ou le taux de rentabilité de chaque activité. Les α_i peuvent aussi être spécifiés de façon plus subjective par les parties prenantes (décideurs et utilisateurs) pour faire référence à la satisfaction des clients ou l'utilisabilité…

Il est possible de définir une formule pondérée pour chaque métrique spécifique ; les poids peuvent être spécifiés de façons différentes suivant les métriques.

6. Conclusion

Ce chapitre propose une définition de l'alignement qui repose sur deux types de liens *correspond* et *représente* que nous avons définis en nous inspirant de différents liens existant dans la littérature. Nous avons défini la notion abstraite d'alignement et l'avons ensuite matérialisée en lui associant un ensemble de dix critères. A chacun de ces critères nous avons ensuite associé une métrique formellement définie. Ces métriques s'expriment entre un modèle de processus et un modèle de système. Elles s'exécutent sur les éléments des modèles de processus et du système.

Les métriques sont d'abord définies à un niveau générique entre le méta-modèle BPRAM et le méta-modèle SRAM, méta-modèles construits à partir des ontologies de Bunge, Wand Weber et de Soffer et Wand. Ces métriques génériques servent de support à un processus de génération de métriques spécifiques associées à deux méta-modèles spécifiques permettant de respectivement représenter des processus d'entreprise et un système. La définition de ces métriques est néanmoins conditionnée par l'existence dans les méta-modèles spécifiques de concepts spécialisant les concepts BPRAM et SRAM intervenant dans la définition des métriques génériques. Nous avons illustré le processus de génération et montré l'utilité de chacune des métriques pouvant être définies pour le diagramme d'activités UML et le méta-modèle O*.

Les métriques permettent d'exploiter la nuance qui peut exister entre parfaitement aligné et pas du tout aligné. Afin d'aller plus loin dans cette idée et d'aider les décideurs à mieux apprécier quand des actions correctives s'imposent pour rétablir l'alignement, nous avons introduit deux notions, le *seuil* et le *poids*. Le *seuil* permet de fixer une limite inférieure en deçà de laquelle la rupture de l'alignement n'est plus acceptable. Le *poids* permet d'attribuer un coefficient particulier à chaque élément de l'ensemble étudié afin de modifier la métrique en fonction de leur importance réelle. Les *seuils* et les *poids* sont définis par les décideurs en fonction de leur connaissance et de leur appréciation du projet.

CHAPITRE 4 : EXEMPLE DE MESURE DE L'ALIGNEMENT

1. Introduction

Dans ce chapitre, nous présentons un cas d'application qui illustre l'utilisation des métriques.

Ce cas est partiellement fictif. Il correspond à la gestion de ressources hôtelières, depuis la création de la liste des produits jusqu'à la consommation par le client. Il a pour but de montrer comment les mesures d'alignement sont calculées et interprétées.

Ce chapitre est organisé de la façon suivante : la section 2 présente le cas d'application ; la section 3 correspond au calcul et à l'interprétation des mesures. La section 4 conclut ce chapitre.

2. Description du cas d'application

Le cas d'application étudié dans ce chapitre correspond à l'analyse d'une application de gestion de réservations de chambres d'hôtels dans des stations de ski françaises. Un ensemble d'hôteliers de la région alpine ont décidé, il y a plusieurs années de devenir partenaires et de centraliser la gestion de leurs réservations afin de lutter contre la concurrence des grandes chaînes hôtelières multi-nationales. Ce partenariat a abouti à la construction d'un processus et d'un système d'information.

Les décideurs souhaitent analyser la situation actuelle afin de comprendre pourquoi leur performance n'est pas aussi bonne que celle qu'ils espéraient. Nous leur suggérons d'évaluer quantitativement l'alignement entre les diagrammes d'activités UML et le système (modélisé avec O*).

2.1. Description des processus d'entreprise

Les processus sont organisés en deux parties correspondant (i) à la création et la gestion de la liste de produits et (ii) à la gestion des réservations de chambres d'hôtel. Nous présentons ici une partie du processus de gestion des réservations : de la création de la demande à la facturation de la réservation. Les autres processus sont présentés dans l'annexe 1.

2.1.1 Présentation générale du processus de gestion des réservations de chambres d'hôtel

Toute personne désireuse de faire une réservation peut téléphoner afin de réserver des chambres. L'opérateur chargé de la réservation lui demande plusieurs renseignements : nom, prénom, adresse, numéro de téléphone, numéro de carte de crédit, type carte, date d'expiration

et diverses indications sur la demande : la période de réservation, le nombre de chambres, la catégorie d'hôtel et la station désirée.

L'opérateur interroge l'application pour vérifier si la demande peut-être satisfaite. Si aucune disponibilité ne permet de satisfaire la demande, l'opérateur sollicite le client pour qu'il formule éventuellement une nouvelle demande ou lui suggère de mettre sa demande en attente. S'il est possible de satisfaire la demande, une réservation comportant tous les renseignements permettant d'envoyer une lettre de confirmation au client est créée.

Les annulations de réservation sont acceptées sans contrepartie financière si elles parviennent au système de réservation au moins une semaine avant le début de la réservation. Après ce délai, le client doit payer 75% du prix.

2.1.2 Diagramme d'activités du processus de gestion de réservations de chambres d'hôtel

Les processus sont représentés en utilisant les diagrammes d'activités UML. La Figure 35 présente le processus de gestion des réservations de chambres d'hôtel. Ce processus se répartit sur deux lignes d'activités, celle du client et celle de l'opérateur.

Le client initie le processus en effectuant une demande, ce qui crée une instance de la classe Demande et met son état à 'créée'. L'opérateur vérifie ensuite si des disponibilités permettent de satisfaire cette demande. Deux cas se présentent :

1. il n'existe pas de possibilités. Dans ce cas, deux alternatives sont possibles :

   - le client a refusé de mettre sa demande en attente, l'état de la demande est modifié, il est désormais égal à 'sans suite' et le processus se termine ;
   - le client a accepté de mettre sa demande en attente et, dans ce cas, l'instance de la classe Demande correspondant au client est modifiée, son état devient 'attente'. Trois chemins différents permettent de continuer le processus : (i) le délai de mise en attente est dépassé, la demande correspond à une période de réservation de chambre débutant moins de quinze jours après la date actuelle ; la demande est mise hors délai (son état est égal à 'hors-délai') et le processus se termine ; (ii) le client annule sa demande ce qui met l'état de la Demande du client à 'annulée' et termine le processus ; (iii) la demande peut être satisfaite ce qui fait qu'on se retrouve dans le deuxième cas décrit ci-dessous.

2. il existe des possibilités, la Demande peut être satisfaite. L'activité 'Satisfaire Demande' permet de changer l'état de la Demande à 'satisfaite'.

La satisfaction de la demande permet de créer une réservation. Une instance de la classe réservation est créée et son état est mis à 'créée'. Il est ensuite nécessaire de gérer les disponibilités associées à la chambre. L'activité 'Gérer disponibilités' créée des instances de la classe Disponibilité. On note qu'il est possible de représenter ces trois activités au sein d'une activité plus abstraite correspondant à la réservation de chambre.

Une fois la réservation créée, il est éventuellement possible à l'opérateur de la modifier. Il doit alors prévenir le client. Cette activité est rarement réalisée, mais permet à l'opérateur de gérer des imprévus.

La réservation est ensuite facturée au client. L'activité 'Facturer' permet de changer l'état de la réservation à 'facturée'. Ceci termine le processus.

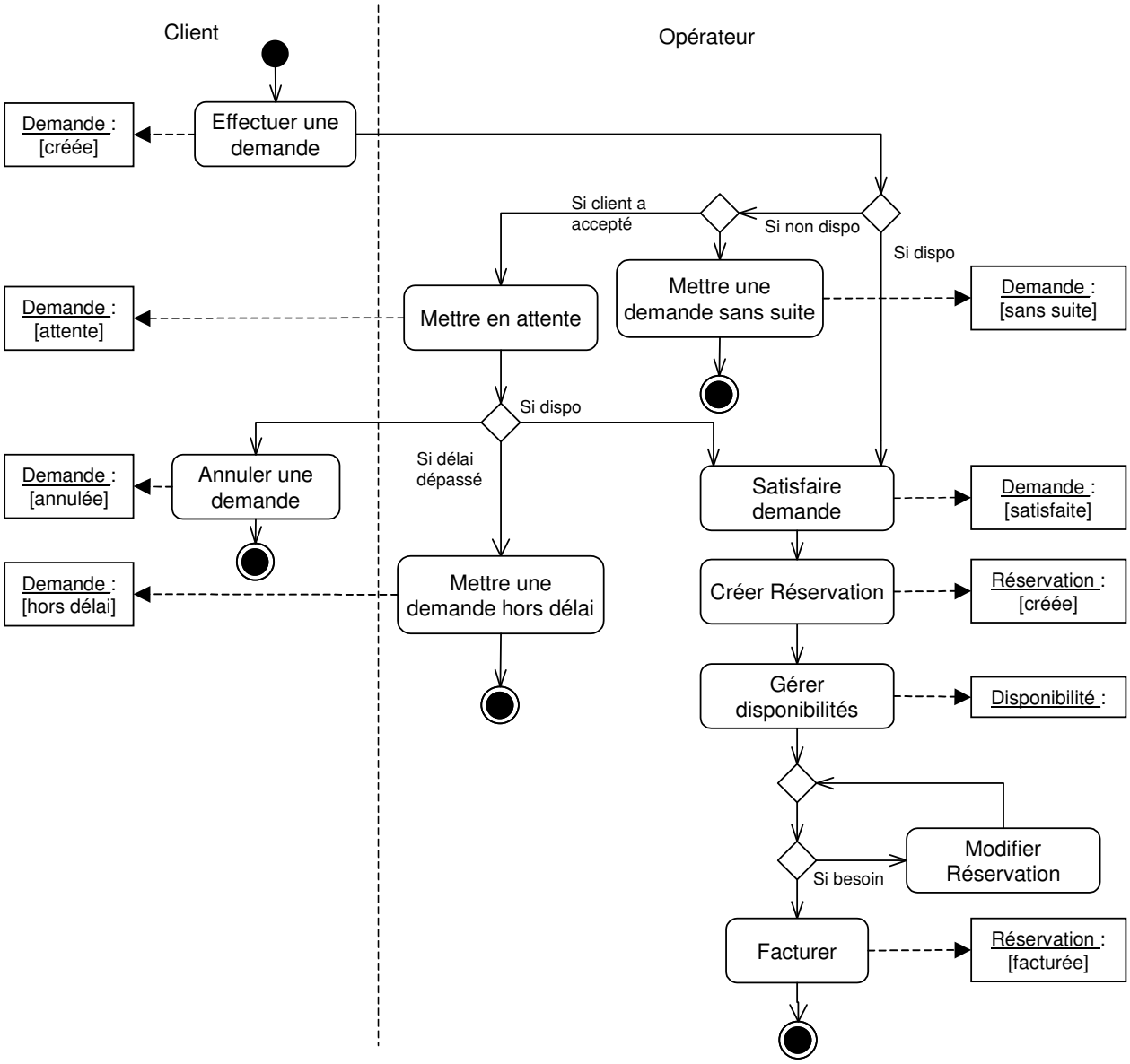


Figure 35 : Diagramme d'activités représentant le processus de gestion de réservations de chambres d'hôtel

Ce processus compte : 7 chemins, 2 acteurs, 10 activités, 3 classes (Demande, Réservation et Disponibilité), 10 états de classes.

Le but de ce processus peut se décrire en termes d'états terminaux d'objets :

$$B = (\text{Demande.état} = \text{'sans-suite'}) \vee (\text{Demande.état} = \text{'hors-délai'}) \vee$$
$$(\text{Demande.état} = \text{'annulée'}) \vee [(\text{Demande.état} = \text{'satisfaite'})$$
$$\wedge (\text{Réservation.état} = \text{'facturée'}) \wedge (\text{new.Disponibilité})$$

2.1.3 Récapitulatif des concepts manipulés

Cette section présente succinctement les différents concepts manipulés dans les processus, en particulier ceux qui interviennent dans le calcul des métriques.

2.1.3.1 Présentation générale des processus

Ce cas de gestion de réservation de chambres d'hôtel est composé de quatorze processus, permettant de gérer les réservations (création, paiement ou annulation) et de gérer les produits (création, modification et fermeture de station, d'hôtel ou de chambre). Chacun de ces processus hormis celui présenté à la Figure 35 sont détaillés en Annexe 1 (page 324).

A chaque processus correspond un but. Celui-ci n'est pas explicite en UML. Il se déduit des derniers états atteints par chacun des objets manipulés par le processus.

Le Tableau 9 présente les quatorze processus ainsi que leurs buts associés.

Processus	Buts
Réservation de chambre	Demande.état = 'sans-suite' ∨ Demande.état = 'hors-délai' ∨ Demande.état = 'annulée' ∨ [Demande.état = 'satisfaite' ∧ Réservation.état = 'facturée' ∧ new.Disponibilité.datefin]
Annulation de réservation	Réservation.état = 'annulée' ∧ new.Disponibilité.datefin ∧ Réservation.pénalité = 0
Paiement de réservation	Réservation.état = 'payée'
Modification d'un hôtel	new.hotel.adresse ∨ new.hotel.catégorie
Modification d'une station	new.station.catégorie ∨ new.station.géographie ∨ new.station.description>
Modification d'une chambre	new.chambre.prixch>
Fermeture définitive d'un hôtel	Hotel.état='fermé' ∧ Réservation.état = 'annulée' ∧ new.Disponibilité.datefin
Suppression de chambre	Chambre.état='fermée' ∧ [Réservation.état = 'annulée' ∨ Réservation.état = annulée'] ∧ new.Disponibilité.datefin
Création d'un hôtel	Hotel.état='ouvert' ∧ Chambre.état ='ouverte' ∧ new.Disponibilité.datefin
Fermeture temporaire d'un hôtel	Hotel.état='fermé_temp' ∧ Réservation.état = 'annulée' ∧ Hotel_Ferm_Temp.état = 'fermée ∧ new.Disponibilité.datefin ∧ new.Période.datefin
Fermeture temporaire d'une station	Station.état='fermée_temp' ∧ Réservation.état = 'annulée' ∧ Station_Ferm_Temp.état = 'fermée ∧ new.Disponibilité.datefin ∧ new.Période.datefin
Fermeture définitive d'une station	Station.état='fermée' ∧ Réservation.état = 'annulée' ∧ new.Disponibilité.datefin
Création d'une station	Station.état='ouverte' ∧ Hotel.état='ouvert' ∧ Chambre.état ='ouverte' ∧ new.Disponibilité.datefin
Création d'une chambre	Hotel.état='ouvert' ∨ Hotel.état='fermé_temp'∧ Chambre.état ='ouverte' ∧ new.Disponibilité.datefin

Tableau 9 : Présentation des processus et de leur but

Les processus mettent en jeu trois acteurs : l'opérateur, le client et l'hôtelier.

Chaque processus se décompose en activités. La section suivante présente l'ensemble des activités des différents processus.

2.1.3.2 Présentation générale des activités

Le processus décrit à la Figure 35 a permis de mettre en évidence 10 activités. Les treize autres processus permettent d'identifier 20 autres activités. Certaines activités peuvent bien entendu être exécutées dans le déroulement de plusieurs processus ; c'est, par exemple, le cas de l'activité 'Gérer disponibilités'.

Nous avons vu au chapitre précédent qu'à chaque activité est associée une transition d'états. Le processus décrit à la Figure 35 a permis de mettre en évidence les transitions d'états associées aux différentes activités de ce processus. En effet, le flux d'informations (indiqué par des flèches pointillées dans la représentation des processus) permet de préciser l'état dans lequel l'activité amène l'objet. Une analyse des différents processus permet de préciser, dans le Tableau 10, la transition d'état correspondant à chaque activité.

Activité	Transition d'état
Effectuer une demande	<-, demande.état='créée'>
Mettre en attente	<demande.état='créée', demande.état='attente'>
Mettre une demande hors délai	<demande.état='attente', demande.état='hors délai'>
Mettre une demande sans suite	<demande.état='créée', demande.état='sans suite'>
Satisfaire demande	<demande.état='créée', demande.état='satisfaite'> ∨ <demande.état='attente', demande.état='satisfaite'>
Créer réservation	<-, réservation.état='créée'>
Gérer les disponibilités	<old.disponibilité, new.disponibilité>
Modifier réservation	<old.réservation.hotel, new.réservation.hotel> ∨ <old.réservation.chambre, new.réservation.chambre>
Facturer	<réservation.état='créée', réservation.état='facturée'>
Annuler Réservation par le client	<réservation.état='créée', réservation.état='annulée'> ∨ <réservation.état='facturée', réservation.état='annulée'>
Facturer pénalité annulation	<old.réservation.penalité, new.réservation.pénalité>
Payer pénalité	<old.réservation.penalité, new.réservation.pénalité = 0>
Consommer	<réservation.état='facturée', réservation.état='consommée'> ∨ <réservation.état='facturée', réservation.état='partiel_conso'>
Facturer pénalité dû à une consommation partielle	<old.réservation.penalité, new.réservation.pénalité>
Payer Réservation	<réservation.état='consommée', réservation.état='payée'> ∨ <réservation.état='partiel_conso', réservation.état='payée'>
Modifier Hotel	<old.hotel.adresse, new.hotel.adresse> ∨ <old.hotel.catégorie, new.hotel.catégorie>
Modifier Station	<old.station.catégorie, new.station.catégorie> ∨ <old.station.géographie, new.station.géographie> ∨ <old.station.description, new.station.description>
Modifier Chambre	<old.chambre.prixch, new.chambre.prixch>
Fermer hotel définitivement	<hotel.état='ouvert', hotel.état='fermé'> ∨ <hotel.état='fermé_temp', hotel.état='fermé'>
Fermer station définitivement	<station.état='ouverte', station.état='fermée'> ∨ <station.état='fermée_temp', station.état='fermée'>
Annuler réservation	<réservation.état='créée', réservation.état='annulée'> ∨ <réservation.état='facturée', réservation.état='annulée'>
Supprimer chambre	<chambre.état='ouverte', chambre.état='fermée'>
Créer Hotel	<-, hotel.état='ouvert'>
Créer Chambre	<-, chambre.état='ouverte'>
Créer Fermeture temporaire station	<old.période, new.période>
Créer Fermeture temporaire Hotel	<old.période, new.période>
Fermer Station temporairement	<station.état='ouverte', station.état='fermée_temp'>
Créer nouvelles réservations par l'opérateur	<-, réservation.état='créée'>
Fermer Hotel Temporairement	<hotel.état='ouvert',hotel.état='fermé_temp'>
Créer station	<-, station.état='ouvert'>

Tableau 10 : Description des activités

2.1.3.3 Description des classes utilisées

Les quatorze processus manipulent les objets de sept classes : Demande, Chambre, Réservation, Disponibilité, Hôtel, Station, Période. Cette dernière permet de gérer les fermetures temporaires de station et d'hôtel. La Figure 36 présente le diagramme de classes constitué à partir des sept classes utilisées par les différents processus. Ce diagramme de classes présente les attributs des classes.

Les attributs sont importants pour le calcul des mesures. En effet, comme nous l'avons vu dans le chapitre précédent, le concept UML d'Attribut peut être mis en relation avec le concept BPRAM de propriété. Or, d'après la définition du lien *correspond*, une chose BPRAM *correspond* une chose SRAM si leurs propriétés sont les mêmes à des isomorphismes près (ces isomorphismes permettent aux propriétés d'avoir des domaines différents).

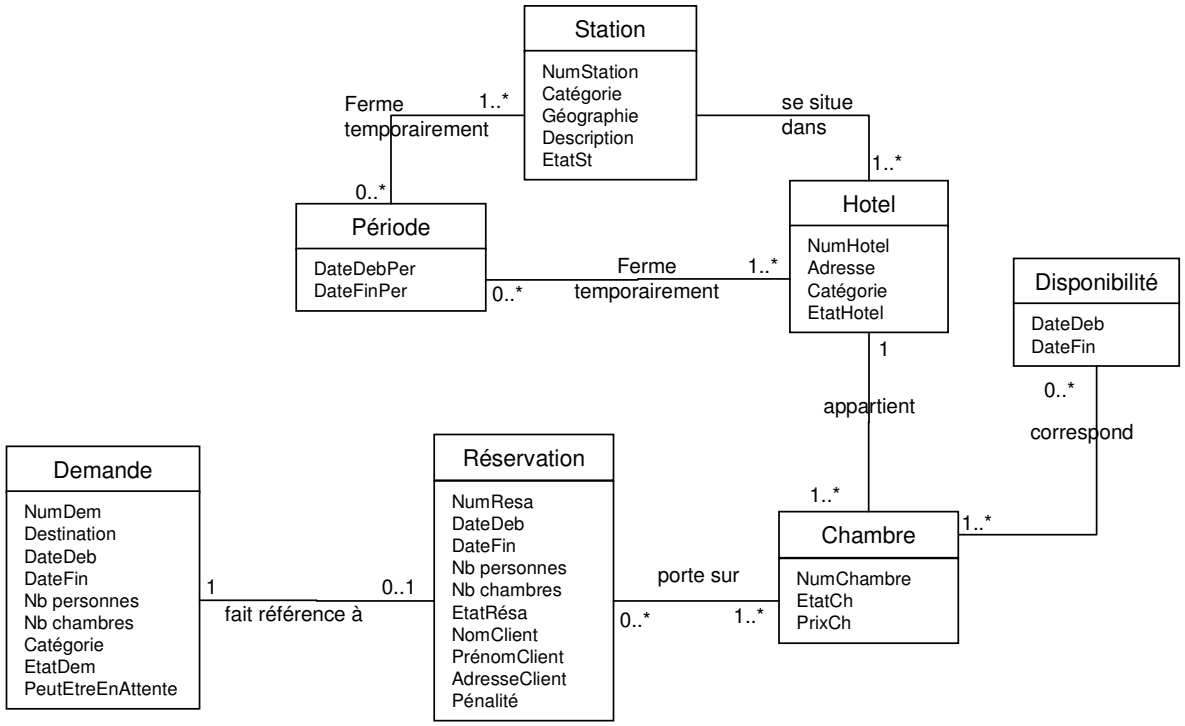


Figure 36 : Diagramme de classes associé aux processus

Chacune de ces classes possède un ensemble d'états remarquables qui sont mis en évidence dans la description des différents processus. Ces états sont rassemblés dans le Tableau 11. Chaque colonne correspond à une classe. Chaque ligne (hormis la première qui correspond au nom de la classe) fait référence à un état de cette classe. Ainsi, on compte 20 états différents, toute classe confondue.

Classes	Demande	Réservation	Chambre	Disponibilité	Hotel	Station	Période
Etats	créée attente annulée sans suite hors_délai satisfaite	créée facturée annulée consommée payée partiel_conso	fermée ouverte		ouvert fermé fermé_temp	ouverte fermée fermée_temp	

Tableau 11 : Etats des classes utilisées par les processus

Le système est modélisé en utilisant le méta-modèle O*. La section suivante présente le modèle O* du système dans le cadre de notre exemple.

2.2. Modèle O* du système de gestion de réservation de chambres hôtelières

Comme nous l'avons vu au chapitre précédent, le modèle O* est composé de deux parties, l'une statique et l'autre dynamique.

2.2.1 Description de la partie dynamique

La Figure 37 présente un fragment de la partie dynamique. Il s'agit du fragment qui permet de gérer les différentes activités décrites à la Figure 35. Dans ce schéma, les rectangles à bord droit correspondent aux classes objets. Ils ont été colorés pour une meilleure lisibilité. Les rectangles aux bouts arrondis correspondent aux classes des deux autres types (Acteurs et Horloge). Les classes Client et Opérateur sont des classes Acteurs. Le style des flèches n'a aucune signification particulière, mais permet de repérer la structure des transitions dynamiques (Objet, Evénement et Opération) associées aux différents événements. Les événements sont représentés par les triangles, les opérations par les rectangles sur les classes. L'exécution de certaines opérations étant soumise à conditions, celles-ci sont précisées par la lettre C suivi d'un chiffre. Il est également possible qu'une même opération ait lieu sur plusieurs instances de la même classe au même moment. On parle alors de facteurs. Ceux-ci sont représentés par une double flèche et leurs paramètres d'exécution sont spécifiés par la lettre F suivi d'un chiffre.

Ainsi, le schéma de la Figure 37 permet de voir :

- l'événement EV1 correspond à l'arrivée d'une nouvelle demande par un client ce qui déclenche l'opération 'créer' sur la classe objet Demande.
- l'événement EV3 correspond à la gestion des demandes. S'il existe des disponibilités permettant de satisfaire la demande du client, cet événement permet l'exécution des opérations 'réserver' sur la classe objet Hôtel, 'satisfaire' sur la classe objet Demande et 'créer' sur la classe objet Réservation. Dans le cas contraire, cet évenement permet de mettre la demande du client en attente (opération 'mettre en attente sur Demande') si le client a accepté (C3), ou de l'annuler (opération 'annuler' sur Demande).
- l'événement EV5 est un événement interne permettant de satisfaire les demandes en attente lorsque le système mesure un accroissement des disponibilités. Ainsi, les opérations 'satisfaire' sur Demande, 'réserver' sur Hôtel et 'créer' sur Réservation sont exécutées.
- l'événement EV6 est un événement temporel. Il se déclenche chaque jour et permet d'annuler (opération 'annuler' sur Demande) toutes les demandes arrivant à échéance à J+15 (F6).
- l'événement EV7 correspond à l'annulation d'une demande en attente sur l'initiative du client. Il déclenche l'opération 'annuler' sur la classe objet Demande uniquement s'il y a une demande en attente associée au client (C5).
- l'événement EV13 correspond à la facturation d'une Réservation qui vient d'être créée ; l'opération 'facturer' est déclenchée.

- l'événement EV14 correspond à la création d'une réservation sur l'initiative d'un opérateur. Les opérations 'créer' sur Réservation et 'réserver sur Hôtel sont alors exécutées.

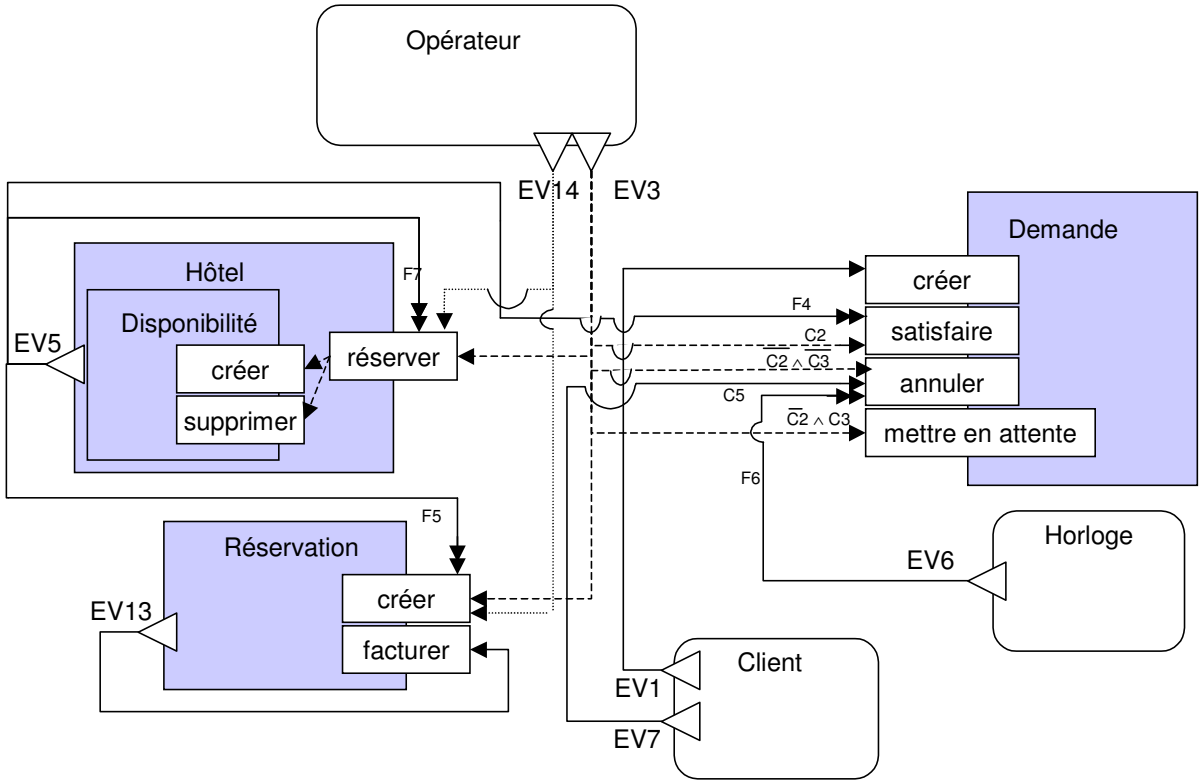


Figure 37 : Description dynamique correspondant à la gestion de réservation de chambres d'hôtel

La description du système comporte 19 événements déclenchant 17 opérations différentes. Certaines opérations comme 'créer' sur Réservation ou 'annuler' sur Demande peuvent être déclenchées par plusieurs événements différents. Le système est en relation avec trois acteurs, le Client, l'Opérateur et l'Hôtelier.

Chaque événement est décrit textuellement, en précisant :

- la classe qui déclenche l'événement ;
- le nom de l'événement ;
- le message si c'est un événement externe, le prédicat si l'événement est interne ou temporel ;
- les opérations que l'événement déclenche.

La Figure 38 présente un exemple de description d'événements déclenchés par l'acteur Opérateur. Il s'agit de l'événement EV3 : arrivée d'une nouvelle demande.

Acteur Client
Evénement
Arrivée d'une nouvelle Demande EV1
Message
Prénom : **string**
Nom : **string**
Date de naissance **date**
Adresse **string**
Téléphone : **string**
[Nom Station : **string**]
[Catégorie Hôtel : **string**]
Check in date : **date**
Check out date : **date**
Nombre de Chambres : **entier**
Déclenchement
Créer sur Demande

Figure 38 : Description de l'événement EV1

La description des autres événements est donnée en Annexe 1 (page 324) de cette thèse.

Associé à la partie dynamique, le modèle O* permet de représenter le système d'un point de vue statique. La section suivante présente cette partie pour le cas d'application.

2.2.2 Description de la partie statique

La partie statique a pour but de mettre en évidence les liens statiques (c'est-à-dire les liens d'héritage, de composition ou de référence) qui existent entre les différentes classes :

- Un lien d'héritage entre deux classes établit une relation de spécialisation - généralisation entre chaque objet de la première classe, dit objet spécialisé, et un objet de la deuxième classe, dit objet généralisé. Par extension, on parlera respectivement de classe spécialisée et de classe généralisée, la classe spécialisée héritant de la classe généralisée.
- Un lien de composition entre deux classes établit une relation de dépendance structurelle entre chaque objet de la première classe, dit objet composite, et un ou plusieurs objets de la deuxième classe, dits objets composants. Par extension on parlera respectivement de classe composite et de classe composante.
- Un lien de référence entre deux classes établit une relation de dépendance temporelle entre chaque objet de la première classe, dit objet référençant, et un ou plusieurs objets de la deuxième classe, dits objets référencés. Par extension on parlera respectivement de classe référençante et de classe référencée.

La Figure 39 présente le schéma de classes du cas de gestion de réservations hôtelières. Celui-ci ne contient que quatre classes objets. Il rassemble dans un même dessin les relations qui existent entre les différentes classes objets. Les liens de référence sont représentés par des flèches en pointillés. La source correspond à la classe référençante et la cible à la classe référencée. Les liens de composition sont dessinés avec des flèches pleines ayant pour source la classe composante et pour cible la classe composite. Les doubles flèches indiquent un lien

multiple entre les classes. Dans notre cas, il s'agit de préciser que la classe Hôtel est composée de plusieurs Disponibilités.

Ainsi, dans notre cas d'application : (i) une Réservation résulte d'une Demande et concerne un Hôtel et (ii) un Hôtel est composé d'un ensemble de DisponibilitéHotel.

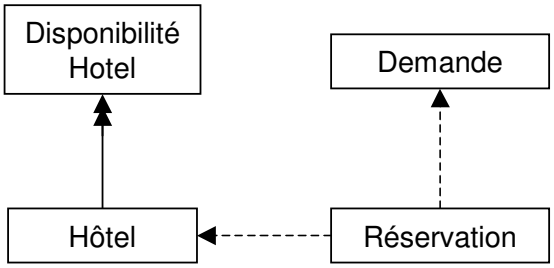


Figure 39 : Schéma de classes O*

La description de chacune des classes est structurée en quatre parties : les propriétés, les opérations, le graphe de transitions d'états et les assertions (c'est-à-dire des contraintes statiques).

La Figure 40 présente la description de la classe objet Demande.

Le graphe de transitions permet de visualiser les différents états caractéristiques de la classe ainsi que les transitions d'états qui permettent de passer d'un état à un autre. A chaque transition d'états correspond une opération, son nom est indiqué sur la flèche représentant la transition. Le sens de la flèche permet de préciser l'état initial et l'état final de la transition. Il arrive qu'une transition d'état boucle sur elle-même, dans ce cas, l'opération associée permet de modifier les propriétés de la classe.

Demande

Propriétés

Numéro : entier
DateDeb : date
DateFin : date
Nb chambres : entier
Catégorie : string
Station : string
PeutEtreEnAttente : booléen

Opérations

Créer
Satisfaire (la demande)
Annuler
Mettre attente

Graphe de transitions d'états

{créée, satisfaite, annulée, attente}

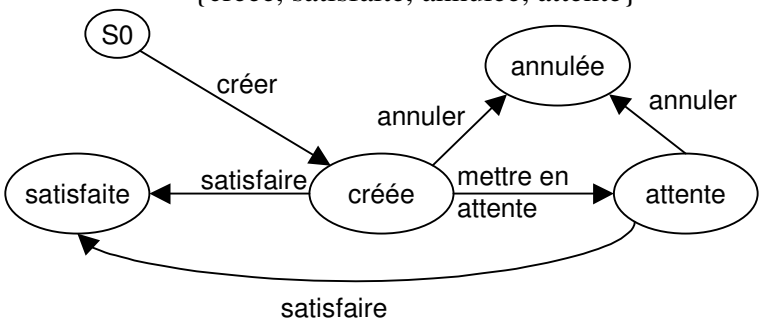


Figure 40 : Description de la classe objet Demande

Les trois autres classes sont décrites en Annexe.

Afin de posséder, dans ce chapitre, l'information nécessaire au calcul des différentes métriques, nous proposons deux tableaux récapitulatifs des descriptions des classes objets présentées en Annexe 1. Ainsi, le Tableau 12 liste les différents états des classes objets que le système utilise.

Classes	Demande	Réservation	DisponibilitéHotel	Hôtel
Etats	Créée Attente Annulée Satisfaite	Créée Facturée Annulée Consommée Payée Partiel_conso		Ouvert Fermé

Tableau 12 : Etats des classes objet utilisées par le système

Le Tableau 13 décrit pour chaque classe les opérations qui lui sont associées ainsi que les transitions d'états correspondantes.

Classe	Opérations	Transition d'état
Demande	Créer	<so, demande.état='créée'>
	Mettre en attente	<demande.etat='créée', demande.état='attente'>
	Annuler	<demande.état='attente', demande.état='annulée'> V <demande.état='créée', demande.état='annulée'>
	Satisfaire	<demande.état='créée', demande.état=satisfaite'> V <demande.état='en attente', demande.état='satisfaite'>
Réservation	Créer	<so, réservation.état='créée'>
	Facturer	<réservation.état='créée', réservation.état='facturée'>
	Annuler	<réservation.état='créée', réservation.état='anulée'> V <réservation.état='facturée', réservation.état='annulée'>
	Pénaliser	<old.réservation.pénalité, new.réservation.pénalité>
	Payer pénalité	<old.réservation.pénalité, new.réservation.pénalité=0>
	Consommer	<réservation.état='facturée', réservation.état='annulée'>
	Interrompre	<réservation.état='facturée', réservation.état='partiel_conso'>
	Payer	<réservation.état='consommée', réservation.état='payée'> V <réservation.état='partiel_conso', réservation.état='payée'>
Hôtel	Mettre à jour	<old.hotel.adresse, new.hotel.adresse> V <old.hotel.categorie, new.hotel.categorie>
	Fermer	<hotel.état='ouvert', hotel.état='fermé'>
	Décroître	<old.hotel.nbchambres, new.hotel.nbchambres>
	Créer	<so, hotel.état='ouvert'>
	Annuler	<old.hotel.disponibilité, new.hotel.disponibilité>
	Réserver	<old.hotel.disponibilité, new.hotel.disponibilité>
	Augmenter	<old.hotel.nbchambres, new.hotel.nbchambres>
DisponibilitéHotel	Créer	<disponibilité.état='supprimée', disponibilité.état='créée'>
	Supprimer	<disponibilité.état='créé'', disponibilité.état='supprimée'>

Tableau 13 : Description des différentes opérations

A partir de ces descriptions des processus et du système, nous proposons de calculer les différentes métriques associées au diagramme d'activité UML et à O* qui ont été définies au chapitre précédent.

3. Evaluation de l'alignement

Il est prévisible à la lecture de la description des processus et du système que les deux entités ne sont pas parfaitement alignées. Dans cette section, nous évaluons l'alignement en calculant les neuf métriques définies pour le diagramme d'activités UML et le méta-modèle O*.

Le reste de la section est organisé comme suit : chaque sous-section hormis la dernière correspond à un facteur. Pour chaque critère associé à ce facteur, nous calculons et analysons le résultat de la mesure obtenue. La dernière section analyse les différentes mesures en introduisant les notions de seuil et de poids.

Commençons par les critères associés au *facteur intentionnel*.

3.1. Facteur intentionnel

Le facteur intentionnel est composé de quatre critères, mais seuls trois d'entre eux, le *Taux de support*, la *Satisfaction des buts*, la *Présence des acteurs* sont décrits ici. La métrique associée au critère *Présence des ressources* ne peut être calculée.

3.1.1 Taux de support

Au niveau spécifique, la métrique associée au critère *Taux de support* mesure la proportion d'activités UML *représentées* par des transitions d'états O* comparée au nombre total d'activités. Cette métrique suivant sa valeur permet de donner une première image de l'alignement qui existe entre des processus d'entreprise et le système.

3.1.1.1 Calcul de la mesure

Afin d'appliquer cette métrique, il est nécessaire d'identifier pour chaque activité UML si elle est représentée par une transition d'états O*. Le Tableau 14 présente pour chaque activité UML la transition d'état qui la *représente*. Si aucune transition d'état n'est précisée, c'est que le système ne prend pas en charge l'activité.

Activités UML	Transition d'état
Effectuer une demande	<so, demande.état='créée'>
Mettre en attente	<demande.etat='créée', demande.état='attente'>
Mettre une demande hors délai	<demande.état='attente', demande.état='annulée'> V <demande.état='créée', demande.état='annulée'>
Mettre une demande sans suite	<demande.état='attente', demande.état='annulée'> V <demande.état='créée', demande.état='annulée'>
Annuler demande	<demande.état='attente', demande.état='annulée'> V <demande.état='créée', demande.état='annulée'>
Satisfaire demande	<demande.état='créée', demande.état=satisfaite'> V <demande.état='en attente', demande.état='satisfaite'>
Gérer les disponibilités	<old.hotel.disponibilité, new.hotel.disponibilité>
Créer réservation	<so, réservation.état='créée'>
Modifier réservation	
Facturer	<réservation.état='créée', réservation.état='facturée'>
Annuler Réservation par le client	<réservation.état='créée', réservation.état='anulée'> V <réservation.état='facturée', réservation.état='annulée'>
Facturer pénalité annulation	<old.réservation.pénalité, new.réservation.pénalité>
Payer pénalité	<old.réservation.pénalité, new.réservation.pénalité=0>
Consommer	<réservation.état='facturée', réservation.état='annulée'> V <réservation.état='facturée', réservation.état='partiel_conso'>
Facturer pénalité due à une consommation partielle	<old.réservation.pénalité, new.réservation.pénalité>
Payer réservation	<réservation.état='consommée', réservation.état='payée'> V <réservation.état='partiel_conso', réservation.état='payée'>
Modifier Hôtel	<old.hotel.adresse, new.hotel.adresse> V <old.hotel.categorie, new.hotel.categorie>
Modifier Station	
Modifier Chambre	
Fermer hôtel définitivement	<hotel.état='ouvert', hotel.état='fermé'>
Annuler Réservation	<réservation.état='créée', réservation.état='anulée'> V <réservation.état='facturée', réservation.état='annulée'>
Supprimer chambre	<old.hotel.nbchambres, new.hotel.nbchambres>
Créer Hôtel	<so, hotel.état='ouvert'>
Créer Chambre	<old.hotel.nbchambres, new.hotel.nbchambres>
Créer Fermeture temporaire station	
Créer Fermeture temporaire hôtel	
Fermer station temporairement	
Créer nouvelles réservations par l'opérateur	<so, réservation.état='créée'>
Fermer hôtel temporairement	
Créer station	

Tableau 14 : Liste des activités UML *représentées* par une transition d'état O*

On compte 22 activités UML représentées chacune par une transition d'état O* pour 31 activités au total. Il est donc possible d'en déduire :

$$Ts\ (M_{proc}, M_{syst}) = 22/31 \approx 0{,}71$$

Où M_{proc} correspond aux diagrammes d'activités UML et M_{syst} au modèle O*.

3.1.1.2 Analyse de la mesure

Une valeur du *Taux de support* égale à 0,71 signifie que près de trente pour cent des activités ne sont pas prises en charge par le système. Ce chiffre est une première explication aux difficultés rencontrées par les utilisateurs du système. Il permet de donner un premier aperçu de la réalité de l'alignement entre les processus et le système. En effet, on ne se rend pas toujours bien compte qu'une activité n'est pas gérée par le système, que l'utilisateur est en quelque sorte obligé de forcer le système pour arriver à ses fins.

Dans le cas de gestion de réservations de chambres d'hôtel, il existe, par exemple, une activité qui consiste à modifier une réservation en cours. Cette activité n'est pas supportée par le système, l'hôtelier ou l'opérateur doit supprimer la réservation et en recréer une autre avec les nouvelles caractéristiques. Les deux façons de procéder ne sont pas exactement équivalentes car dans le deuxième cas il peut y avoir perte d'information au moment de la suppression. Mais surtout, la réservation du client a changé de numéro, il est donc nécessaire de l'en informer afin qu'il ne rencontre pas de problème au moment de récupérer les clés de sa chambre.

Autre exemple, la modification des propriétés ou de l'état des chambres n'est pas possible dans le système, la chambre n'est pas une classe objet à part entière, seul le nombre de chambres est présent sous la forme de propriété de la classe objet Hôtel.

3.1.2 Satisfaction des buts

La métrique générique *Satisfaction des buts* compare le nombre de buts pouvant être satisfaits par le système et le nombre global de buts dans les différents processus d'entreprises. Le concept de but tel qu'il est défini dans le méta-modèle BPRAM correspond à un ensemble d'états que l'on cherche à atteindre dans le diagramme d'activités UML. Ainsi, un but est satisfait par le système si chaque état constituant le but *correspond* à un état d'une classe objet dans le méta-modèle O*. Cette deuxième métrique permet de montrer une autre facette de l'alignement entre les processus et le système.

3.1.2.1 Calcul de la mesure

Nous avons étudié pour chacun des buts des quatorze processus s'il est pris en charge par le système. Le Tableau 15 montre le résultat de notre étude.

N°	Buts	Description	Satisfait
1	Demande.état = 'sans-suite' ∨ Demande.état = 'hors-délai' ∨ Demande.état = 'annulée' ∨ [Demande.état = 'satisfaite' ∧ Réservation.état = 'facturée' ∧ new.Disponibilité.datefin]	Traiter une demande complètement (de la création de la demande à la facturation de la réservation)	Non
2	Réservation.état = 'annulée' ∧ new.Disponibilité.datefin ∧ Réservation.pénalité = 0	Annuler une réservation	Non
3	Réservation.état = 'payée'	Payer une réservation	Oui
4	new.hotel.adresse ∨ new.hotel.catégorie	Mettre à jour les caractéristiques d'un hôtel	Non
5	new.station.catégorie ∨ new.station.géographie ∨ new.station.description>	Mettre à jour les caractéristiques d'une station	Non
6	new.chambre.prixch>	Mettre à jour le prix d'une chambre	Non
7	Hotel.état='fermé' ∧ Réservation.état = 'annulée' ∧ new.Disponibilité.datefin	Fermer définitivement un hôtel	Non
8	Chambre.état='fermée' ∧ [Réservation.état = 'annulée' ∨ Réservation.état = annulée'] ∧ new.Disponibilité.datefin	Fermer une chambre	Non
9	Hotel.état='ouvert' ∧ Chambre.état ='ouverte' ∧ new.Disponibilité.datefin	Ouvrir un hôtel	Non
10	Hotel.état='fermé_temp' ∧ Réservation.état = 'annulée' ∧ Hotel_Ferm_Temp.état = 'fermée ∧ new.Disponibilité.datefin ∧ new.Période.datefin	Fermer temporairement un hôtel	Non
11	Station.état='fermée_temp' ∧ Réservation.état = 'annulée' ∧ Station_Ferm_Temp.état = 'fermée ∧ new.Disponibilité.datefin ∧ new.Période.datefin	Fermer temporairement une station	Non
12	Station.état='fermée' ∧ Réservation.état = 'annulée' ∧ new.Disponibilité.datefin	Fermer définitivement une chambre	Non
13	Station.état='ouverte' ∧ Hotel.état='ouvert' ∧ Chambre.état ='ouverte' ∧ new.Disponibilité.datefin	Ouvrir une station	Non
14	Hotel.état='ouvert' ∨ Hotel.état='fermé_temp'∧ Chambre.état ='ouverte' ∧ new.Disponibilité.datefin	Créer une chambre	Non

Tableau 15 : Analyse de la prise en charge par le système des buts du processus

Seul un but sur 14 est pris en charge par le système.

$$Sb\ (M_{proc}, M_{syst}) = 1/14 \approx 0,07$$

3.1.2.2 Analyse de la mesure

La *Satisfaction des buts* vaut 1/14, ce qui signifie que seul 7% des buts que les processus permettent d'atteindre sont pris en charge par le système. Sont considérés comme pris en charge par le système uniquement les buts pour lesquels chaque partie peut être satisfaite par le système. Or, plusieurs sous-buts ne sont que partiellement gérés.

Comme nous le verrons par la suite, plusieurs raisons peuvent expliquer cette faible valeur :

- La gestion des disponibilités se fait par chambre dans les processus et globalement, au niveau de l'hôtel, dans le système. Ainsi, tous les buts des processus incluant la gestion des disponibilités ne sont pas pris en charge par le système.
- Le système ne prend pas en charge le concept de chambre, il gère uniquement le nombre de chambres des hôtels. Les buts 6, 8 et 14 correspondant respectivement à la mise à jour des prix de chambre, la fermeture et l'ouverture d'une chambre ne sont pas satisfaits par le système.
- De façon analogue, les buts 11, 12 et 13 permettant de gérer les stations ne sont pas satisfaits par le système. En effet, dans ce dernier, contrairement au processus, la station n'est pas un concept à part entière, mais une propriété de l'hôtel.

3.1.3 Présence des acteurs

Au niveau générique, la métrique *Présence des acteurs* calcule le taux d'acteurs des processus d'entreprise présents dans le système aux acteurs des processus. Le concept d'acteur existe dans le diagramme d'activités UML. Les classes acteurs en O* correspondent aux agents à l'interface du système et de son environnement qui déclenche un événement externe. Un acteur UML est présent dans le système s'il *correspond* à une classe acteur O*. Cette métrique permet de trouver une éventuelle explication à un faible *Taux de support*.

3.1.3.1 Calcul de la mesure

La description des processus nous a permis de mettre en évidence trois acteurs, le Client, l'Opérateur et l'Hôtelier. Dans le modèle O*, trois acteurs ont également été identifiés, le Client, l'Opérateur et l'Hôtelier. Chacun des acteurs des processus *correspond* à un acteur dans le système. On en déduit que

$$Pa(M_{proc}, M_{syst}) = 3/3 = 1$$

3.1.3.2 Analyse de la mesure

Dans notre cas d'étude, les trois acteurs des processus sont présents dans le système. Cela ne signifie pas forcément que tous les événements qu'ils déclenchent au niveau des processus sont représentés dans le système. En revanche, si l'un des acteurs des processus ne *correspond* à aucun acteur du système cela signifie que certains événements externes ne sont pas déclenchés de la même façon dans le système et dans les processus. L'activité causée par cet événement risque alors d'être mal gérée par le système. Ainsi, si le client n'existe pas en tant qu'acteur au niveau du système, l'activité de demande de réservation risque de ne pas être prise en charge convenablement par le système.

3.2. Facteur informationnel

Les deux sections suivantes présentent le calcul des métriques associées aux critères *Complétude de l'information* et *Exactitude de l'information*.

3.2.1 Complétude de l'information

La *Complétude de l'information* au niveau générique permet de mesurer la proportion de choses du modèle de processus d'entreprise qui *correspondent* chacune à une chose du modèle du système. Les concepts de chose des méta-modèles BPRAM et SRAM sont respectivement reliés aux classes UML et aux classes objet O*. La métrique spécifique associée au critère *Complétude de l'information* mesure donc la proportion de classes UML supportées par le système, c'est-à-dire tel qu'il existe une classe objet O* qui lui *corresponde*. Cette métrique donne une vision globale de la façon dont l'information manipulée dans les processus d'entreprise est gérée dans le système. Une faible valeur pour cette métrique a des répercussions sur de nombreuses métriques.

3.2.1.1 Calcul de la mesure

Pour appliquer la métrique associée au critère *Complétude de l'information*, il est nécessaire d'étudier pour chaque classe UML si elle *correspond* à une classe objet O*. Rappelons qu'une classe UML *correspond* à une classe objet O* si tous les attributs de la première sont égaux à un isomorphisme près aux propriétés de la seconde.

Le Tableau 16 précise pour chaque classe UML si elle *correspond* à une classe objet O*. La construction de ce tableau s'appuie sur la description des classes objets O* présentée en Annexe 1 ainsi que sur le diagramme de classes de la Figure 36.

Classe UML	Présente dans le système
Demande	Oui
Réservation	Oui
Chambre	Non
Disponibilité	Non
Hôtel	Non
Station	Non
Période	Non

Tableau 16 : Etude de la *correspondance* entre les classes UML et les classes objets O*

A partir de cette étude, on peut affirmer que :

$$Ci\ (M_{proc}, M_{syst}) = 2/7 \approx 0,29$$

3.2.1.2 Analyse de la mesure

Seules 2 classes UML sur 7 *correspondent* à une classe O*. En particulier, les concepts de Chambre et de Station ne sont pas des classes en O* mais juste des propriétés de la classe Hôtel. La classe Hôtel des processus ne *correspond* donc pas à la classe Hôtel du modèle du système car leurs propriétés sont différentes.

Nous avons vu que l'absence de la classe Chambre dans le système a des répercussions sur le *Taux de support* et la *Satisfaction des buts*. Nous verrons par la suite qu'une faible valeur de la métrique *Complétude de l'information* a des incidences sur pratiquement toutes les métriques. C'est pourquoi, dans un processus de correction de l'alignement, nous proposons de commencer par identifier des actions correctives permettant d'augmenter la valeur de la métrique associée au critère *Complétude de l'information*. Agir pour augmenter cette valeur, revient à augmenter la quantité d'informations manipulées par les processus qui est gérée par le système. Une bonne gestion de l'information par le système est la base d'un bon alignement et permet ensuite de pouvoir agir sur d'autres métriques.

3.2.2 Exactitude de l'information

Si l'information est gérée par le système cela ne signifie pas qu'elle est bien gérée. La métrique associée au critère *Exactitude de l'information* permet d'évaluer la proportion d'informations bien gérées par le système. Au niveau générique, cette métrique met en jeu les concepts BPRAM et SRAM d'états. Ces deux concepts sont respectivement spécialisés par les concepts d'état UML et d'état O*. Au niveau spécifique, la métrique d'*Exactitude de*

l'information permet de comparer le nombre d'états de classes intervenant dans des activités UML qui *correspondent* à un état O* et le nombre total d'états de ces classes UML.

3.2.2.1 Calcul de la mesure

Le calcul de cette mesure repose sur la *correspondance* des états des classes UML et des états des classes objets O*. Rappelons que deux états se *correspondent* (i) si les classes auxquelles ils sont associés se *correspondent* et (ii) si les valeurs caractéristiques des états sont les mêmes.

Ainsi, pour calculer cette métrique, il est nécessaire d'étudier en détail les états des classes UML Demande et Réservation. En effet, les autres classes UML ne *correspondant* à aucune classe objet O*, leurs états ne peuvent *correspondre* à un état du système. Le Tableau 17 présente le résultat de cette étude.

Classes UML	Etat	Présent
Demande	Créée	Oui
	Attente	Oui
	Annulée	Oui
	Sans suite	Non
	Hors_délai	Non
	Satisfaite	Oui
Réservation	Créée	Oui
	Facturée	Oui
	Annulée	Oui
	Consommée	Oui
	Payée	Oui
	Partiel_conso	Oui

Classes UML	Etat	Présent
Chambre	Fermée	Non
	Ouverte	Non
Hôtel	Ouvert	Non
	Fermée	Non
	fermé_temp	Non
Station	Ouverte	Non
	Fermée	Non
	fermée_temp	Non

Tableau 17 : Etude de *correspondance* entre états

A partir de ce tableau, on peut en déduire :

$$Ei\ (M_{proc}, M_{syst}) = 10/20 = 0,5$$

3.2.2.2 Analyse de la mesure

Il est clair que dans la mesure où seule deux classes UML sur sept *correspondent* à des classes O*, la valeur de la mesure associée au critère *Exactitude de l'information* vaut 0,5 ce qui signifie que 50% des états des classes UML *correspondent* à des états de classe objet O*. Cette valeur trouve son origine principale dans l'absence de certaines classes UML dans le système, mais d'autres raisons existent pour expliquer cette faible valeur.

Si la classe UML Chambre n'est pas supportée dans le système, les états 'créée', 'réservée' et 'fermée' ne sont pas, eux non plus, supportés.

En revanche, ce n'est pas parce qu'une classe UML *correspond* à une classe objet O* que tous ses états existent dans le système. Par exemple, l'état 'hors délai' de la Demande correspond à l'état dans lequel se trouve une demande en attente qui n'a pu être satisfaite quinze jours avant la date de début de réservation du client. Cet état n'existe pas dans le système. Celui-ci ne différencie pas la façon dont la demande est annulée : sur l'initiative du client, par dépassement du délai de mise en attente ou tout simplement car il n'y a pas de disponibilités et que le client refuse de mettre sa demande en attente.

Cette faible valeur pour la métrique associée au critère d'*Exactitude de l'information* explique en grande partie pourquoi peu de buts des processus sont pris en charge par le système. En effet, compte tenu de la définition des buts, la *correspondance* entre états est indispensable à la *correspondance* entre but.

Agir sur la métrique *Exactitude de l'information* est la deuxième chose à prendre en compte pour rétablir l'alignement après avoir travaillé sur la métrique associée au critère *Complétude de l'information*, c'est-à-dire dans notre cas sur la *correspondance* entre classe UML et classe objet O*. En effet, compte tenu de la définition des événements, des processus, des buts et des transformations, le concept d'état joue un rôle prépondérant dans la définition de nos métriques. Il est donc important que ceux-ci soient bien gérés par le système et que la métrique d'*Exactitude de l'information* ait une valeur élevée.

3.3. Facteur fonctionnel

Les deux métriques du facteur fonctionnel permettent de s'intéresser aux détails des différentes activités. Ils doivent être calculés pour chacune d'entre elles. Nous présentons en Annexe 1 le détail du calcul pour chacune des activités. Nous illustrons ici quatre d'entre elles.

3.3.1 Complétude de l'activité

Au niveau générique, la *Complétude de l'activité* se focalise sur une activité donnée et fournit de l'information sur les choses. Le concept d'activité du méta-modèle BPRAM est mis en relation avec celui d'activité du diagramme d'activités UML. Ainsi, au niveau spécifique, ce critère permet d'étudier en détail une activité donnée en calculant le nombre de classes UML intervenant dans cette activité *correspondant* chacune à une classe objet O* par comparaison au nombre total de classes intervenant dans l'activité.

3.3.1.1 Calcul de la mesure

Cette section applique la métrique *Complétude de l'activité* pour quatre activités : (i) 'créer chambre', (ii) 'mettre une demande sans suite', (iii) 'fermer station définitivement' et (iv) 'modifier réservation'. Les deux premières sont gérées par le système, mais pas les deux dernières.

L'activité 'créer chambre' utilise deux classes, Chambre et Hôtel et laisse la classe Chambre dans l'état 'créée'. Nous avons vu précédemment que la classe Hôtel ne *correspond* à aucune classe O*. Il en est de même pour la classe Chambre. La *Complétude de l'activité* est donc égale à 0 pour cette activité.

L'activité 'mettre une demande sans suite' permet de changer l'état d'une demande de 'créée' à 'sans suite'. Ainsi, seule la classe Demande est utilisée. Celle-ci *correspond* à la classe Demande O*. La *Complétude de l'activité* vaut donc 1 pour cette activité.

L'activité 'fermer station définitivement' est associée à la transition d'états : <station.état = 'ouverte', station.état = 'fermée'> ∨ <station.état = 'fermée_temp', station.état = 'fermée'>.

Seule la classe Station est utilisée dans cette activité. Elle ne *correspond* à aucune classe objet O*. La *Complétude de l'activité* est donc, ici, égale à 0.

L'activité 'modifier réservation' permet à l'opérateur de modifier une réservation en cas de nécessité, c'est-à-dire de changer l'hôtel ou la chambre réservé. Si l'hôtel est modifié, le client doit en être informé. Cette activité utilise trois classes UML, la Réservation, la Chambre et l'Hôtel. La *Complétude de l'activité* est donc égale à 0,33 pour cette activité.

3.3.1.2 Analyse de la mesure

Dans les deux premiers cas, l'activité est gérée par le système. La métrique permet d'évaluer si l'activité est bien gérée en analysant la façon dont l'information utilisée par les processus est représentée dans le système. Concernant les classes manipulées, l'activité 'mettre une demande sans suite' est gérée de la même façon dans les processus et dans le système. Ce n'est pas le cas pour l'activité 'créer chambre'.

Dans le cas où l'activité ne serait pas supportée par le système, mesurer l'alignement sur le critère *Complétude de l'activité* permet d'évaluer l'effort à mettre en œuvre pour que cette activité soit prise en charge convenablement. En effet, peuvent intervenir, dans cette activité, des classes UML qui *correspondent* à des classes objets O*. Ainsi, l'activité 'modifier réservation' n'est pas gérée par le système. Pourtant, une partie de l'information nécessaire à cette activité au niveau des processus est présente dans le système. En revanche, l'information associée à l'activité 'fermer définitivement station' n'est pas présente dans le système. Ceci peut-être une explication au fait que l'activité n'est pas prise en charge par le système.

3.3.2 Exactitude de l'activité

Au niveau générique, l'*Exactitude de l'activité* se focalise sur les états d'une activité donnée. Au niveau spécifique, ce critère permet d'évaluer pour une activité donnée le taux d'états des classes intervenant dans cette activité qui individuellement *correspondent* à un état d'une classe objet O*.

3.3.2.1 Calcul de la mesure

On s'intéresse aux mêmes activités que pour la description de la *Complétude d'activité*, à savoir : (i) 'créer chambre', (ii) 'mettre une demande sans suite', (iii) 'fermer station définitivement' et (iv) 'modifier réservation'.

Pour les activités 'créer chambre' et 'fermer station définitivement' dont les classes UML manipulées n'existent pas dans le système, les états utilisés ne peuvent *correspondre* à des états de classe O*. L'*Exactitude de l'information* vaut donc 0.

Pour les deux autres activités, l'analyse est différente. L'activité 'mettre une demande sans suite' requiert les états 'créée' et 'sans suite' de la classe Demande. Or parmi ces deux états, seul le premier *correspond* à un état d'une classe O*. *L'Exactitude de l'information* est donc égale à 0,5 pour cette activité.

Concernant l'activité 'modifier réservation', il s'agit de modifier le lien entre Réservation et Chambre ou entre Réservation et Hôtel. Aucun des états intervenant dans l'activité ne

correspond à un état du système. Pour cette activité, l'*Exactitude de l'information* vaut donc 0.

3.3.2.2 Analyse de la mesure

Ces quatre exemples montrent que :

- une activité UML peut être représentée par une transition d'états O* sans que les classes UML ou les états des classes correspondent individuellement respectivement à une classe ou à un état.
- une activité gérée par le système, peut avoir sa mesure de Complétude d'activité égale à 1 et celle de l'Exactitude de l'information différente de 1. Dans ce cas, l'activité est mal gérée.
- il existe un lien fort entre les deux métriques du facteur fonctionnel.

Ce critère permet aussi de trouver les causes d'une mauvaise ou de l'absence de prise en charge d'une activité. Il aide ainsi à mettre en place des actions correctives en fournissant une information détaillée pour chaque activité.

Ces exemples montrent qu'évaluer l'alignement entre les processus et le système uniquement en évaluant la proportion des activités supportées par le système est très réducteur. Nos métriques proposent un éclairage plus complet du lien entre ces deux entités.

3.4. Facteur dynamique

Les deux critères associés au facteur dynamique permettent de s'intéresser à la dynamicité de processus. Ils apportent un éclairage complémentaire aux autres critères en se focalisant sur le lien qui existe entre les activités.

3.4.1 Fiabilité du système

La métrique générique de *Fiabilité du système* compare le nombre de transformations de processus qui sont implémentées dans le système au nombre total de ces transformations. Une transformation BPRAM est implémentée dans le système si chacun des états intervenant dans cette transformation *correspond* à un état d'une chose dans le système et si les transformations entre ces états de choses du modèle de processus sont possibles entre les états des choses du système. Les concepts d'états dans les méta-modèles BPRAM et SRAM sont mis en relation avec les concepts d'états en UML et en O*. Le concept BPRAM de transformation est quant à lui mis en relation avec celui de transition d'états UML. Cette métrique permet d'évaluer la proportion de transitions d'états UML implémentée dans le système.

3.4.1.1 Calcul de la mesure

Le calcul de la métrique *Fiabilité du système* repose donc sur l'analyse des transitions d'états. Pour chacune d'entres elles, nous étudions si elle est implémentée dans le système. Les transitions d'états sont considérées comme implémentées dans le système si les états qui les

composent *correspondent* chacun à des états du système et si la transition d'états entre les états des classes O* est possible. Le Tableau 18 résume notre étude.

Transition d'état	Implémentée
<-, demande.état='créée'>	oui
<demande.état='créée', demande.état='attente'>	oui
<demande.état='attente', demande.état='hors délai'>	non
<demande.état='créée', demande.état='sans suite'>	non
<demande.état='créée', demande.état='satisfaite'>	oui
<demande.état='attente', demande.état='satisfaite'>	oui
<-, réservation.état='créée'>	oui
<old.disponibilité, new.disponibilité>	non
<réservation.état='créée', réservation.état='créée'>	oui
<réservation.état='créée', réservation.état='facturée'>	oui
<réservation.état='créée', réservation.état='annulée'>	oui
<réservation.état='facturée', réservation.état='annulée'>	oui
<old.réservation.hotel, new.réservation.hotel>	non
<old.réservation.chambre, new.réservation.chambre>	non
<old.réservation.penalité, new.réservation.pénalité>	oui
<old.réservation.penalité, new.réservation.pénalité = 0>	oui
<réservation.état='facturée', réservation.état='partiel_conso'>	oui
<réservation.état='facturée', réservation.état='consommée'>	oui
<old.réservation.penalité, new.réservation.pénalité>	oui
<réservation.état='consommée', réservation.état='payée'>	oui
<réservation.état='partiel_conso', réservation.état='payée'>	oui
<old.hotel.adresse, new.hotel.adresse>	oui
<old.hotel.catégorie, new.hotel.catégorie>	oui
<old.station.catégorie, new.station.catégorie>	non
<old.station.description, new.station.description>	non
<old.station.géographie, new.station.géographie>	non
<old.chambre.prixch, new.chambre.prixch>	non
<hotel.état='ouvert', hotel.état='fermé'>	oui
<hotel.état='fermé_temp', hotel.état='fermé'>	non
<station.état='ouverte', station.état='fermée'>	non
<station.état='fermée_temp', station.état='fermée'>	non
<réservation.état='créée', réservation.état='annulée'>	oui
<réservation.état='facturée', réservation.état='annulée'>	oui
<chambre.état='ouverte', chambre.état='fermée'>	non
<-, hotel.état='ouvert'>	oui
<-, chambre.état='ouverte'>	non
<old.période, new.période>	non
<-, Hotel_Fermeture_temp.état='créée'>	non
<station.état='ouverte', station.état='fermée_temp'>	non
<hotel.état='ouvert',hotel.état='fermé_temp'>	non
<-, station.état='ouvert'>	non

Tableau 18 : Liste des transitions d'états

A partir de ce tableau, nous pouvons calculer la métrique associée au critère *Fiabilité du système* :

$$Fs\ (M_{proc}, M_{syst}) = 22/41 \approx 0{,}53$$

3.4.1.2 Analyse de la mesure

Environ cinquante-deux pour cent des transitions d'états UML sont représentées dans le modèle O*. Ce chiffre s'explique ici par la faible proportion d'états de classes UML qui

correspondent à des états de classes objet O*. Il se pourrait que, malgré la présence dans le système d'états *correspondant* à des états de classe UML la transition ne soit pas possible au niveau du système. Ceci signifie par exemple qu'une règle de gestion n'est pas convenablement implémentée dans le système. Un tel phénomène n'a pas lieu dans notre exemple.

3.4.2 Réalisme dynamique

Le propos du *Réalisme dynamique* est de comparer le nombre de chemins présents dans le système au nombre total de chemins. Le concept de chemin tel que défini dans le méta-modèle BPRAM correspond au chemin du diagramme d'activités UML. Un chemin est présent dans le système si chaque état qui le constitue *correspond* à un état de classe objet en O* et si la succession des états du système est possible.

3.4.2.1 Calcul de la mesure

Le calcul de cette métrique repose sur l'étude des chemins. Parmi les quatorze processus, on compte 28 chemins. Pour chacun d'entre eux, il est nécessaire d'étudier si chacun des états qui le composent *correspond* à un état d'une classe objet O* et si la succession des transitions d'états est possible dans le modèle O*. Cette étude nous a permis d'identifier que seuls 5 chemins sont entièrement implémentés dans le système. Ainsi, on peut en déduire que :

$$Rd\ (M_{proc}, M_{syst}) = 5/28 \approx 0,18$$

3.4.2.2 Analyse de la mesure

Cette métrique s'intéresse au déroulement d'un processus dans sa globalité. Elle permet de prendre en compte les différentes alternatives modélisées pour représenter la diversité des situations rencontrées. Dans notre exemple, à peine plus de 18% des chemins de processus sont implémentés dans le système. Cela signifie que pour 82% des chemins, les utilisateurs sont amenés au moins une fois à 'forcer' le système ou à exécuter certaines parties des processus manuellement. Par exemple, le système ne fait pas la distinction entre fermeture temporaire d'un hôtel et fermeture définitive. Les hôteliers sont donc obligés de fermer définitivement un hôtel, qui n'apparaît donc plus pour le client dans le catalogue alors que la fermeture peut correspondre simplement aux congés annuels de l'hôtelier. Cependant le résultat n'est pas le même puisque l'hôtel disparaît complètement du catalogue.
Cette faible valeur repose en grande partie sur la façon dont l'information est gérée dans les processus et dans le système. Elle pourrait être accentuée par le fait que certaines transitions d'états ne sont pas implémentées dans le système alors que les deux états qui la composent *correspondent* chacun à un état du système. Comme nous l'avons vu précédemment, un tel phénomène n'a pas lieu dans notre cas d'application. En d'autres termes, la valeur de la métrique associée au critère *Réalisme dynamique* dépend de la valeur de la métrique *Fiabilité du système* qui elle-même dépend très fortement des métriques associées au facteur informationnel (*Complétude de l'information* et *Exactitude de l'information*).

3.5. Utilisation de seuils et de poids

Les notions de seuils et de poids permettent de prendre davantage en compte la spécificité d'un projet en apportant des points de vue différents.

3.5.1 Utilisation de seuils

Pour ce projet, les décideurs ont décidé de fixer la même valeur de seuil pour toutes les mesures. Cette valeur a été fixée à 0,5. Ainsi, toutes les mesures qui se situent en dessous ce cette valeur requièrent la mise en œuvre d'actions correctives.

Les mesures correspondant à la *Satisfaction des buts* (0,07), la *Complétude de l'information* (0,29) et le *Réalisme dynamique* (0,18) ont des valeurs inférieures au seuil. Concernant la *Complétude de l'activité* et l'*Exactitude de l'activité*, les valeurs obtenues sont inférieures à 0,5 pour trois activités : 'créer chambre', 'fermer station définitivement' et 'modifier réservation'.

La mise en œuvre d'actions correctives doit d'abord concerner ces mesures afin de les amener à une valeur supérieure au seuil. Ainsi, on commencera par corriger l'alignement en cherchant à accroître la *Complétude de l'information* puisque cette métrique a des répercussions sur les autres.

3.5.2 Utilisation de poids

Il existe différents critères pour déterminer un poids. Les dirigeants du groupement d'hôtels ont décidé d'attribuer un poids à chacune des activités en fonction de la fréquence d'exécution de cette activité. Le Tableau 19 présente les poids attribués à chacune des activités.

Poids	Activités UML	Poids	Activités UML
5	Effectuer une demande	1	Modifier Hôtel
3	Mettre en attente	0,5	Modifier Station
2	Mettre une demande hors délai	1,5	Modifier Chambre
3,5	Mettre une demande sans suite	0,5	Fermer hôtel définitivement
1	Annuler demande	0,5	Fermer station définitivement
4	Satisfaire demande	1	Annuler Réservation
5	Gérer les disponibilités	1	Supprimer chambre
4	Créer réservation	1,5	Créer Hôtel
1	Modifier réservation	1,5	Créer Chambre
4	Facturer	0,5	Créer Fermeture temporaire station
2,5	Annuler Réservation par le client	1	Créer Fermeture temporaire hôtel
1,5	Facturer pénalité annulation	0,5	Fermer station temporairement
1,5	Payer pénalité	1	Créer nouvelles réservations par l'opérateur
3	Consommer	1	Fermer hôtel temporairement
1,5	Facturer pénalité due à une consommation partielle	0,5	Créer station

Tableau 19 : Poids attribués à chacune des activités

Le Taux de support pondéré peut s'écrire de la façon suivante :

$$Ts_p = \sum \alpha^r_i \ / \sum \alpha_j = 53/60 \approx 0,88$$

Ainsi, on remarque qu'en pondérant les activités en fonction de leur fréquence d'exécution, la valeur de la mesure est nettement supérieure. Cette différence entre Ts_p et Ts signifie que les activités les plus fréquemment exécutées sont majoritairement gérées par le système. En d'autres termes, la valeur de Ts_p précise que dans 88% des cas, l'activité effectuée est prise en charge par le système. Il est important néanmoins de noter que le nombre d'activités gérées par le système ne change pas, c'est juste l'importance relative de chaque activité qui est modifiée et apporte un éclairage nouveau.

De la même façon, il est possible de pondérer chacune des huit autres mesures, éventuellement suivant des critères différents.

4. Conclusion

Dans ce chapitre, nous avons montré comment utiliser les métriques et analyser les valeurs correspondantes, à partir d'un cas d'application de la gestion de réservation de chambres d'hôtel.

Ces neuf métriques correspondent à neuf façons différentes et complémentaires de raisonner sur l'alignement.

Les différentes valeurs obtenues pour les métriques montrent que la focalisation de l'évaluation de l'alignement sur le *Taux de support* est très réductrice. En effet, dans le cas développé dans ce chapitre cette métrique vaut 0,71 alors que les métriques *Exactitude de l'information* et *Réalisme dynamique* sont respectivement égales à 0,5 et à 0,18. Ces deux valeurs signifient que l'information n'est pas gérée de la même façon dans les processus et le système et que seule une faible proportion des chemins peut être effectuée naturellement avec l'aide du système.

Une faible valeur pour certaines de ces métriques peut avoir des répercussions sur d'autres métriques, ce qui, à l'inverse, signifie qu'engager des actions correctives pour rétablir l'alignement et chercher à augmenter la valeur d'une mesure peut aussi bénéficier à d'autres mesures.

L'exemple présenté dans ce chapitre a aussi montré que certaines métriques pouvaient atteindre la valeur 1, correspondant à l'alignement parfait, alors que d'autres restent faibles (inférieures à 0,2).

L'utilisation de poids, correspondant par exemple à la fréquence d'utilisation (de l'activité, de la classe ou de la transition d'états) permet d'atténuer certaines des mauvaises valeurs calculées. En effet, en identifiant précisément les différentes activités, on se rend compte que les métriques *Complétude de l'activité* et *Exactitude de l'activité* sont les plus élevées pour tout ce qui touche à la réservation. En revanche les activités concernant la gestion de la liste de produits obtiennent de très mauvaises valeurs, souvent nulles. Cependant, ces activités sont indispensables même si elles ont un impact économique moindre que les autres et sont moins fréquemment utilisées.

L'utilisation de seuil permet aux dirigeants de préciser leurs priorités. Ceci permet ainsi d'identifier des actions correctives pour rétablir l'alignement.

CHAPITRE 5 : LA METHODE ACEM

1. Introduction

Dans la première partie de cette thèse, nous avons défini le concept d'alignement ; nous avons également proposé un ensemble de métriques pour évaluer l'alignement entre un modèle de processus d'entreprise et un modèle de système d'information.

Dans cette deuxième partie, nous nous intéressons à l'ingénierie de l'alignement. Nous proposons la méthode ACEM (**A**lignment **C**orrection and **E**volution **M**ethod) pour :

1. adapter le modèle de processus d'entreprise et/ou le modèle de système afin de rétablir l'alignement entre ces deux entités et,
2. faire évoluer conjointement les modèles de processus d'entreprise et de système pour prendre en compte des exigences d'évolution.

La Figure 41 schématise notre approche.

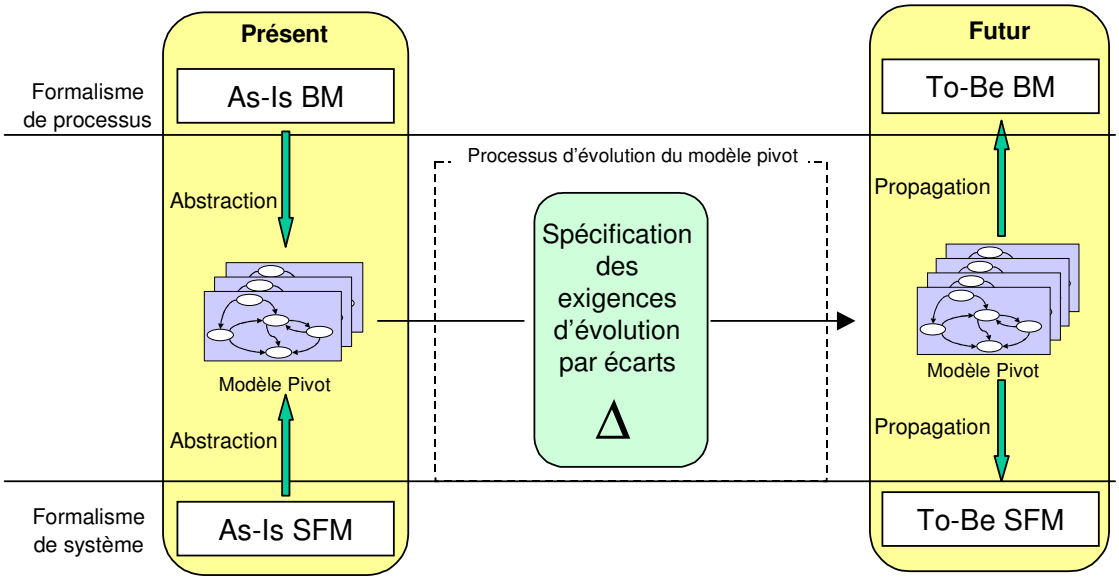


Figure 41 : Schéma de l'approche ACEM

L'approche ACEM se place dans un cadre de conduite du changement qui permet à une organisation de passer d'une situation présente à une situation future. La situation présente est caractérisée par les modèles As-Is BM (Business Model) et As-Is SFM (System Functionality Model) qui représentent les processus d'entreprise et les fonctionnalités du système. La situation future est caractérisée par les modèles To-Be BM et To-Be SFM représentant la situation du métier et du système respectivement, après évolution.

Les processus d'entreprise et le système sont également représentés de façon unifiée au sein d'un modèle, le modèle pivot. C'est ce modèle que nous faisons évoluer pour prendre en compte des exigences d'évolution exprimées sous la forme d'écarts. Ces écarts sont ensuite propagés sur les modèles de système et de processus.

Les sous-sections suivantes présentent succinctement chacune des spécificités de cette approche.

2. Cadre de changement

La méthode ACEM adopte le cadre de changement de Jarke [Jarke93]. Nous considérons que le changement crée un mouvement d'une situation existante vers une nouvelle situation. Le changement est donc perçu comme un phénomène discret ou discontinu. Les processus d'entreprise et le système d'information ne sont pas considérés comme étant en perpétuelle évolution. Au contraire, les situations stables alternent avec les mises en œuvre de changements [Lewis98].

La situation existante est représentée par les modèles As-Is ; la nouvelle situation par les modèles To-Be. Selon Jackson [Jackson95], les modèles As-Is décrivent des propriétés indicatives, c'est-à-dire possédées par le système alors que les modèles To-Be décrivent des propriétés optatives, c'est-à-dire qu'on aimerait que le système possède.

Dans chacune des situations, nous identifions deux modèles représentant respectivement le système et les processus d'entreprise. Ces modèles n'utilisent pas les mêmes formalismes.

La Figure 42 présente le cadre de changement général que nous adoptons et dans lequel quatre modèles sont utilisés ;

- le modèle As-Is BM (Business Model) représente les processus d'entreprise dans la situation existante,
- le modèle As-Is SFM (System Functionality Model) rassemble les fonctionnalités du système telles qu'elles existent avant la mise en œuvre du changement,
- le modèle To-Be BM correspond aux processus d'entreprise une fois les changements réalisés et
- le modèle To-Be SFM précise les futures fonctionnalités du système.

Ces quatre modèles sont organisés sur deux axes :

- l'axe du *temps*, horizontal, qui permet de faire la distinction entre les situations présentes (modèles As-Is) et futures (modèles To-Be) ;
- l'axe du *sujet*, vertical, qui permet de positionner le système (modèles SFM) par rapport aux processus d'entreprise (modèles BM).

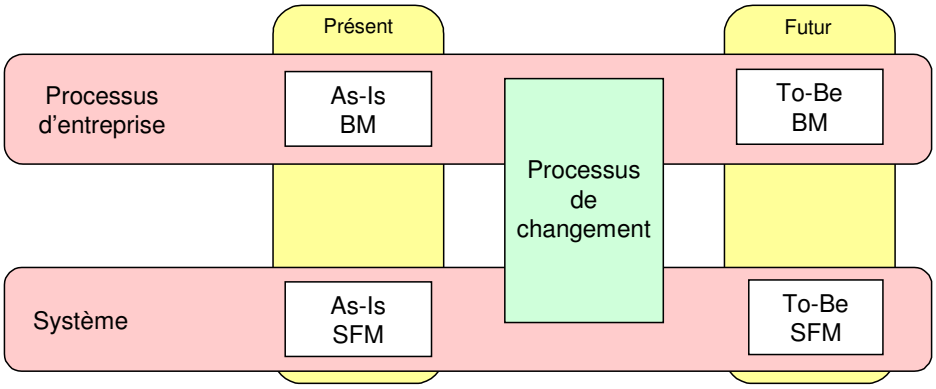


Figure 42 : Cadre de changement

Ce cadre de changement sert de base à notre approche mais a été modifié pour prendre en compte ses spécificités. La section suivante présente les modifications liées à l'utilisation d'un modèle pivot.

3. Modèle pivot

La spécificité de l'approche ACEM réside dans l'utilisation d'un modèle pivot pour matérialiser la relation d'alignement. Ce modèle modifie la structure générale du cadre de changement définie ci-dessus. Il permet de représenter à la fois le système et les processus d'entreprise au sein d'un même modèle. Nous ajoutons donc au cadre de changement général un niveau intermédiaire sur l'axe du sujet qui permet de représenter conjointement, au sein d'un même modèle, le système et les processus. De cette façon, on différencie le niveau de l'organisation, le niveau du système et le niveau commun.

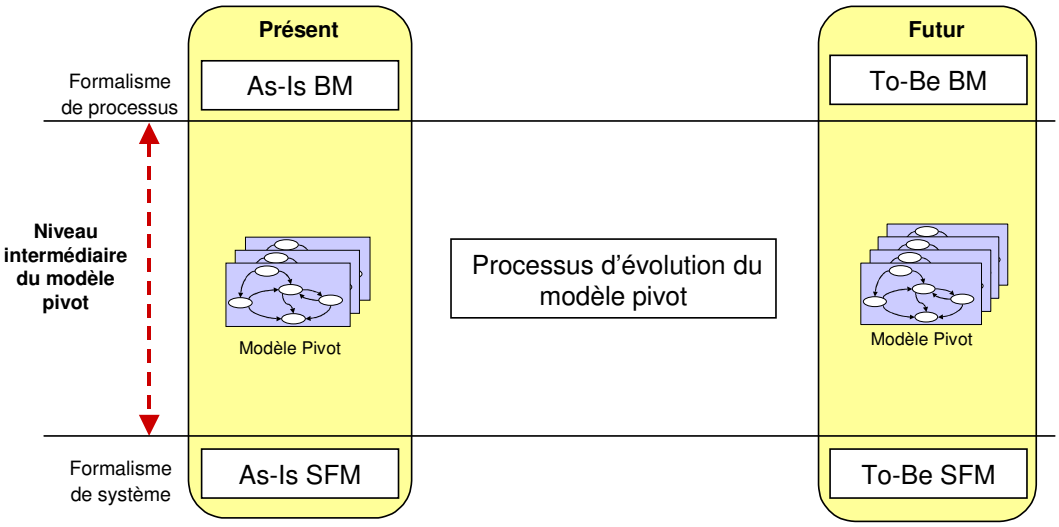


Figure 43 : Utilisation d'un niveau intermédiaire

L'utilisation d'un modèle pivot est motivée par la discordance de langages utilisés pour représenter les systèmes et les processus. En effet, les modèles de systèmes utilisent des concepts tels que "objets", "états", "opérations" ou "événements". Les modèles de processus d'entreprise manipulent les concepts "d'acteurs", de "rôles", "d'activités".

Inspirée par l'ingénierie des exigences, notre approche se base sur un lien que l'on peut établir entre ce que le système peut faire (le "quoi" décrit par les spécifications du système) et pourquoi il doit le faire (le "pourquoi" matérialisé par les exigences des parties prenantes). L'abstraction des fonctionnalités du système pour en éliminer les détails et en déterminer l'essentiel permet de représenter le système en termes de buts à atteindre. De la même façon, on peut distinguer les tâches des processus de leur réalisation pour se concentrer sur les objectifs métiers. Il est ainsi possible d'exprimer le point de vue du système et celui des processus avec le même langage. Des mécanismes d'affinement permettent de détailler les objectifs à atteindre pour représenter les spécifications du système et les tâches des processus. C'est la position que nous avons retenue.

Nous avons choisi d'utiliser le méta-modèle de Carte comme base du méta-modèle pivot. Cependant, ce méta-modèle a été enrichi afin de permettre une représentation explicite de la relation d'alignement. Le méta-modèle pivot possède différents avantages : (i) en tant que

méta-modèle intentionnel il permet de modéliser dans les mêmes termes, le système et les processus, (ii) un principe de coloration permet de représenter le système et les processus au sein du même modèle, (iii) la relation d'alignement est explicite et son degré est précisé dans le modèle pivot et (iv) un mécanisme d'affinement permet de représenter la relation d'alignement à différents niveaux d'abstraction.

4. Mise en œuvre de l'évolution

Nous avons identifié deux raisons principales à l'évolution du modèle pivot : (1) le besoin de rétablir l'alignement et (2) la prise en compte de nouveaux besoins. Dans chacun des deux cas, l'évolution du modèle pivot se base sur l'expression explicite des exigences d'évolution au moyen d'écarts. Cette façon de procéder permet de gagner du temps en se focalisant sur ce qui change et en évitant de re-décrire tout ce qui reste inchangé entre le présent et le futur.

Un *écart* exprime une différence entre une situation initiale, le modèle As-Is, et la situation future, le modèle To-Be. Le changement est alors modélisé sous la forme d'une collection d'écarts [Salinesi04b]. Le modèle futur est ainsi construit pas à pas sans idée préconçue, mais en identifiant précisément ce qui doit changer par rapport à l'état actuel (Figure 44).

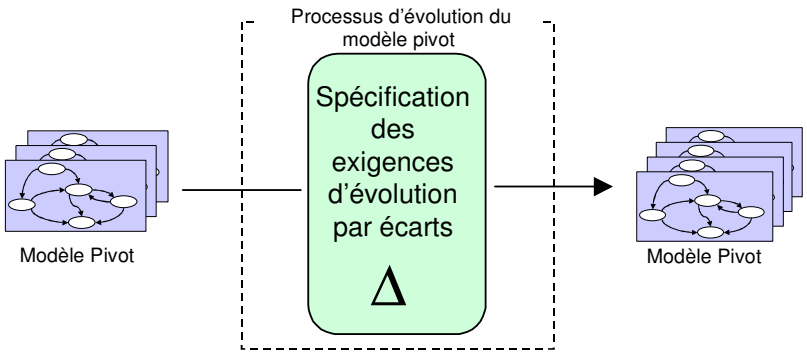


Figure 44 : Processus d'évolution du modèle pivot

Les écarts expriment ce qui doit être ajouté, fusionné, remplacé… Les transformations du modèle As-Is pour obtenir le modèle To-Be sont exprimées par des *opérateurs*.

Afin de proposer un ensemble d'écarts indépendants de tout modèle et de faciliter la construction de typologies spécifiques associées à des modèles particuliers, nous définissons une typologie générique d'écarts et un processus pour générer les premières à partir de cette dernière.

5. Proposition d'une démarche guidée

La méthode ACEM repose sur un processus en trois étapes (Figure 45) :

1. l'obtention du modèle pivot
2. l'évolution du modèle pivot
3. la propagation des écarts.

La première étape correspond à la construction du modèle pivot à partir des modèles de processus et de système. Il est pour cela nécessaire (1) d'abstraire les fonctionnalités du

système et les activités métiers dans deux ensembles de cartes séparés et (2) de fusionner ces deux ensembles en précisant le degré d'alignement entre sections.

La deuxième étape correspond à l'évolution du modèle pivot. Elle repose sur l'identification d'écarts correspondant aux exigences d'évolution et exprimés à partir d'opérateurs d'écart [Rolland04].

La troisième étape correspond à la propagation des écarts identifiés sur le modèle pivot dans les modèles du système et des processus. Cette partie du processus s'appuie sur la traçabilité des liens de correspondance entre les éléments du modèle pivot et ceux du modèle de processus (ou du système).

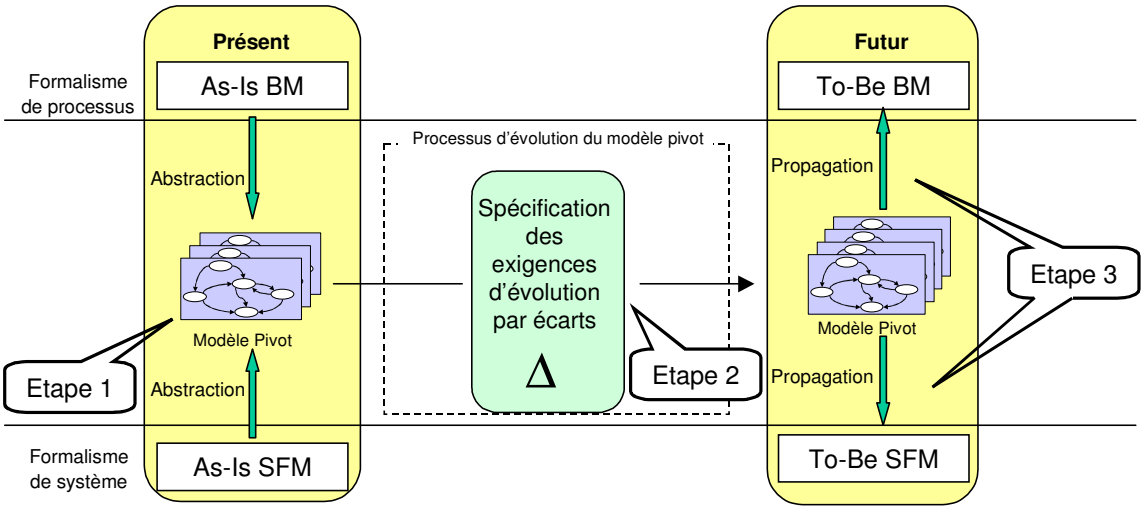


Figure 45 : Processus en trois étapes

Les écarts identifiés lors de la deuxième étape peuvent être de natures différentes : ils peuvent exprimer une évolution ou une amélioration de l'alignement. La méthode ACEM propose néanmoins une seule démarche guidée : elle est multi-démarche.

Le méta-modèle de Carte est utilisé dans la méthode ACEM comme modèle de description de guides méthodologiques utilisables de manières flexibles. Ce méta-modèle permet de décrire plusieurs chemins au sein d'un même processus ; chaque chemin décrivant une variante du processus. Le méta-modèle de Cartes propose des directives définies sous formes d'aides méthodologiques qui aident les ingénieurs (i) à choisir le chemin le mieux approprié à la situation pour et (ii) à réaliser les différentes étapes du processus ACEM.

La suite de la thèse est organisée en quatre chapitres :

- le Chapitre 6 présente le méta-modèle pivot.
- le Chapitre 7 décrit les opérateurs d'écarts permettant d'exprimer les exigences d'évolution.
- le Chapitre 8 correspond à la description formelle du processus de la méthode ACEM.
- le Chapitre 9 illustre la méthode ACEM.

CHAPITRE 6 : LE META-MODELE PIVOT

1. Introduction

La méthode ACEM propose de faire évoluer conjointement un système d'information et les processus d'entreprise associés. Or l'expérience montre que l'évolution de l'une des deux entités aboutit souvent à une rupture de l'alignement [Soffer04b]. Il semble donc important d'être capable de maîtriser l'évolution de ces deux entités et en particulier d'interdire (ou tout au moins de réduire) toute possibilité d'évolution divergente. Nous proposons d'utiliser un modèle pivot pour représenter la relation d'alignement entre système et processus et ainsi permettre une évolution conjointe de ces deux entités.

Nous pensons que la représentation de l'alignement doit satisfaire différents principes :

- Abstraire les processus d'entreprise et le système au niveau intentionnel ;
- Rendre explicite la relation d'alignement ;
- Affiner la relation d'alignement.

Afin de respecter ces principes nous proposons d'utiliser le méta-modèle intentionnel de Carte [Rolland01] comme base du méta-modèle pivot ACEM et de l'enrichir avec un système de coloration permettant (i) la représentation, au sein d'un même modèle, du système et des processus et (ii) la spécification du degré d'alignement.

Dans ce chapitre, nous présentons différents principes régissant la représentation de l'alignement et précisons les caractéristiques que doit posséder le méta-modèle pivot ACEM pour les satisfaire ; la section 3 présente en détail les concepts du méta-modèle pivot ; la section 4 conclut le chapitre.

2. Principes régissant la représentation de l'alignement

2.1. Abstraire les processus d'entreprise et le système au niveau intentionnel

La difficulté d'aligner système et processus d'entreprise tient en partie à la disparité des langages utilisés pour spécifier ces deux entités. Les experts du système et les experts métier emploient chacun leur documentation, leurs propres outils, possèdent leur propre langage et rencontrent des difficultés à se faire comprendre de l'autre partie [Luftman00]. Le langage des experts d'entreprise repose sur les concepts d'acteurs, de buts, d'activités, de façon d'atteindre un but… Le langage de l'ingénieur système est technique. Il se base sur les modèles de base de données, d'interface, de collaboration entre objets, etc. Il fait référence aux concepts d'objets, d'événements, d'états… Le risque lié à cette incompréhension mutuelle est (i) de construire un système qui ne correspond pas aux besoins réels des futurs utilisateurs, d'aboutir à un mauvais alignement entre le système et le processus d'entreprise ou même à l'échec du

projet [Standish95], [Davenport98], [Davenport00] ou (ii) de faire évoluer de façon disjointe le système et les processus.

Ces différences rendent le problème de l'alignement difficile voire impossible à résoudre de manière systématique. Nous pensons qu'une meilleure homogénéité de langage est susceptible d'améliorer l'expression des besoins de l'organisation et celle des fonctionnalités du système et peut forcer une réflexion conjointe de l'évolution de l'ensemble formé par le système et les processus.

Nous proposons de *représenter l'alignement au niveau intentionnel* (Figure 46). Les descriptions du système et celles des processus doivent donc être abstraites pour s'adapter aux contraintes et caractéristiques d'une expression intentionnelle. Les détails sont abstraits pour se concentrer sur l'essence du système et des processus, à savoir les objectifs qu'ils permettent d'atteindre et ainsi rendre les activités des processus et les fonctionnalités du système indépendantes de leur réalisation. La quantité de détail à manipuler est ainsi plus faible, permettant une meilleure maîtrise du système, des processus et de leur alignement.

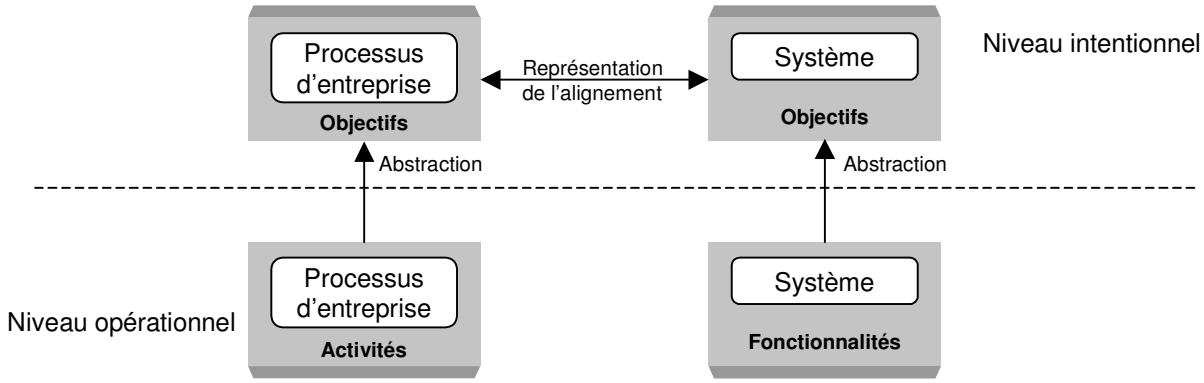


Figure 46 : Représentation de l'alignement au niveau intentionnel

En ingénierie des besoins, les langages intentionnels (orientés but) ont montré qu'ils permettaient un meilleur alignement des exigences du système aux besoins de l'organisation [Potts94], [Bubenko94], [Rolland98]. On peut donc penser que les approches orientées but peuvent aider à représenter l'alignement en proposant un langage commun pour représenter le système et les processus d'entreprise.

2.2. Rendre explicite la relation d'alignement

En dépit du fait que la relation d'alignement est mentionnée dans de nombreuses approches, elle est rarement considérée comme un concept à part entière. Par exemple, la plupart des méthodologies orientées-objet récentes défendent le fait que les modèles du système et de processus doivent être liés. Cependant, elles adoptent un processus aboutissant à un couplage faible et indirect. Une partie de processus est liée à un composant du système au travers d'un grand nombre d'étapes de développement. La relation entre un cas d'utilisation et un composant dans Catalysis [D'Souza99] ou le RUP [Jacobson99] sont des exemples de couplage faible. Les liens entre le système et les processus ne sont pas explicites ce qui peut entraîner des difficultés pour maintenir l'alignement, en particulier lors d'évolutions.

Les langages orientés but captent explicitement le “quoi” et le “pourquoi” des fonctionnalités du système et des processus d’entreprise [Yu01] et [Lamsweerde01]. Les approches utilisant ces langages aident à établir un couplage direct entre processus d’entreprise et système, dans la mesure où un but est opérationnalisable en fonctionnalités du système. Par exemple, dans une approche d’ingénierie des exigences telle que KAOS [Lamsweerde01], un but opérationnalisable est directement relié à une contrainte du système exprimé en termes d’objets, d’actions et d’événements du système qui contribue à la réalisation du but.

Nous proposons de représenter la relation d’alignement de manière explicite dans un modèle que nous qualifions de pivot. Le modèle pivot est le résultat de la *fusion* de deux ensembles de cartes [Rolland01], l’un représentant le système et l’autre le processus. Il permet ainsi de modéliser conjointement le système et les processus et de définir des évolutions conjointes de ces deux entités. Un *principe de coloration* enrichit le méta-modèle de Carte afin (i) de rendre explicite la relation d’alignement (ii) d’en préciser le degré et (iii) de spécifier, en cas d’alignement partiel ou de non alignement si la section modélise le système ou les processus.

2.3. Affiner la relation d’alignement

Il n’est pas possible de représenter de façon précise dans une seule carte pivot la complexité du système et des processus. Afin de gérer cette complexité, nous proposons d’utiliser l’affinement qui est un mécanisme d’abstraction permettant de voir une entité comme un ensemble d’entités reliées les unes aux autres. Nous pensons qu’un tel mécanisme d’affinement est nécessaire pour gérer la relation d’alignement de façon *systématique* et *contrôlée*. Il permet une approche par niveaux et aide ainsi à maîtriser progressivement la complexité de la relation d’alignement.

Nous proposons d’adopter le mécanisme d’affinement du méta-modèle de Carte qui permet de décrire une section par un graphe d’intentions et de stratégies. Associé au principe de coloration, le mécanisme d’affinement permet d’étudier la relation d’alignement à différents niveaux d’abstraction. Ainsi, ce qui peut sembler parfaitement aligner à un niveau d’abstraction élevé peut présenter des ruptures d’alignement à un niveau d’abstraction plus faible.

3. Description du méta-modèle pivot

Le méta-modèle pivot que nous proposons est présenté à la Figure 47. Il utilise les concepts du méta-modèle de Carte [Rolland01] permettant une représentation intentionnelle du système et des processus au sein du modèle pivot. Ces concepts ont été enrichis pour rendre explicite la relation d’alignement et préciser les objets métier manipulés par les processus d’entreprise et les objets utilisés par le système.

La partie grisée de la Figure 47 correspond à la partie intentionnelle.

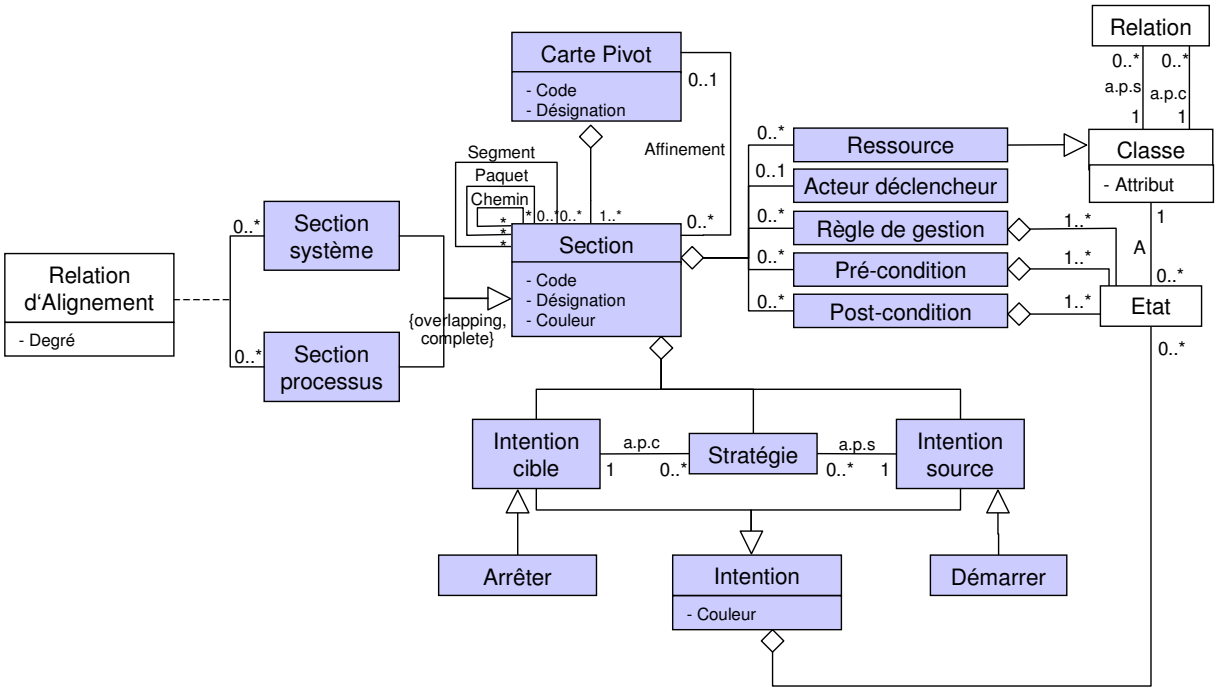


Figure 47 : Méta-modèle pivot

Le reste de la section présente les différents concepts du méta-modèle pivot en commençant par les concepts intentionnels issus du méta-modèle de Carte, puis le principe de coloration permettant de représenter explicitement l'alignement.

3.1. Carte pivot

Le modèle pivot se présente sous forme de graphes appelés cartes pivot. Le méta-modèle pivot présenté à la Figure 47 montre que :

- Une *carte pivot* a un code et une désignation permettant de l'identifier.
- Une carte pivot est composée de plusieurs sections ; une section étant une agrégation de deux intentions liées par une stratégie.

Une carte pivot se présente sous la forme d'un graphe orienté de *Démarrer* à *Arrêter*. Les intentions sont les nœuds et les stratégies sont les liens entre les nœuds. Un lien entre dans un nœud si la stratégie correspondante peut être utilisée pour atteindre l'intention en question. Dans la mesure où il peut y avoir plusieurs liens entrant dans un même nœud, le modèle pivot est capable de représenter plusieurs stratégies pour atteindre une même intention. La Figure 48 montre un exemple de carte pivot correspondant à la gestion de réservation de produits hôteliers. Cette carte pivot compte deux intentions (*Offrir des packages* et *Gérer la relation client*) en plus des intentions *Démarrer* et *Arrêter* et 14 stratégies.

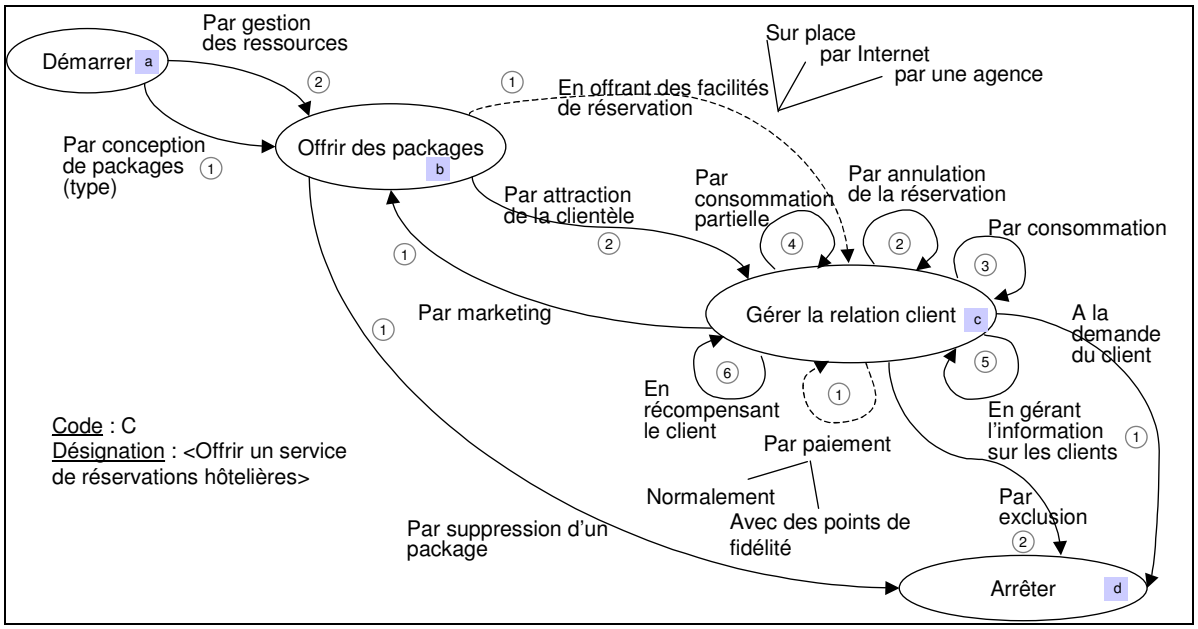


Figure 48 : Un exemple de carte pivot de gestion de réservation de produits hôteliers

3.2. Intention

Une *Intention* est un but qui peut être atteint par la réalisation d'un ensemble d'objectifs ou d'activités. Une intention est selon [Jackson95] une déclaration 'optative' qui exprime ce que l'on veut, un état ou un résultat que l'on cherche à atteindre. Par exemple, *Offrir des packages* est une intention correspondant à la conception, la mise à jour et la mise à disposition d'un catalogue de packages (c'est-à-dire un ensemble de produits hôteliers).

Nous postulons que toute carte pivot a deux intentions particulières, *Démarrer* et *Arrêter* pour respectivement commencer et terminer une carte.

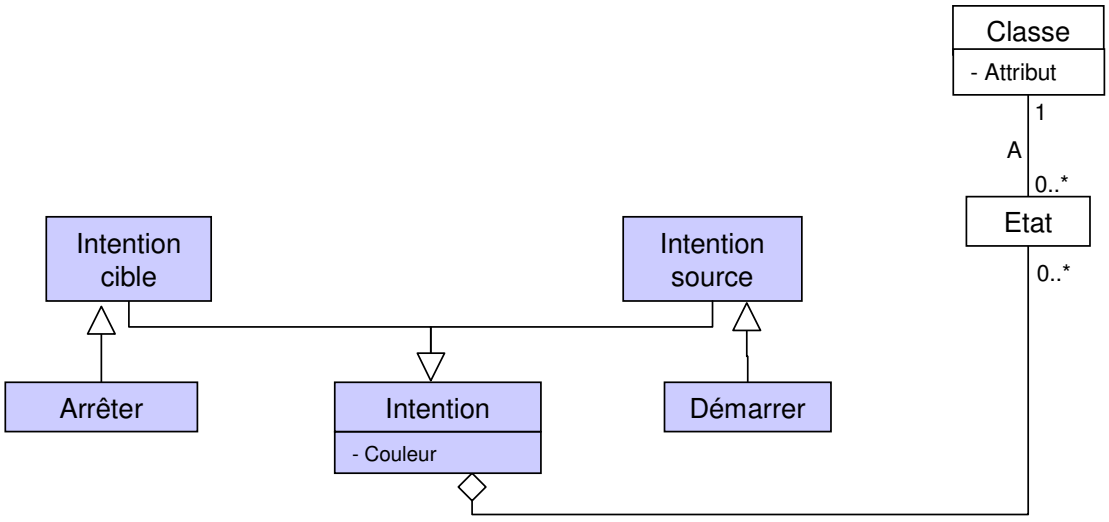


Figure 49 : Caractéristiques d'une intention

Le modèle pivot peut être vu comme un ensemble de processus complexes et alternatifs. A cet égard et comme tout processus, il agit sur des objets. En d'autres termes, l'exécution des processus modélisés dans le modèle pivot correspond à des transformations successives

d'objets. Afin de préciser les états que peuvent atteindre ces objets, toute intention I est définie comme un ensemble *d'états désirables* G_I. Ces états sont des états de classes.

Dans l'exemple des réservations hôtelières, l'intention *Offrir des packages* de la Figure 48 a pour conséquence que (i) un catalogue de produits est offert, (ii) des disponibilités sont proposées (iii) les réservations affectées par des mises à jour de produit sont modifiées, si besoin. Cette intention, conformément au méta-modèle pivot peut s'écrire en terme d'états à atteindre de la façon suivante :

G_I = {package.état = 'créé', package.état = 'fermé_temp', package.état = 'fermé', disponibilité.état = 'supprimée', disponibilité.état = 'créée', réservation.état = 'créée'}

où package, réservation et disponibilité sont des classes.

L'étude des intentions permet d'identifier les classes manipulées par les processus d'entreprise ou le système.

3.3. Stratégie

Une *Stratégie* est une façon, une manière d'atteindre une intention. Elle se représente sous forme de flèche dans le graphe afin d'indiquer l'intention que l'on cherche à réaliser (intention cible), à partir d'une situation donnée (exprimée par l'intention source).

Dans notre exemple, il existe différentes façons d'offrir des packages : par conception de package ou par gestion des ressources. *Par conception de package* est une stratégie ; c'est une façon d'atteindre l'intention *Offrir des packages*.

Le concept de stratégie permet (i) de distinguer le but (*Offrir des packages*) de la façon de l'atteindre (*Par conception de package*) et (ii) d'exprimer des approches alternatives (*Par gestion des ressources*).

3.4. Section

Une *Section* est une agrégation d'une *Intention source*, d'une *Intention cible* et d'une *Stratégie*. Une section exprime la stratégie selon laquelle en partant de l'intention source, l'intention cible peut être atteinte. Sa désignation correspond au triplet formé par l'intention source, l'intention cible et la stratégie.

Par exemple, comme le montre la Figure 48, l'agrégation de l'intention source *Démarrer*, de l'intention cible *Offrir des packages* et de la stratégie *Par conception de package* définit une section dont la désignation est <*Démarrer*, *Offrir des packages*, *Par conception de package*>. Ici, la stratégie *Par conception de package* caractérise le flux de l'intention source *Démarrer* vers l'intention cible *Offrir des packages* et la façon dont cette dernière peut être atteinte.

Selon le méta-modèle pivot, une section peut être vue comme la transition d'une situation initiale, obtenue par la réalisation de l'intention source, vers une situation finale résultant de la réalisation de l'intention cible par l'exécution de règles de gestion associées à la section. Ces

aspects sont spécifiés par trois propriétés associées à l'élément section dans le méta-modèle pivot. La Figure 50 présente un extrait de ce méta-modèle :

- la *pré-condition* caractérise la situation initiale, c'est-à-dire les conditions devant être satisfaites pour que la section puisse être exécutée ;
- la *post-condition* définit la situation finale. La post-condition permet de préciser les états des classes après la réalisation de la section.
- les *règles de gestion* permettent de préciser les contraintes régissant la réalisation d'opérations entraînant le changement d'état d'objets.

Par exemple, la pré-condition de la section <*Démarrer*, *Offrir des packages*, *Par gestion des ressources*> consiste en l'existence de produits hôteliers appelés packages. La post-condition correspond au fait que le produit a été modifié et éventuellement que les disponibilités ont été mises à jour. La règle de gestion permet de préciser que seul les packages créés peuvent être fermés temporairement.

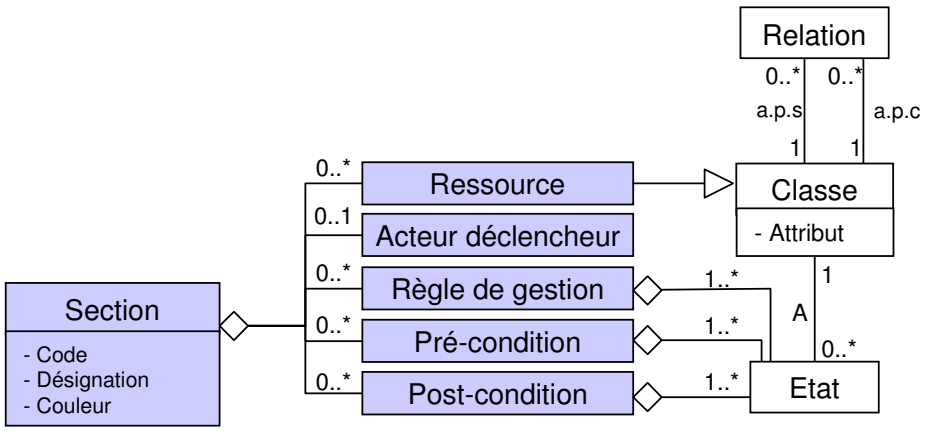


Figure 50 : Caractéristiques d'une section

Les pré-conditions, post-conditions et règles de gestion font référence à des états de classes. Pour la section S dont l'intention source est I, l'intention cible est J et la stratégie St, la pré-condition de la section S est un sous-ensemble de G_I. De la même façon, la post-condition de la section S est un sous-ensemble de G_J. Les règles de gestion, quant à elles, utilisent les états du sous-ensemble $G_I \cup G_J$. Ainsi, si nous prenons l'exemple de la section <*Démarrer*, *Offrir des packages*, *Par gestion des ressources*> :

- la pré-condition s'écrira *package.état = 'créé'* ∨ *package.état = 'fermé_temp'* ∨ *package.état = 'fermé'* ce qui signifie que l'état de la classe package est créé, fermé_temp (correspondant à une fermeture temporaire) ou fermé,
- la post-condition s'écrira *package.état = 'créé'* ∨ *package.état = 'fermé_temp'* ∨ *package.état = 'fermé'*,
- la règle de gestion s'écrira par exemple <*package.état = 'créé', package.état = 'fermé_temp'*> ce qui signifie qu'un changement d'états est possible entre les états 'créé' et *'fermé_temp'* de la classe package.

Une section S de l'intention I à l'intention J part d'un sous ensemble $I_S \subseteq G_I$, et finit dans un sous-ensemble $F_S \subseteq G_J$ [Soffer05].

Le méta-modèle ACEM permet également de préciser deux autres caractéristiques significatives : l'acteur qui déclenche la réalisation de la section et les ressources utilisées lors de cette réalisation. Ceci se matérialise par l'existence des concepts *Acteur déclencheur* et Ressources dans le méta-modèle pivot. Dans notre exemple de la section <*Offrir des packages*, *Gérer la relation client*, *En offrant des facilités de réservation*>, l'acteur déclencheur est le client qui fait une demande de réservation… Toute section n'est pas forcément déclenchée par un acteur. Certaines sections peuvent être déclenchées par ce qu'on appelle un événement interne, c'est-à-dire une modification d'objet dont le changement d'état provoque à son tour un changement.

Les ressources correspondent aux classes dont la section a besoin pour sa réalisation mais qui ne sont l'objet d'aucune action. Dans notre exemple de réservation, la section <*Offrir des packages*, *Gérer la relation client*, *En offrant des facilités de réservation*> utilise les ressources suivantes : la station, l'hôtel, les activités (composant les packages) et les chambres si les disponibilités sont gérées par chambre. En effet, l'existence d'un catalogue de produits est indispensable à la réservation.

3.5. Liens entre sections

Le méta-modèle pivot précise que les sections sont connectées entre elles de différentes façons comme le montre la Figure 51 :

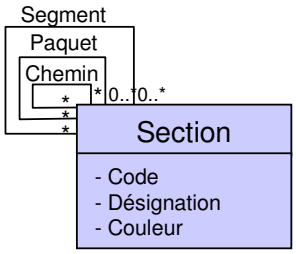


Figure 51 : Liens entre sections

Ces trois liens ont des sémantiques différentes. Ils permettent de :

- établir quelles intentions précèdent ou succèdent telles autres. La relation de précédence indique qu'une intention ne peut être réalisée que si une autre intention a été réalisée. La relation de précédence conduit à ordonner les sections dans un *chemin*. Dans un chemin, l'intention cible de la section qui précède est l'intention source de la section qui lui succède. Un chemin est donc un sous-ensemble de sections de la carte pivot où les sections sont reliées par une relation de précédence.

  Dans l'exemple de la Figure 48, il apparaît que la section <*Offrir des packages*, *Gérer la relation client*, *Par attraction de la clientèle*> ne peut être satisfaite que si des packages sont proposés au sein d'un catalogue de produits. Il existe donc un chemin entre la section <*Démarrer*, *Offrir des packages*, *Par conception de package*> et la section <*Offrir des packages*, *Gérer la relation client*, *Par attraction de la clientèle*>. *Offrir des packages* est l'intention cible de la première section et l'intention source de la deuxième. Si, à partir d'une intention source, il est possible d'atteindre une même intention cible en suivant plusieurs chemins dans la carte pivot, on parle alors de topologie *multi-chemin*.

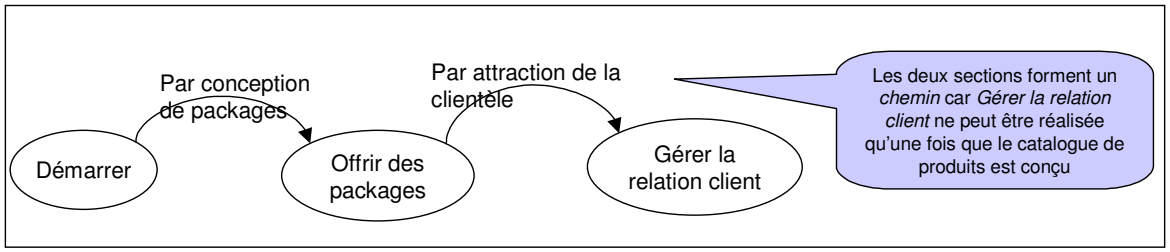


Figure 52: Exemple de chemin

• spécifier qu'une intention donnée peut être réalisée selon plusieurs stratégies différentes. Ceci se représente dans une carte pivot par plusieurs sections entre une même paire d'intentions. Toutes les sections ont la même intention source et la même intention cible. Une telle topologie s'appelle un *multi-segment*. Une ou plusieurs de ces sections peuvent être réalisées. Ces sections sont reliées entre-elles par une relation OR.

Par exemple, considérons qu'il existe deux façons différentes d'arrêter le processus, à la demande du client ou par exclusion. Les deux stratégies *A la demande du client* et *Par exclusion* ont la même intention source *Gérer la relation client* et la même intention cible *Arrêter*. Les deux sections <*Gérer la relation client*, *Arrêter*, *A la demande du client*> et <*Gérer la relation client*, *Arrêter*, *Par exclusion*> forment une topologie *multi-segment*.

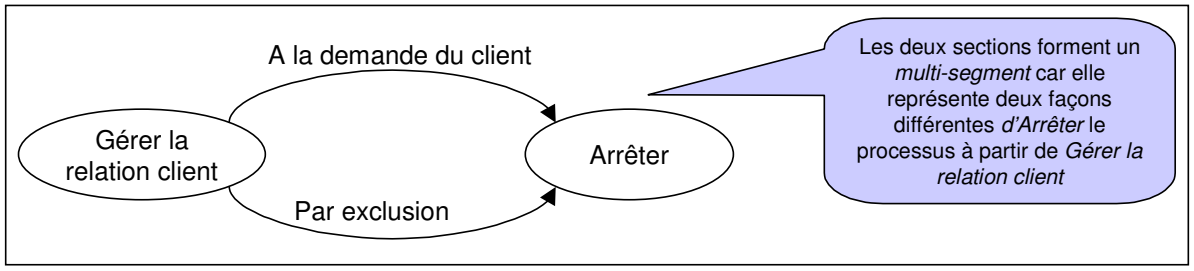


Figure 53: Exemple de multi-segment

• préciser que des sections ayant les mêmes intentions sources et les mêmes intentions cibles sont mutuellement exclusives. On dit alors que ces sections forment un paquet. Le choix d'une de ces sections pour la réalisation de l'intention cible empêche la réalisation des autres sections formées des autres stratégies. Les différentes sections sont mutuellement exclusives. Elles sont reliées par une relation XOR (OU exclusif).

Considérons les intentions *Offrir des packages* et *Gérer la relation client*. Admettons également que l'hôtel offre trois façons différentes de réserver un package : sur place, par Internet ou par l'intermédiaire d'une agence de voyage ('ou' est exclusif). Un seul des trois modes de réservation doit être utilisé. Le nom du paquet est *En offrant des facilités de réservation*. Il regroupe trois alternatives exclusives, *Sur place, par Internet* et *par une agence*.

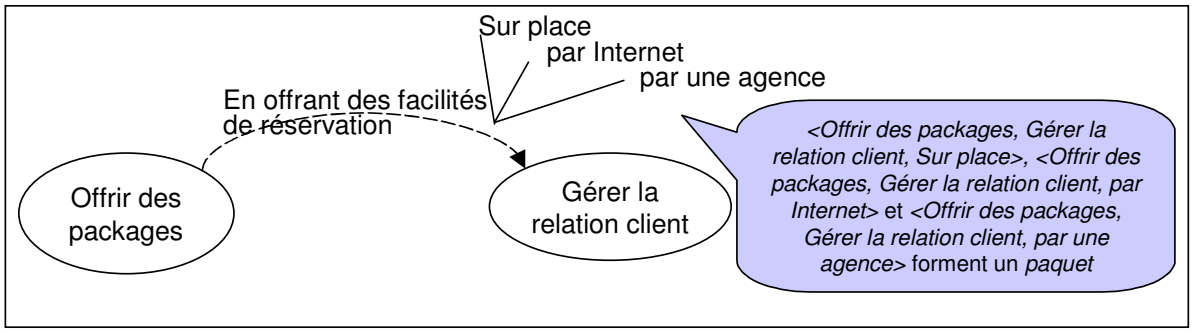


Figure 54 : Exemple de paquet

3.6. Expression de la relation d'alignement dans la carte pivot

La carte pivot présente explicitement la relation d'alignement entre système et processus d'entreprise. Elle est le résultat de la fusion de deux cartes : l'une représentant le système et l'autre les processus d'entreprise. Elle permet ainsi de décrire conjointement le système et les processus au sein d'un même modèle et de palier au problème d'évolution disjointe.

La Figure 55 présente un exemple de carte pivot coloriée.

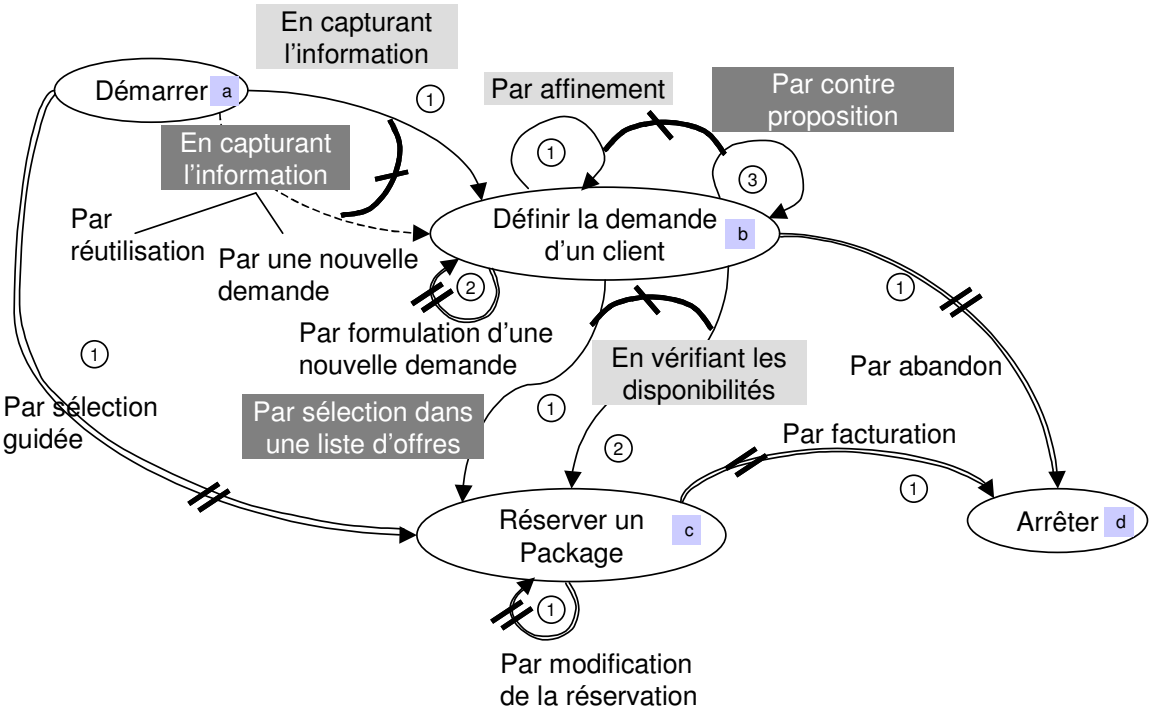


Figure 55 : Exemple de coloration

3.6.1 Principe de coloration

Nous distinguons trois couleurs :

- un élément coloré en blanc traduit la fusion d'un élément d'une carte système et un élément d'une carte processus.

- un élément coloré en gris clair ne peut être considéré que du point de vue système. Il provient de la carte système.
- un élément coloré en gris foncé ne peut être considéré que du point de vue processus. Il provient de la carte processus.

Le méta-modèle (Figure 56) spécifie que seules les intentions et les sections peuvent être colorées. Dans ce dernier cas, d'un point de vue graphique, c'est le nom de la stratégie qui porte la couleur de la section.

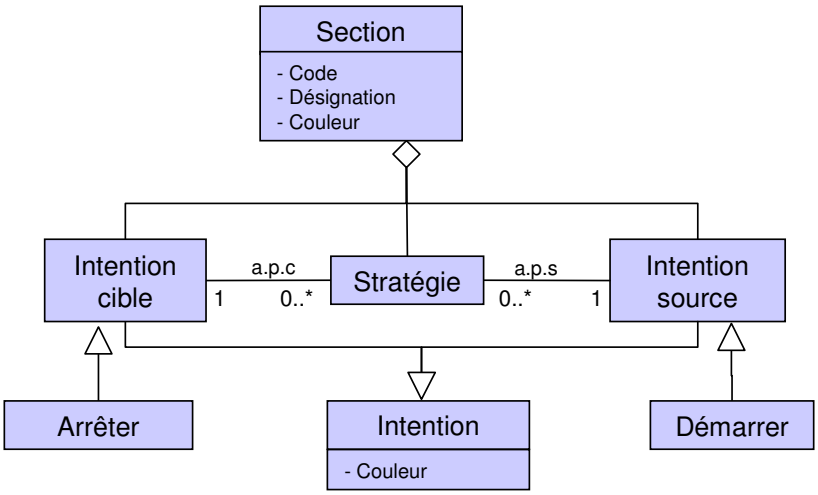


Figure 56 : Coloration des sections et des intentions

Une section peut être colorée en blanc uniquement si son intention source et son intention cible sont blanches. D'un point de vue graphique, la stratégie de la section est représentée avec une flèche double permettant de préciser que la section peut être vue comme une section système et une section processus. En revanche, une section peut être colorée en gris clair (respectivement en gris foncé) si son intention source et son intention cible sont colorées en blanc ou en gris clair (respectivement en gris foncé).

A la Figure 55, les quatre intentions sont colorées en blanc. Cinq sections sont colorées en blanc, 3 en gris clair et 3 en gris foncé. Par exemple, la section <*Définir la demande d'un client, Définir la demande d'un client*, *Par affinement*> est colorée en gris clair. Elle n'est présente que dans la carte système. Elle signifie que des fonctionnalités du système permettent d'affiner une demande d'un client afin de la préciser et réduire le nombre de produits y répondant. Cette fonctionnalité n'est pas prévue dans les processus d'entreprise. Cette section n'existe donc pas dans la carte des processus mais uniquement dans la carte système. Cependant, la couleur blanche de l'intention *Définir la demande d'un client* permet de préciser que cette intention existe et peut être atteinte aussi bien dans le système que dans les processus.

L'utilisation de couleurs permet de visualiser les relations d'alignement entre sections et de préciser le degré d'alignement qui existe entre elles.

3.6.2 Relation d'alignement

La relation d'alignement (Figure 57) est définie entre (i) des sections représentant le point de vue système et (ii) des sections représentant le point de vue processus. Elle est caractérisée

par la propriété “degré” qui peut prendre trois valeurs : *totalement aligné*, *partiellement aligné* et *pas du tout aligné*. De plus, le méta-modèle précise que la couverture est complète (“complete”). En effet, toute section représente soit le système (“section système”), soit les processus (“section processus”).

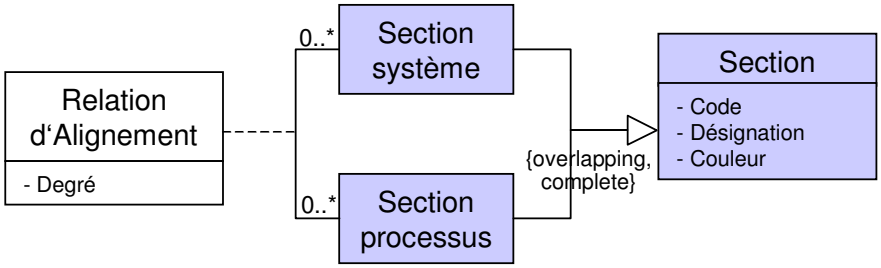


Figure 57 : Expression de la relation d’alignement

On distingue donc trois cas d’alignement :

1. *totalement aligné* : on considère qu’une section système et une section processus sont totalement alignées si elles sont fortement similaires et peuvent représenter deux points de vue : celui du système et celui des processus. Ceci explique que l’héritage multiple (“overlapping”) est possible : une même section pouvant être une section représentant le système et une section représentant les processus. Graphiquement, on précise le degré d’alignement (totalement aligné) en ajoutant le symbole // sur les stratégies de ces sections.

2. *partiellement aligné* : on considère qu’une section système et une section processus d’entreprise sont partiellement alignées si elles possèdent certaines similarités c’est-à-dire que leur intention source et/ou leur intention cible sont similaires (elles sont alors colorées en blanc) mais que leur stratégie sont différentes. On notera qu’une même section du système peut être partiellement alignée avec plusieurs sections des processus. Ceci est précisé dans le méta-modèle par l’utilisation de cardinalités multiples (0...*). Graphiquement, on utilise le symbole ⟩ entre ces sections afin de préciser qu’elles sont partiellement alignées, comme le montre la Figure 58.

3. *pas du tout aligné* : on considère que les sections ne sont pas du tout alignées lorsque ni l’intention source, ni l’intention cible ne sont similaires dans les cartes système et les cartes processus. Ceci signifie, par exemple, qu’une activité des processus n’est pas du tout prise en charge par le système ou qu’à l’inverse, une fonctionnalité du système ne peut être utilisée lors de l’exécution des processus. Graphiquement, ces sections sont marquées d’une croix X.

La Figure 58 résume les différents cas rencontrés.

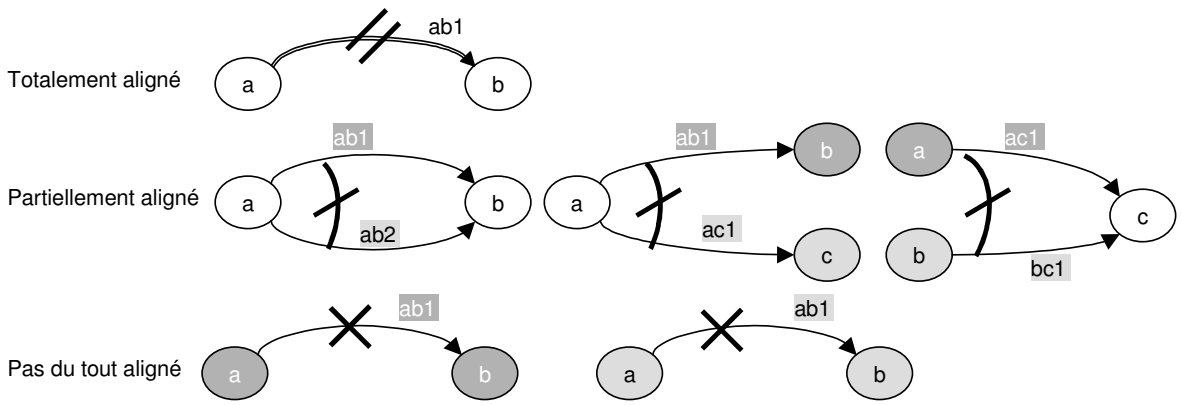


Figure 58 : Les différents cas de coloration

Ainsi par exemple, à la Figure 55, les sections <*Définir la demande d'un client*, *Définir la demande d'un client*, *Par affinement*> et <*Définir la demande d'un client*, *Définir la demande d'un client*, *Par contre proposition*> sont partiellement alignées. En effet, chacune de ces sections proposent de définir une demande à partir d'une première demande déjà formulée par le client. Elles ne sont pas totalement alignées car, dans le premier cas, la demande du client nécessite d'être clarifiée. Dans le second cas, aucune disponibilité ne correspond exactement à la demande du client, mais en la modifiant sensiblement, elle pourrait être satisfaite.

3.7. Lien d'affinement

Le méta-modèle pivot ACEM précise, comme le montre la Figure 59, qu'une section à un niveau *i* peut être affinée par une carte pivot entière à un niveau d'affinement plus élevé *i+1* (c'est-à-dire un niveau d'abstraction moins élevé). L'affinement produit ainsi une structure multi-chemin, multi-segment au niveau i+1. En conséquence, pour une section donnée du niveau i, non seulement : (i) la structure multi-segment décrit une alternative de sous-sections au niveau i+1, mais aussi (ii) la structure multi-chemin introduit plusieurs combinaisons différentes de sous-sections. Ainsi, l'affinement d'une section est une structure plus complexe qu'une simple décomposition ET/OU de buts.

L'affinement est donc défini ici comme un mécanisme d'abstraction complexe selon lequel un assemblage de sections au niveau *i+1* est vu comme une unique section au niveau *i*.

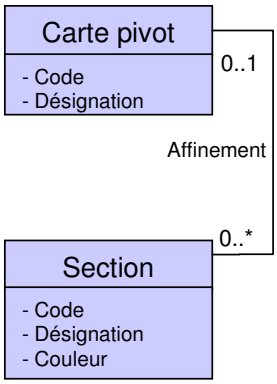


Figure 59: Caractéristique de l'affinement

Une section ne peut être affinée que par une carte pivot au maximum. Il est possible qu'une section ne soit pas affinée, on dit, dans ce cas, qu'elle est opérationnalisable. En revanche, des

sections système et des sections processus totalement ou partiellement alignées peuvent être affinées par la même carte pivot, ce qui se traduit dans le méta-modèle pivot par une cardinalité multiple. De cette façon, chaque carte pivot représente conjointement le système et les processus et permet de préciser la relation d'alignement à plusieurs niveaux.

Le mécanisme d'affinement débouche ainsi sur une modélisation à plusieurs niveaux de détail. Par exemple, dans la carte de la Figure 48, on peut affiner la section <*Offrir des packages, Gérer la relation client, En offrant des facilités de réservation*> par la carte de la Figure 60. Cette carte décrit en détail comment gérer la relation client en offrant des facilités de réservation. Elle est composée de deux intentions en plus de *Démarrer* et *Arrêter*. Ces intentions permettent respectivement de *Définir la demande d'un client* et de *Réserver un package*. Différentes stratégies permettent de naviguer dans cette carte.

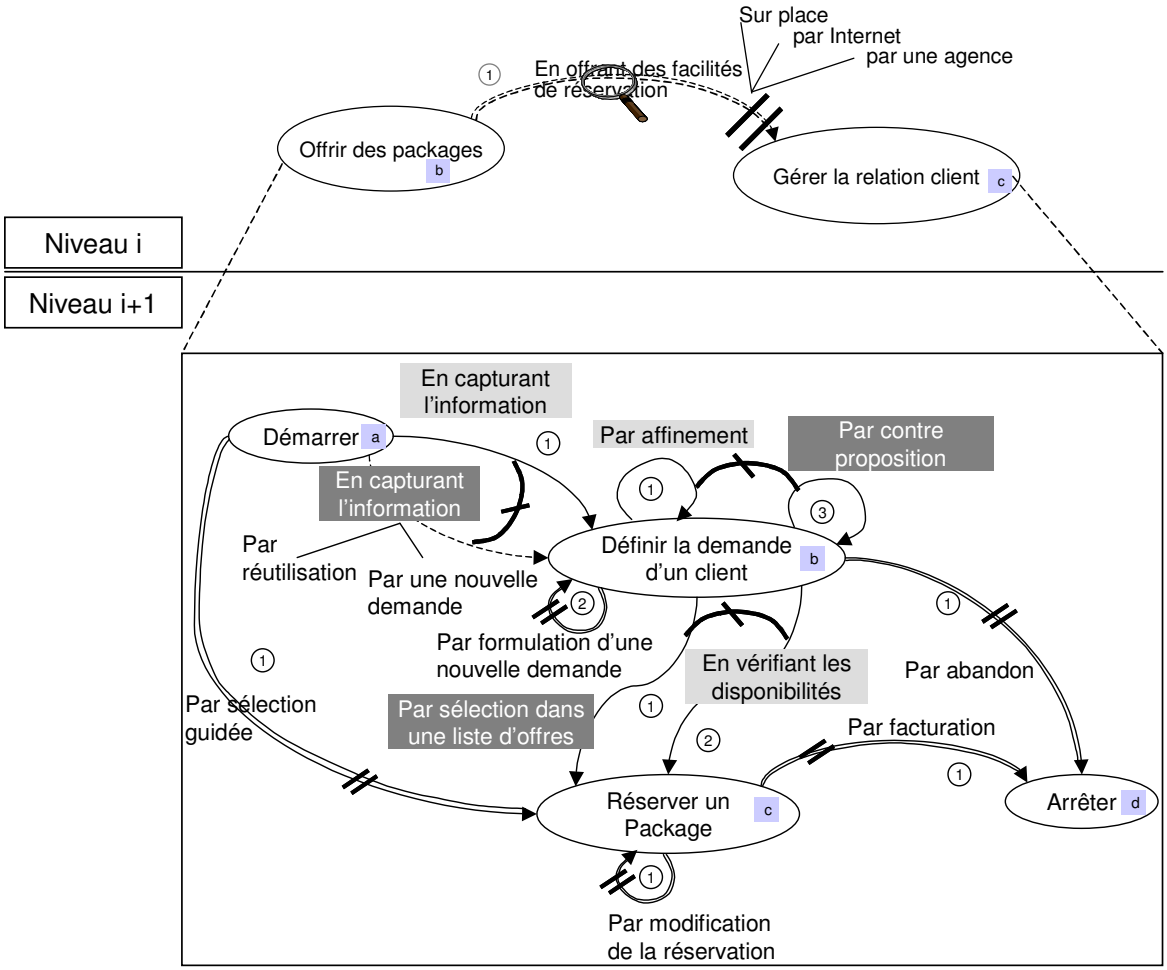


Figure 60 : Affinement de la section <Offrir des packages, Gérer la relation client, En offrant des facilités de réservation>

Cette relation d'affinement entre sections et cartes pivot conduit à un modèle pivot prenant la forme d'une hiérarchie de cartes qui forme le modèle pivot. Celle-ci permet une modélisation des processus et du système à différents niveaux d'abstraction, offrant ainsi une lecture des processus d'entreprise ou du système allant du niveau le plus stratégique au niveau le plus opérationnel.

3.8. Modèle pivot : un ensemble de cartes

Le concept de modèle pivot permet de représenter dans un même schéma l'ensemble des cartes pivot se rapportant à un même objectif. Il se présente sous la forme d'un ensemble de cartes pivot reliées entre elles par des liens d'affinement. Plus précisément, ces liens d'affinement ont pour source une section à un niveau d'abstraction *i* et pour cible une carte pivot entière à un niveau *i+1*. Plusieurs sections dans une même carte pivot pouvant être affinées, une carte 'mère' peut avoir plusieurs 'filles'.

Dans le modèle pivot, il n'y a qu'une seule carte pivot 'racine', elle est la mère de toutes les autres. Elle correspond à la carte pivot du plus haut niveau d'abstraction et permet de représenter de façon abstraite les processus et système. A l'autre extrémité, se trouvent les cartes pivot ne pouvant plus être affinées, ce sont des cartes pivot exécutables.

Le modèle pivot peut rassembler de nombreuses cartes pivot. Afin de faciliter la manipulation de ces cartes pivot sur plusieurs niveaux d'abstraction, il est utile de suivre une codification précise. Il existe deux façons d'appréhender la codification, (i) dans le contexte d'une carte pivot particulière ou (ii) de façon absolue.

Codification locale

Les conventions de codification que nous proposons sont les suivantes :

- chaque intention est codée par une lettre de l'alphabet. Cela permet de manipuler 26 intentions dans une même carte pivot, ce qui est très largement suffisant : notre expérience de projets industriels montre qu'une carte compte rarement plus de dix intentions.
- les stratégies sont numérotées relativement à leurs intentions cible et source. Ainsi, deux stratégies ayant les mêmes intentions source et cible seront respectivement numérotées 1 et 2. Une carte peut donc avoir plusieurs stratégies numérotées de la même façon.
- les sections sont codées par la juxtaposition de trois éléments (i) la lettre de l'intention source, (ii) la lettre de l'intention cible et (iii) le numéro de la stratégie. La section *ab1* permet, en partant de l'intention *a*, d'atteindre l'intention *b* en suivant la stratégie *1*.

La Figure 48 représente un exemple de codification d'une carte, où on a les sections suivantes : *ab1*, *bc1*, *bc2*, *bd1*, *cb1*, *cc1*, *cc2*, *cc3*, *cc4*, *cd1* et *cd2*.

Afin de représenter le modèle pivot et de distinguer les différentes cartes pivot, nous proposons une codification absolue.

Codification absolue

Avec la codification absolue, la carte racine du modèle pivot est appelée C. Ses intentions sont nommées C.a, C.b… et ses sections C.ab1, C.ab2, C.bc1… Si l'une de ces sections (par exemple C.ab1) est affinée par une carte, celle-ci a pour code $C.C_{ab1}$. Ses 3 intentions (a, b et c) et 4 sections (ab1, ab2, bc1 et bc2) sont alors codées de façon absolue comme suit :

- Les intentions : $C.C_{ab1}.a$, $C.C_{ab1}.b$ et $C.C_{ab1}.c$
- Les sections : $C.C_{ab1}.ab1$, $C.C_{ab1}.ab2$, $C.C_{ab1}.bc1$ et $C.C_{ab1}.bc2$

Ainsi, le code $C.C_{ab1}.ab2$ correspond à la section 2 entre l'intention a et l'intention b de la carte C_{ab1}.

Certaines des sections de cette carte peuvent être affinées. On aura par exemple la carte $C.C_{ab1}.C_{.ab2}$ et ainsi de suite.

La Figure 61 présente un exemple de liens entre les différentes cartes pivot constituant le modèle pivot.

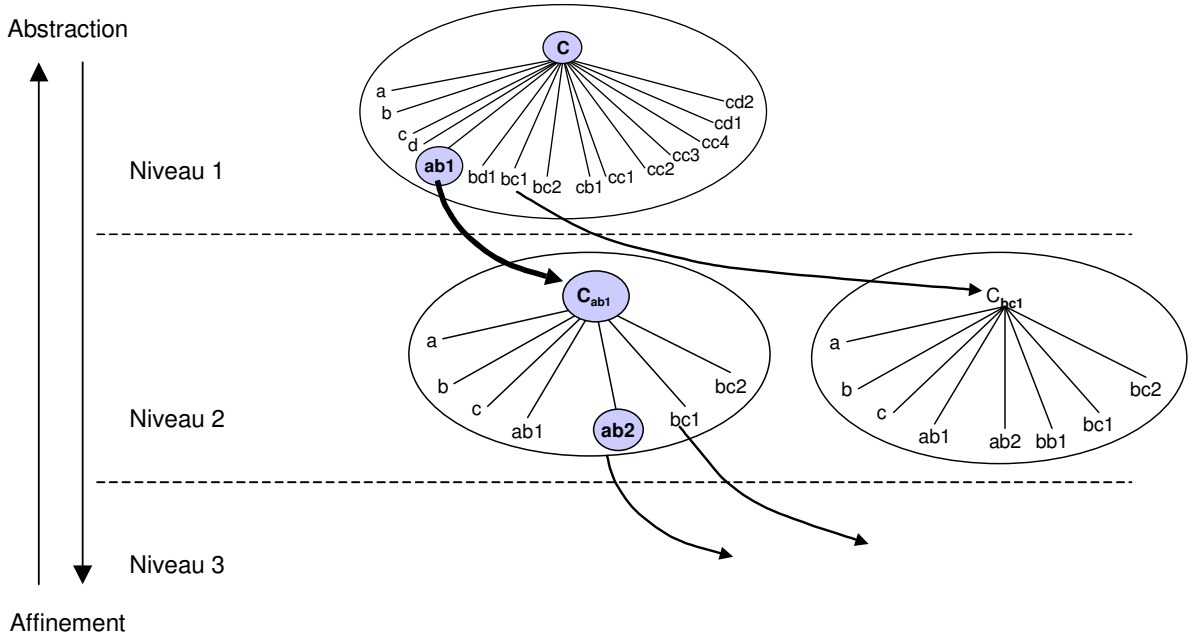


Figure 61 : Exemple de liens entre cartes.

3.9. Invariants et règles de validité du modèle pivot

Pour que le modèle pivot soit correct, il doit respecter un certain nombre d'invariants. Pour qu'il soit valide il doit vérifier des règles de validité. Cette section présente les invariants puis les règles.

3.9.1 Invariants du modèle pivot

Le méta-modèle pivot est accompagné d'un certain nombre de propriétés que doit satisfaire tout modèle pivot. Ces propriétés, en référence à [Banerjee87], sont appelées des invariants et permettent de vérifier qu'un modèle pivot est *correct*.

Nous avons identifié neuf invariants qui portent sur les différents éléments du modèle pivot :

- **I1**. Toute carte pivot a une et une seule intention qui n'est la cible d'aucune stratégie ; l'intention *Démarrer*.
- **I2**. Toute carte pivot a une et une seule intention qui n'est la source d'aucune stratégie ; l'intention *Arrêter*.

- **I3**. Toute intention dans une carte pivot doit être quasi-vivace. Une intention est quasi-vivace si elle peut être atteinte au moins une fois, c'est-à-dire s'il existe un chemin de *Démarrer* à cette intention.
- **I4**. Au sein d'une carte pivot, toute intention est unique.
- **I5**. Toute carte pivot a une et une seule intention qui peut être vue comme un ensemble vide d'états, l'intention *Démarrer*.
- **I6**. Dans un modèle pivot, une section est la source d'au plus un lien d'affinement.
- **I7**. L'ensemble des classes est un graphe connecté.
- **I8**. Toute classe est unique.

De ces neuf invariants, nous avons déduit dix corollaires :

- **C1**. Une carte pivot est un graphe connecté ; il n'y a ni intention ni stratégie isolée.
- **C2**. Toute intention dans une carte pivot est la source d'au moins une stratégie sauf l'intention *Arrêter*.
- **C3**. Toute intention dans une carte pivot est la cible d'au moins une stratégie sauf l'intention *Démarrer*.
- **C4**. Il existe toujours un chemin de *Démarrer* à *Arrêter*.
- **C5**. Toute section appartient à au moins un chemin entre *Démarrer* et *Arrêter*.
- **C6**. Toute intention correspond à un but différent et se voit attribuer un nom différent de celui des autres intentions de la carte pivot.
- **C7**. Toute intention est constituée d'au moins un état sauf l'intention Démarrer.
- **C8**. Toute section a sa pré-condition qui diffère de l'ensemble vide sauf les sections ayant pour source l'intention Démarrer.
- **C9**. Toute intention correspond à un ensemble différent d'états.
- **C10**. Toute carte pivot de niveau *i+1* ($i \neq 0$) est la cible d'un lien d'affinement ayant pour source une section du niveau *i*.
- **C11**. Il n'y a pas de classe ni de relation isolée.

3.9.2 Règles de validité d'une carte

Les règles de validité s'appuient sur la sémantique des éléments de la carte et non plus comme pour les invariants et les corollaires sur la syntaxe. Une carte est valide si elle vérifie les règles suivantes [Rolland01] :

- **R1**. Aucune intention dans la carte ne peut être considérée comme une sous-partie d'une autre intention. Ceci peut s'écrire de façon plus formelle :

$$\forall I_i, \nexists\ Ij, j \neq i\ G_i \subseteq G_j$$

Avec Gi ensemble *d'états désirables* G de I

- **R2**. Aucune stratégie dans la carte ne peut être considérée comme une sous-partie d'une autre stratégie.
- **R3**. Aucune intention ne doit être une façon d'atteindre une autre intention.
- **R4**. Les intentions ayant pour résultat la même partie de produit doivent être agrégées. Ceci peut s'écrire :

$$\forall I_i, \nexists\ I_j, j \neq i\ G_i = G_j$$

- **R5**. Les sections représentant des manières exclusives de produire un même résultat doivent être regroupées au sein d'un paquet.
- **R6**. Les intentions considérées comme faisant partie d'une transaction doivent être abstraites au sein d'une unique intention.
- **R7**. Les intentions qui se complémentent mutuellement et vont ensemble doivent être agrégées au sein d'une unique intention.

4. Conclusion

Le méta-modèle pivot présenté dans ce chapitre s'appuie sur les concepts du méta-modèle de Carte. Ces concepts permettent une représentation intentionnelle du système et des processus. Ainsi, le modèle pivot permet de modéliser dans les mêmes termes le système et les processus. Les concepts du méta-modèle de Carte ont cependant été enrichis en particulier afin de rendre explicite la relation d'alignement et de spécifier les parties totalement, partiellement ou pas du tout alignées. Le mécanisme d'affinement permet d'appréhender la relation d'alignement sur différents niveaux d'abstraction et à en maîtriser sa complexité.

L'utilisation du modèle pivot ainsi que d'opérateurs d'écarts (définis dans le chapitre 7) permet de faire évoluer conjointement le système et/ou les processus ou de réduire la rupture d'alignement entre ces deux entités en fonction des sections sur lesquels ils s'appliquent. Le chapitre 8 aide l'ingénieur d'alignement à construire le modèle pivot et à le faire évoluer.

CHAPITRE 7 : SPECIFICATION DES EXIGENCES D'EVOLUTION

1. Introduction

L'approche ACEM repose sur deux principes clé : (1) l'utilisation d'un modèle pivot pour représenter conjointement les processus d'entreprise et le système, et donc aussi l'alignement entre ces deux entités et (2) la spécification explicite des exigences d'évolution à l'aide d'opérateurs exprimant les écarts entre les situations As-Is et To-Be.

Les modèles As-Is et To-Be instancient le méta-modèle pivot. Les écarts expriment les changements à apporter au modèle As-Is. Ils sont spécifiés par des opérateurs organisés en une typologie associée au méta-modèle pivot.

La définition ad hoc d'une typologie associée à un méta-modèle spécifique peut être source d'erreur, puisqu'elle s'appuie uniquement sur la connaissance du méta-modèle et l'expérience de l'ingénieur. De plus, la typologie résultante est dépendante du formalisme utilisé. Pour surmonter ces difficultés, nous avons défini une typologie générique qui est indépendante du formalisme utilisé pour représenter les modèles As-Is et To-Be. La typologie d'écarts associée au méta-modèle pivot est une instance de la typologie générique d'écarts. D'autres typologies telles qu'une typologie d'écarts associée à un méta-modèle orienté-objet ou à un méta-modèle de but pourraient également être générée.

De nombreuses typologies d'opérateurs ont été proposées dans différents domaines permettant de spécifier les évolutions à mettre en œuvre, en ingénierie des bases de données, pour assurer l'évolution de workflow ou de modèle de processus d'entreprise ou plus récemment pour faire évoluer des DTD XML. La section suivante présente une revue de la littérature sur la problématique des opérateurs d'évolution. La section 3 présente notre approche de spécification des exigences d'évolution ; la section 4 correspond à la description de la typologie générique ; à la section 5, nous présentons le processus permettant de générer une typologie spécifique d'opérateurs d'écarts ; nous illustrons ce processus à la section 6.

2. Etat de l'art sur les opérateurs d'évolution

L'utilisation d'opérateurs pour exprimer les différences entre deux versions d'un même modèle et mettre en œuvre l'évolution est largement répandue, pour différents types de modèles. [Banerjee87], [Andany91], [Zicari92], [Aj-Jadir95], [Estublier00], [Delgado03] s'intéressent à l'évolution des modèles de bases de données, [Al-Jadir03] définit des opérateurs d'évolution pour des DTD XML. [Kradolfer00], [Reichert98], s'intéressent à l'évolution de modèles de Workflow. [Soffer04a] définit un ensemble d'opérateurs de changement pour faire évoluer un modèle de processus. [Mens01] gère l'évolution des logiciels en utilisant des opérateurs de transformation. [Zowghi96] et [Krishna04a] définissent des opérateurs pour faire évoluer des spécifications de besoins.

Les opérateurs sont donc utilisés quelque soit l'aspect du système d'information que l'on souhaite voir évoluer

Dans la suite de cette section, nous présentons six approches d'évolution et analysons les typologies d'opérateurs de changement.

[Banerjee87] définit 21 types de changement supportés par ORION, un méta-modèle de bases de données orientées objet. Ces types de changements sont classés en trois catégories :

1. les changements du contenu d'une classe (ses attributs et ses méthodes). On trouve ainsi dans cette catégorie des types de changement comme 'ajouter une nouvelle variable d'instance', 'supprimer une méthode existante' ou 'changer le nom d'une méthode d'une classe'.

2. les changements sur les liens entre les classes (en l'occurrence uniquement des liens d'héritage). Cette catégorie ne compte que trois types de changement parmi lesquels 'faire de la classe M une super-classe de la classe C'.

3. les changements sur les classes considérées comme un tout. Cette catégorie compte, elle aussi, trois types de changement dont 'supprimer une classe existante'.

Dans chaque catégorie, les types de changement permettent d'ajouter, de supprimer ou de changer un élément.

ORION est le premier système à introduire des invariants et des règles comme moyen de vérifier la correction d'un ensemble d'opérateurs. Après l'application des opérateurs, le modèle doit satisfaire les invariants. Les types de changement définis dans [Banerjee87] vérifient aussi la propriété de complétude, c'est-à-dire qu'ils permettent de mettre en œuvre n'importe quel changement.

[Tamzalit03] s'intéresse également à l'évolution de modèles orientés-objet. Les auteurs considèrent deux structures : le nœud (une classe ou une instance) et la connexion (un lien entre deux nœuds). Les nœuds peuvent être des classes ou des instances. Trois types de connexion sont considérés : (i) entre classes, (ii) entre une classe et une instance et (iii) entre deux instances. Les auteurs définissent 7 types d'opérations d'évolution classés en deux catégories :

- les opérations unaires qui agissent sur un seul nœud ou une seule connexion telles que l'addition, la modification ou la suppression et,

- les opérations binaires appliquées à une structure source pour produire une structure cible. Parmi les opérations binaires, on trouve, le transfert, la fusion, la division et le croisement. Aucune information n'est donnée quant aux propriétés que satisfont les opérations définies.

[Mens00] utilise les graphes pour représenter les différents artefacts des logiciels afin d'être indépendant du domaine d'utilisation. L'auteur utilise la réécriture de graphe pour modéliser

l'évolution et définit 6 primitives de modification : l'ajout, la suppression et le retypage d'un nœud ou d'un lien dans un graphe. Si le graphe est un graphe imbriqué, c'est-à-dire que les nœuds correspondent eux-mêmes à des graphes, alors trois primitives de modification supplémentaires sont nécessaires pour définir tous les types de changement : *Promotion* et *Relégation*, pour respectivement élever ou baisser un nœud dans la hiérarchie d'imbrication, et *Déplacer*, afin de déplacer un nœud imbriqué dans un nouveau nœud parent situé au même niveau que le nœud parent actuel. Enfin, l'auteur précise qu'il est possible de construire des types de primitives de modification composite tel que *RedirigerSource* ou *CréerSuperClasse*. L'utilisation de telles primitives permet de ne pas gérer certains conflits résultant de l'application séparée des primitives qui les composent.

[Soffer04a] définit un modèle de processus comme un quadruplet M_p = <S, L, I, G> où :

- S est un ensemble d'états définissant le domaine du processus,
- L est une loi définie sur S,
- I est un ensemble d'états instables de S considéré comme l'ensemble des états initiaux possibles et
- G est un ensemble d'états stables, l'ensemble des buts.

L'auteur présente une typologie composée de 19 modifications classées en quatre catégories suivant que les modifications s'appliquent à S, L, I ou G. Parmi ces modifications, certaines permettent d'ajouter des éléments, d'en supprimer ou de changer des propriétés. L'impact de chacune des modifications est analysé. Certaines modifications nécessitent l'exécution d'une seconde modification.

[Casati96] s'intéresse à l'évolution des modèles de workflow. Les auteurs définissent un ensemble de 7 primitives d'évolution classées en deux catégories : les primitives de déclaration et les primitives de flux. Les primitives de la première catégorie (*AjouterVariable*, *SupprimerVariable* et *AjouterTâche*) permettent de modifier la déclaration des variables et des tâches. Les autres primitives modifient la structure du flux (*AjouterSuccesseur*, *SupprimerSuccesseur*, *ModifierType* et *ModifierCondition*). Les auteurs considèrent que cet ensemble de primitives vérifie trois propriétés : la complétude, la minimalité et la correction.

[Krishna04a] considère seize catégories de changements qui peuvent avoir lieu dans un modèle i*. Il s'agit respectivement de l'addition et de la suppression des huit éléments suivants : Dépendance, Tache, But, Ressource, But faible (Softgoal), Lien de moyen, Lien de décomposition de tâche et Acteur. Ces changements sont analysés afin de définir leurs impacts respectifs sur un modèle Z.

Ces quelques références montrent que la terminologie relative aux opérateurs est diversifiée (opérateurs de changement, opérateurs d'évolution, transformation, modifications, opérations…). Il en est de même pour les opérateurs eux-mêmes. Dans le domaine de l'évolution des bases de données, par exemple, [Estublier00] remarque que l'évolution est

maîtrisée différemment selon les SGBD. Chaque système a ses propres opérateurs avec leurs propres sémantiques. La disparité des ensembles d'opérateurs rend leur comparaison difficile.

Cependant, malgré de nombreuses différences, ces typologies partagent les points suivants:

1. elles peuvent être structurées selon que les opérateurs modifient la structure, i.e. les relations entre éléments ou les éléments eux-mêmes ;
2. elles permettent généralement d'ajouter et de supprimer un concept, que ce soit un lien, un élément ou une propriété, et
3. elles satisfont un certain nombre de critères de qualité, en particulier la complétude et la correction.

On remarque également que les typologies qui proposent des opérateurs complexes tels que Fusionner ou Diviser sont rares. L'opérateur Changer est néanmoins parfois utilisé dans la mesure où il permet de ne pas perdre d'information, contrairement à la succession Ajouter, Supprimer [Al-Jadir03].

Pour réduire les difficultés liées à la disparité des modèles et des opérateurs qui leurs sont associés, nous proposons de dériver le jeu des opérateurs spécifiques d'un modèle x d'un ensemble générique d'opérateurs attachés à un modèle lui-même générique.

La section suivante présente notre approche et les qualités que doit satisfaire la typologie générique.

3. Approche de spécification des exigences d'évolution

Nous adoptons une approche systématique pour (i) identifier des opérateurs d'écarts associés à un méta-modèle donné satisfaisant un certain nombre de propriétés et (ii) fournir les moyens d'adapter les opérateurs d'écarts à un projet particulier [Rolland04].

3.1. Présentation de l'approche

Comme le montre la Figure 62, l'approche est à trois niveaux : le niveau modèle, le niveau méta-modèle et le niveau méta-modèle générique. Ces trois niveaux correspondent respectivement aux niveaux M1, M2 et M3 de l'OMG [OMG], [Breton02].

Au *niveau modèle*, sont définis les modèles (aussi désignés produits ou schémas) avant et après évolution. C'est à ce niveau qu'est défini le modèle pivot représentant les processus d'entreprise et les fonctionnalités du système. C'est aussi à ce niveau que sont définies les exigences d'évolution (représentées à la Figure 62 par la lettre grecque Δ), sous la forme d'écarts exprimant les évolutions à mettre en œuvre entre les deux modèles. Dans la mesure où, dans cette thèse, les modèles As-Is et To-Be s'expriment dans le langage pivot, nous faisons l'hypothèse que les deux modèles As-Is et To-Be sont définis avec le même langage. Nous ne nous intéressons donc pas, comme dans [Terrasse03] et [Bezivin01], aux évolutions dans les cas où les modèles As-Is et To-Be sont des instances de deux méta-modèles différents.

Le *niveau méta-modèle* contient la spécification d'un méta-modèle spécifique et la spécification d'une typologie d'écarts associée. Le méta-modèle spécifique indique le type

des éléments utilisés dans les modèles As-Is et To-Be ; ces derniers étant des instances du méta-modèle spécifique. De la même manière, la typologie spécifique d'écarts spécifie le type des opérateurs d'écarts définis au niveau modèle. Les écarts identifiés entre le modèle As-Is et le modèle To-Be sont des instances de la typologie spécifique d'écarts.

Le *niveau méta-modèle générique* propose une typologie générique d'écarts et un méta-modèle générique à partir desquels sont respectivement définis la typologie spécifique et le méta-modèle spécifique. Le méta-modèle générique identifie les concepts génériques nécessaires à la définition d'opérateurs génériques rassemblés au sein de la typologie générique d'opérateurs d'écarts. Le méta-modèle générique permet, par instanciation, de rendre explicites les éléments et la structure des méta-modèles spécifiques.

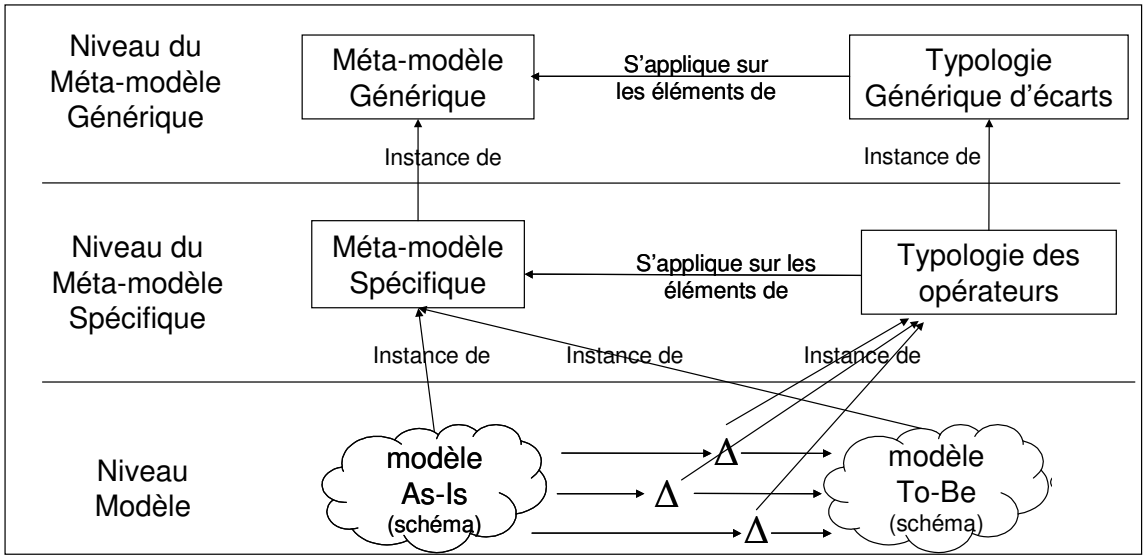


Figure 62 : Aperçu de l'approche

Nous pensons que le développement, de façon ad-hoc, d'une typologie d'opérateurs d'écarts pour chaque projet est source d'erreurs car :

- il repose sur la connaissance et le savoir-faire de quelques personnes,
- il n'est pas systématique et
- il peut être influencé par le contexte du projet.

C'est pourquoi nous avons choisi de dériver la typologie des opérateurs spécifiques d'un méta-modèle spécifique. Dans notre approche de co-évolution, le modèle pivot instancie le méta-modèle pivot. Nous aurions donc pu construire une typologie d'écarts associée à ce méta-modèle spécifique. Cependant cette typologie aurait été dépendante du formalisme utilisé dans notre approche alors que le principe d'évolution par écarts peut être utilisé dans un autre contexte que celui de la co-évolution, par exemple pour faire évoluer un modèle de base de données ou un modèle de workflow. Il peut donc être nécessaire de construire une typologie d'écarts associée au méta-modèle choisi (Carte, Cas d'utilisation, Entité-Relation, méta-modèle de workflow…) pour représenter les situations As-Is et To-Be. Cependant, la spécification au niveau méta-modèle d'une typologie d'écarts n'est pas systématique et reste souvent influencée par le projet. De plus, comme le montre l'état de l'art de la section 2, si chaque typologie est construite de façon spécifique et indépendante, il est difficile de les comparer car elles dépendent chacune d'un méta-modèle spécifique différent.

L'utilisation d'une typologie générique relative à un méta-modèle générique permet de résoudre ces problèmes en étant indépendant de tout projet et de tout méta-modèle. Le méta-modèle générique peut être instancié par chacun des méta-modèles utilisés. La typologie générique associée au méta-modèle générique est adaptée pour correspondre à chacun des méta-modèles spécifiques. Une telle approche permet d'identifier systématiquement les aspects sémantiques et structurels susceptibles de faire l'objet d'un écart sur les différents types d'éléments qui composent le méta-modèle spécifique.

Par exemple, si on utilise le méta-modèle Entité-Relation pour représenter le modèle d'une base de données, alors les écarts seront exprimés au niveau modèle entre deux modèles Entité-Relation. Les écarts entre les modèles As-Is et To-Be, expriment ce qui change ou doit être adapté entre les deux situations. Ils instancient les opérateurs de la typologie spécifique. Ils peuvent par exemple permettre d'exprimer que l'entité type Réservation doit être éclatée en deux entités types Réservation et Demande et que la relation type 'correspond' qui a pour source Réservation et pour cible Demande doit être ajoutée. Ces types d'écart sont définis avec l'opérateur associé par adaptation de la typologie générique d'écarts en conformité avec le méta-modèle spécifique Entité-Relation.

3.2. Qualités attendues de la typologie d'écarts

Comme nous l'avons vu dans l'état de l'art de la section 2, beaucoup d'auteurs cherchent à définir une typologie qui soit complète et correcte. Il est donc indispensable que notre approche permette de construire des typologies qui satisfont ces deux propriétés. D'autres propriétés comme la *consistance*, la *minimalité* et *la richesse sémantique* [Teeuw97] ou [Casati96] sont également considérées comme des critères de qualité. Nous pensons donc qu'une typologie d'écarts doit être complète, correcte, consistante et soit sémantiquement riche soit minimale [Etien03a].

Un ensemble d'opérateurs est considéré comme *complet* s'il subsume tout type de changement possible. En d'autres termes, un ensemble d'opérateurs est complet si n'importe quel modèle peut être dérivé de n'importe quel autre modèle [Kradolfer00].

Selon Banerjee [Banerjee87], un ensemble d'opérateurs est *correct* si chacun de ses éléments est correct. On dit qu'un opérateur est correct s'il n'aboutit pas à un modèle incorrect. Dans [Banerjee87], la correction d'un modèle est définie par un ensemble d'invariants qui sont des conditions sur le modèle. Les invariants doivent être satisfaits avant et après toute modification du modèle.

Selon Teeuw [Teeuw97], un ensemble d'opérateurs est *consistant* si les définitions des différents opérateurs n'entrent pas en conflit les unes par rapport aux autres. Des définitions sont en conflit si un même écart ou changement peut être interprété comme se référant à plusieurs opérateurs différents.

Un ensemble d'opérateurs est considéré comme *sémantiquement riche* si chaque type de changement peut être exprimé en n'utilisant qu'un seul opérateur sans avoir à combiner plusieurs opérateurs. Des opérateurs sémantiquement riches permettent de ne pas perdre de l'information. En effet, si un élément est supprimé puis qu'un autre relativement semblable est ajouté cela ne revient pas exactement au même que modifier le premier élément car, dans le premier cas, l'information contenue dans l'élément est perdue [Al-Jadir03]. De plus, des

opérateurs sémantiquement riches permettent d'être plus proche des besoins des utilisateurs en associant un seul opérateur et non une suite d'opérateurs à un type de changement.

La *minimalité* fait référence à l'obtention de la propriété de complétude par un ensemble minimal d'opérateurs. En d'autres termes, un ensemble d'opérateurs est minimal s'il est complet et s'il ne contient aucun opérateur qui puisse être obtenu par composition d'autres opérateurs [Casati96].

Il est clair que ces deux dernières propriétés ne peuvent pas être satisfaites par le même ensemble d'opérateurs dans la mesure où elles se contredisent. Nous proposons donc de préciser, au sein des différentes typologies, les opérateurs constituant l'ensemble minimal, mais nous spécifions également d'autres opérateurs qui permettent à la typologie d'être sémantiquement riche.

4. La typologie générique

Afin d'aider les ingénieurs à définir une typologie d'opérateurs d'écarts de façon systématique et indépendante d'un projet ou d'un méta-modèle particulier, nous proposons une typologie générique d'écarts. Celle-ci est indépendante du formalisme utilisé pour représenter les modèles As-Is et To-Be. Elle prend la forme d'un ensemble d'opérateurs applicables à des éléments génériques qui composent tout méta-modèle. Pour cela, il est nécessaire d'abstraire les spécificités de chaque méta-modèle particulier (comme le méta-modèle pivot) pour généraliser les éléments des méta-modèles et leurs relations.

Afin de construire une typologie générique d'écarts nous développons donc d'abord un méta-modèle générique permettant de mettre en évidence les éléments et la structure de tout méta-modèle.

4.1. Méta-modèle générique

De nombreux auteurs ont essayé de rendre explicites les éléments qui composent tout méta-modèle c'est-à-dire de définir un méta-méta-modèle [IRDS90], [Martiin94], [Grundy96], [Plihon97], [Prakash99]. Il existe différents méta-méta-modèles qui varient selon leur objectif. Ainsi, par exemple, IRDS [IRDS90] est un standard qui permet de faciliter l'évolution des représentations de modèles dans les outils CASE. Prakash [Prakash99] propose une définition formelle d'une méthode. Martiin [Martiin94] cherche une structure générique d'entrepôt d'environnements meta-Case. Plus récemment, les travaux de l'OMG [OMG] sur le MOF et UML cherchent à définir des standards pour représenter de façon générique différents aspects des systèmes d'information, mais aussi pour faciliter l'utilisation et l'évolution de modèles.

Le méta-modèle générique, que nous proposons, a pour but d'aider à définir les transformations élémentaires qui peuvent intervenir sur les éléments d'un méta-modèle. Il permet d'identifier les éléments clé et la structure de n'importe quel méta-modèle ayant une représentation graphique.

Ce méta-modèle générique est présenté à la Figure 63 en utilisant les notations UML. Il considère que tout méta-modèle est constitué d'*Eléments*, ayant un *Nom* et un *Type*, et étant caractérisés par un ensemble de *Propriétés*. Dans le méta-modèle E/R, par exemple, une

Entité Type, un *Attribut* et une *Relation Type*, ainsi que la relation *Is-A* sont des *éléments*. *Domaine* est une *propriété* d'*Attribut*.

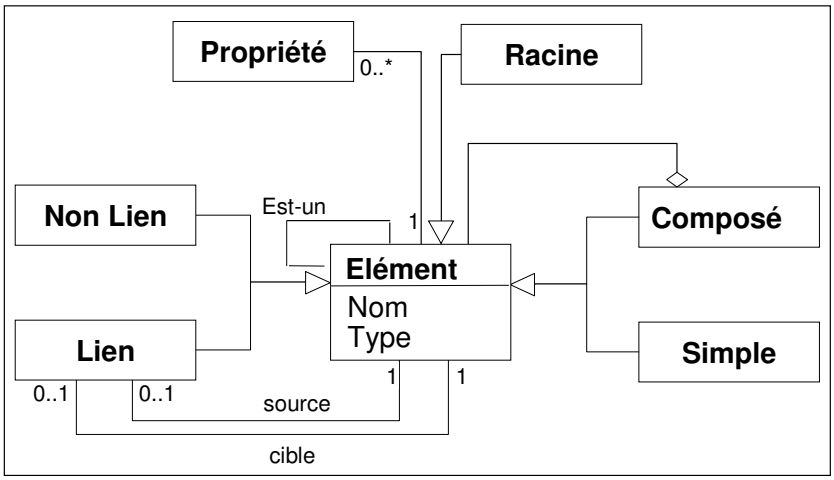


Figure 63 : Méta-modèle générique pour définir une typologie d'écarts

Il existe deux façons orthogonales de classer les *Eléments*. La première classification fait la distinction entre éléments *Simples* et *Composés*. Les *Eléments Composés* sont décomposables en éléments plus fins (qui peuvent être simples ou à leur tour composés). Les *Eléments Simples*, quant à eux, ne sont pas décomposables en d'autres *Eléments*. Par exemple, dans le méta-modèle E/R, une *Entité Type* est un *élément composé* fait d'*Attributs* qui sont des *éléments simples*.

La seconde classification différencie les éléments *Liens* et *Non liens*. Un *Elément Lien* est un connecteur entre deux éléments, l'un jouant le rôle de *Source* et l'autre celui de *Cible*. Les éléments qui ne sont pas des liens sont appelés *Non liens*. Une relation *Is-A* du méta-modèle E/R est un *Lien* : il connecte une *Entité Type* source à une *Entité Type* cible. A l'opposé, une *Entité Type* est un élément *Non lien*.

Ce méta-modèle spécifie également qu'un élément *Est-un* autre élément c'est-à-dire qu'il peut hériter d'un autre élément.

Enfin, tout modèle est un élément composé qui peut être réduit à l'élément *Racine* quand on a supprimé tous les autres éléments. L'élément *Racine* correspond, par exemple, à la classe Objet dans une hiérarchie de classes.

4.2. Typologie générique d'écarts

La typologie générique est un ensemble d'opérateurs qui expriment les types d'écarts s'appliquant sur les différents concepts du méta-modèle générique.

Avant de définir formellement les opérateurs génériques, nous les présentons en fonction du type de changement qu'ils permettent de réaliser et du concept sur lequel ils s'appliquent.

4.2.1 Présentation de la typologie générique d'opérateurs d'écarts

La typologie générique d'écarts est composée d'un ensemble d'opérateurs. Chaque opérateur identifie un type de changement qui peut être réalisé sur un élément d'un modèle As-Is. L'opérateur identifie la différence entre le modèle As-Is et le modèle To-Be. Par exemple,

Renommer est un opérateur, *Renommer Elément* sera un changement qui caractérise la transformation d'un élément du As-Is dans le modèle To-Be.

Les deux sous-sections suivantes présentent les quatorze opérateurs de la typologie générique d'écarts selon, respectivement, le type de changement qu'ils permettent de réaliser et le concept du méta-modèle générique sur lequel ils s'appliquent.

4.2.1.1 Trois types de changement

La typologie générique d'écarts identifie trois types majeurs de changements : les changements de *nommage*, les changements *intrinsèques* et les changements *structurels*.

Les changements de *nommage* affectent seulement la façon dont les organisations veulent se référer à un élément. Ce type ne compte qu'un seul opérateur générique *Renommer*.

Les changements *intrinsèques* affectent les éléments en se limitant aux éléments eux-mêmes sans tenir compte de leur relation avec d'autres éléments : modifier le *domaine* d'un *Attribut* est un exemple d'un tel changement localisé. Le Tableau 20 propose quatre opérateurs pour spécifier les changements au sein même d'un élément, à savoir, *Modifier*, *Joindre*, *Enlever*, qui concernent les propriétés des éléments et *Retyper* qui change le type d'un élément et permet, par exemple, de transformer une *Relation Type* du As-Is en *Entité Type* du To-Be.

Les changements *structurels* font intervenir plusieurs éléments et correspondent, non plus à des transformations concernant un élément unique, comme c'est le cas des changements intrinsèques, mais à des modifications de plusieurs, voire de tous les éléments qui composent le méta-modèle. Il y a neuf opérateurs pour spécifier ce type de changement : *Ajouter*, *Supprimer*, *Remplacer*, *Fusionner*, *Diviser*, *ChangerOrigine*, *AjouterComposant*, *SupprimerComposant* et *DéplacerComposant*. Par exemple, ajouter ou supprimer des relations types ou des entités types dans un modèle E/R As-Is pour construire le modèle To-Be sont des changements structurels. *AjouterComposant* consiste à ajouter un élément dans un élément composé autre que le modèle lui-même. Ainsi, pour ajouter un attribut dans une entité type, il faudra utiliser l'opérateur *AjouterComposant* dans la mesure où, comme nous l'avons vu précédemment, une Entité Type est un *Elément* composé d'Attributs.

Le Tableau 20 résume la typologie générique d'écarts composée de 14 opérateurs.

Type	Opérateur	Description
Nommage	Renommer	Changer le nom d'un élément dans le modèle To-Be
Intrinsèque	Modifier	Changer la propriété d'un élément du To-Be
	Joindre	Ajouter une propriété à l'élément du To-Be
	Enlever	Supprimer une propriété à l'élément du To-Be
	Retyper	Les éléments du As-Is et du To-Be ont des types différents
Structurel	Ajouter	Ajouter un élément dans le modèle To-Be
	Supprimer	Supprimer un élément du As-Is dans le modèle To-Be
	Remplacer	Un élément du As-Is est remplacé par un autre élément
	Fusionner	Deux éléments du As-Is sont remplacés par un unique élément
	Diviser	Un élément du As-Is est décomposé en deux éléments
	ChangerOrigine	La source ou la cible d'un lien est modifiée
	AjouterComposant	Un composant est ajouté dans l'élément du To-Be
	SupprimerComposant	Un composant du As-Is est supprimé de l'élément To-Be
	DéplacerComposant	Un composant est repositionné dans la structure de l'élément To-Be

Tableau 20: Typologie générique d'écarts

Il est également possible de classer ces opérateurs selon les concepts du méta-modèle générique sur lesquels ils s'appliquent.

4.2.1.2 Classification en fonction des concepts génériques

Il est important de remarquer que tous les opérateurs ne s'appliquent pas à tous les types d'éléments, c'est-à-dire à un *Elément* mais aussi à toutes ses spécialisations. Ainsi, les opérateurs *Renommer*, *Retyper*, *Ajouter*, *Supprimer*, *Remplacer*, *Fusionner* et *Diviser* s'appliquent à n'importe quel type d'élément et donc aussi bien aux *Liens*, *Non Liens*, *Eléments Composés* et *Simples*. L'opérateur *ChangerOrigine* ne s'applique qu'aux *Liens*. En effet, cet opérateur permet de modifier l'élément *source* ou l'élément *cible* d'un élément *Lien*. *Modifier*, *Joindre* et *Enlever* s'applique uniquement aux *Propriétés*. Enfin, les opérateurs *AjouterComposant*, *SupprimerComposant* et *DéplacerComposant* ne concernent que les *Eléments Composés* et permettent de changer leurs structures en ajoutant, supprimant ou déplaçant des composants. La Figure 64 montre sur un même modèle le méta-modèle générique et les opérateurs génériques (en gris clair).

L'élément *Racine* jouant un rôle particulier, tous les opérateurs s'appliquant à *Elément* ne s'appliquent pas forcément à la *Racine*. En effet, il est seulement possible d'ajouter et de supprimer l'élément *Racine*. Cela correspond respectivement à la première activité lors de la création d'un modèle et à la dernière activité lors de la destruction du modèle. Cet élément ayant un sens universel (comme la classe objet dans une hiérarchie de classe) n'est présent que pour des raisons techniques, comme nous le verrons dans la suite de ce chapitre.

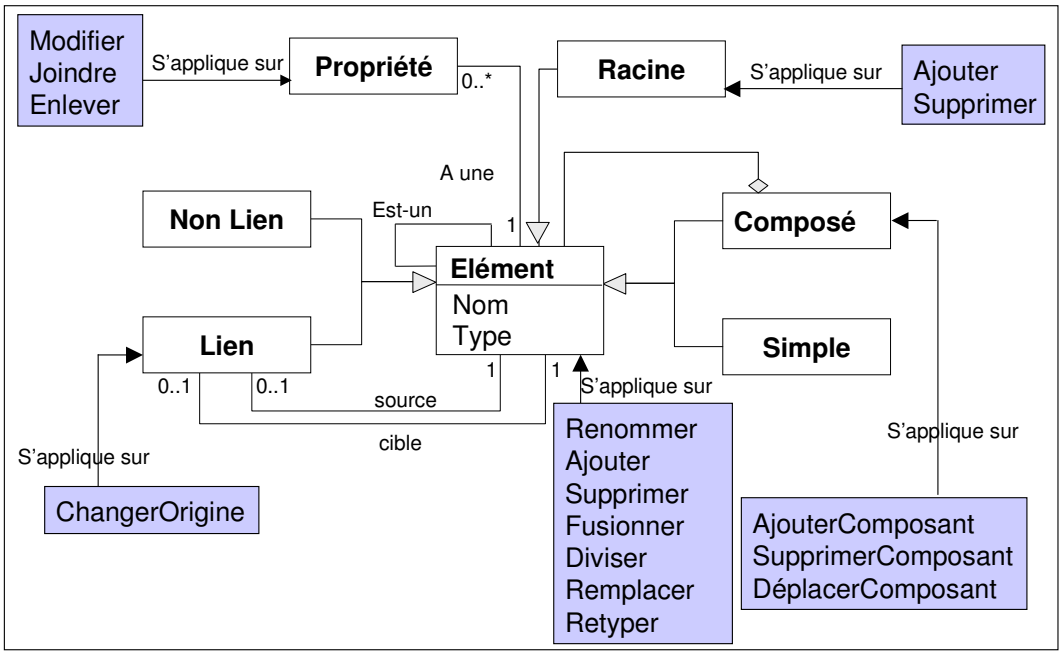


Figure 64 : Typologie générique d'écarts en fonction du type d'élément sur lequel l'opérateur s'applique

4.2.2 Définition formelle des opérateurs génériques

Dans cette section, nous définissons formellement les quatorze opérateurs de la typologie générique d'écarts en suivant la même structure. Celle-ci est présentée dans la première sous-section. Les opérateurs génériques sont définis dans la sous-section suivante.

4.2.2.1 Structure d'un opérateur

La définition des opérateurs repose sur deux concepts : une *signature* et un *prédicat*.

La *signature* identifie le type des éléments impliqués dans les modèles As-Is (c'est-à-dire avant l'exécution de l'opérateur) et To-Be (c'est-à-dire après l'exécution de l'opérateur).

Le *prédicat* est composé de deux éléments : une expression de la logique du premier ordre et éventuellement des paramètres. *L'expression* n'indique pas comment modifier le modèle As-Is mais spécifie les conditions que doit satisfaire le modèle To-Be. Elle repose sur les concepts du méta-modèle spécifique ; un concept étant un *Elément* ou une *Propriété*. Un *paramètre* fait référence à un concept.

La Figure 65 représente la structure d'un opérateur en utilisant la notation UML.

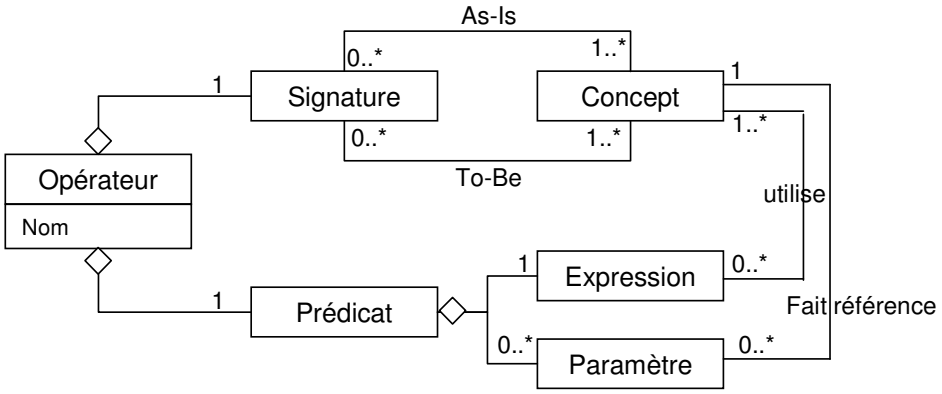


Figure 65 : Modèle d'opérateur

Afin de tenir compte des concepts du méta-modèle générique et des liens qui existent entre eux, l'expression du prédicat utilise les fonctions suivantes :

- a() : qui s'applique à un *Elément* et prend en paramètre une *Propriété*. Cette fonction permet de préciser qu'un *Elément* a une *Propriété*, ce qui s'écrit E.a(P) avec E, un *Elément* et P, une *Propriété*.
- nom() : qui s'applique à un *Elément* et prend en paramètre un *String* c'est-à-dire une chaîne de caractères. E.nom(N) signifie que *l'Elément* E a pour *Nom* la chaîne de caractères N.
- type() qui s'applique à un *Elément* et prend en paramètre un *Type*. E.type(T) spécifie que la valeur de l'attribut *Type* de *l'Elément* E est T.
- a-pour-source() : qui s'applique à un *Lien* et qui prend en paramètre un *Elément*. Cette fonction permet de préciser *l'Elément* qui est la source d'un élément *Lien*. On écrit par exemple L.a-pour-source(E) où L est un *Lien* et E un *Elément*.
- a-pour-cible() : cette fonction est construite exactement comme la précédente. Elle permet de préciser l'*Elément* cible d'un élément *Lien*.
- est-composé-de() : qui s'applique à un élément *Composé* et prend en paramètre un *Elément*. Cette fonction permet de spécifier que ce dernier est un élément qui compose l'élément *Composé*. E_1.est-composé-de(E_2) spécifie que *l'Elément* E_1 est composé de *l'Elément* E_2.

A partir de cette structure et de ces fonctions nous définissons formellement, dans la suite, les quatorze opérateurs de la typologie générique présentés au le Tableau 20.

4.2.2.2 Définition des opérateurs

Nous avons dissocié deux types d'opérateurs : ceux correspondant à l'ensemble minimal (*Ajouter*, *Supprimer*, *Joindre*, *Enlever*, *AjouterComposant* et *SupprimerComposant*) et ceux qui permettent à la typologie générique d'être sémantiquement riche (*Renommer*, *Modifier*, *Retyper*, *Remplacer*, *Fusionner*, *Diviser*, *ChangerOrigine* et *DéplacerComposant*). Tous les opérateurs sont présentés en appliquant un cadre contenant les parties suivantes :

- Motivation : cette partie est une présentation informelle de l'opérateur en précisant à quoi il sert, quelles sont les motivations qui justifient son utilisation, quel est son objectif, etc.
- Définition : dans cette partie nous proposons une définition formelle de l'opérateur, qui suit le modèle de la Figure 65 et utilise les fonctions présentées précédemment.
- Description : cette partie correspond à une définition informelle de l'opérateur.
- Exemple : dans cette partie nous proposons un exemple d'application de l'opérateur.

Les six premiers opérateurs qui suivent constituent l'ensemble minimal des opérateurs génériques.

Ajouter

Motivation : Afin de tenir compte de nouveaux besoins dans le modèle To-Be, il peut parfois être nécessaire de créer un nouvel élément. L'opérateur *Ajouter* permet de créer un nouvel *Elément* dans le modèle As-Is pour construire le modèle To-Be.

Définition : L'opérateur *Ajouter* se définit de façon différente suivant que l'élément à ajouter est un élément *Lien* ou un élément *Non Lien*.

$$\text{AjouterLien} : \text{NonLien}^2 \rightarrow \text{Lien}, \text{NonLien}^2$$
$$\text{AjouterLien}(NL_1, NL_2) = L \in M \wedge \text{L.a-pour-source}(NL_1) \wedge \text{L.a-pour-cible}(NL_2) \mid L \in \text{Lien}, NL_1, NL_2 \in \text{NonLien}, M \in \text{Modèle}$$

$$\text{AjouterNonLien} : \text{Modèle} \rightarrow \text{NonLien}$$
$$\text{AjouterNonLien}(M) = NL \in M \mid NL \in \text{NonLien}, M \in \text{Modèle}$$

Description : L'opérateur *AjouterLien* permet d'ajouter un *Lien* L entre deux Eléments *Non Lien* NL_1 et NL_2. Après application de l'opérateur, dans le modèle To-Be, L est un élément du Modèle M. L a pour source l'élément NL_1 et pour cible l'élément NL_2.

L'opérateur *AjouterNonLien* permet d'ajouter l'élément *Non Lien* NL dans le Modèle M. Après l'application de l'opérateur, NL appartient au Modèle M.

Le modèle est toujours présent avant et après application de l'opérateur. Il apparaît comme élément dans la signature uniquement lorsque c'est le seul élément permettant de spécifier la situation As-Is ou la situation To-Be, comme c'est le cas pour l'opérateur *AjouterNonLien*.

Exemple : Dans le cas de gestion de réservations de chambre d'hôtel, il est par exemple possible d'utiliser l'opérateur *AjouterNonLien* pour ajouter l'entité type Chambre, afin de pouvoir considérer, dans le modèle Entité-Relation To-Be, la Chambre comme un concept à part entière.

L'opérateur *AjouterLien* peut servir à ajouter un *Lien* entre les entités types Demande et Réservation afin qu'il soit possible dans le système To-Be de préciser, à quelle Demande fait référence une Réservation et de ne plus supprimer les Demandes comme dans la situation As-Is.

Supprimer

Motivation : Lors de l'évolution du modèle As-Is, il est possible de se rendre compte qu'un élément *Lien* ou *Non Lien* est devenu inutile pour différentes raisons : (i) un nouveau concept ajouté dans le modèle To-Be le remplace, (ii) il existe un autre élément ayant la même sémantique mais de structure plus riche qui peut le remplacer, (iii) cet élément n'est plus utilisé dans le modèle To-Be. Il faut donc supprimer cet *Elément* grâce à l'opérateur *Supprimer*.

Définition :

Supprimer : Elément $\rightarrow$ Modèle
Supprimer (E) = E $\notin$ M | M $\in$ Modèle, E $\in$ Elément

Description : L'opérateur générique *Supprimer* est défini de la même façon que l'élément à supprimer soit un *Lien* ou un *Non Lien*. La définition de l'opérateur permet de préciser que *l'Elément* qui existe dans le modèle As-Is, n'appartient pas au modèle To-Be après application de l'opérateur.

Exemple : En considérant que le processus de gestion de réservation de chambres d'hôtel est modélisé en utilisant le diagramme d'activités UML, l'opérateur *Supprimer* permet par exemple de supprimer l'activité 'Gestion des demandes en attente'. Cette activité est estimée trop coûteuse et source d'ennuis avec le client. Elle n'apparaît donc plus dans le modèle To-Be.

Joindre

Motivation : L'addition de nouvelles propriétés permet d'enrichir *l'Elément* auquel on les attache, de modifier sa structure ou de l'adapter.

Définition :

Joindre : Elément $\rightarrow$ Elément, Propriété
Joindre (E) = E.a(P) | P $\in$ Propriété, E $\in$ Elément

Description : L'opérateur *Joindre* permet d'ajouter une propriété à un élément. L'élément existe aussi bien dans le modèle As-Is que dans le modèle To-Be. En revanche la propriété n'existe que dans la situation To-Be. La fonction a() permet de préciser à quel élément la propriété appartient.

Exemple : L'opérateur *Joindre* permet par exemple d'ajouter la pré-condition 'date du jour > date de fin de la réservation' à la section <Gérer Relation Client, Gérer Relation Client, En récompensant le client> de la carte 'Gestion des réservations'.

Enlever

Motivation : L'opérateur *Enlever* permet de supprimer une propriété P d'un *Elément* E. Il peut, par exemple, servir à supprimer une propriété qui n'a plus de raison d'être dans la situation To-Be.

Définition :

Enlever : Elément, Propriété → Elément
Enlever (E, P) = ¬ E.a(P) | P ∈ Propriété, E ∈ Elément

Description : L'opérateur *Enlever* permet de supprimer une propriété d'un *Elément*. Sa définition précise que dans le modèle As-Is, *l'Elément* E possédait la propriété P, mais que dans la situation To-Be, cela n'est plus le cas.

Exemple : Il est, par exemple, possible de supprimer la pré-condition 'réservation facturée' à la section <Gérer les réservations, Terminer, Par paiement>. Ceci permet au client, dans la carte To-Be, de payer sa réservation sans qu'elle ait forcément été facturée au préalable.

AjouterComposant

Motivation : Comme le montre le méta-modèle générique, un *Elément* peut être *Composé*. Cet opérateur peut être utilisé pour diverses raisons : après l'ajout d'un élément composé, après la fusion de deux éléments composés, pour prendre en compte de nouvelles exigences…

Définition :

AjouterComposant : Elément → Elément²
AjouterComposant (E) = E.est-composé-de (E_1) | E, E_1 ∈ Elément

Description : L'opérateur *AjouterComposant* permet de préciser que l'élément ajouté fait partie d'un autre élément autre que le modèle lui-même. La signature de l'opérateur précise que, dans la situation To-Be, on a un élément de plus que dans la situation As-Is initiale. Le prédicat permet de spécifier que l'élément ajouté E_1 fait partie de l'élément *Composé* E.

Exemple : Il est, par exemple, possible d'ajouter des attributs à l'entité type Chambre que l'on vient de créer. L'ajout d'attribut se fait en utilisant l'opérateur *AjouterComposant* dans la mesure où *l'Elément* entité type est un *Elément Composé*. L'opérateur *AjouterComposant* permet d'ajouter l'attribut 'nombre de lits' dans l'entité type Chambre.

SupprimerComposant

Motivation : Suite à une création, une fusion, une division ou, tout simplement, pour satisfaire de nouvelles exigences, il peut s'avérer qu'un élément composant un autre élément soit devenu inutile. L'opérateur *SupprimerComposant* permet de le supprimer.

Définition :

SupprimerComposant : Elément² → Elément
SupprimerComposant (E, E_1) = ¬ E.est-composé-de (E_1) ∧ E_1 ∉ M | E, E_1 ∈ Elément

Description : L'opérateur *SupprimerComposant* permet de supprimer un élément intervenant dans la composition d'un élément Composé. Comme le montre la définition formelle de l'opérateur, après application de cet opérateur, le modèle possède un élément de moins et l'élément composé ne contient plus l'élément supprimé.

Exemple : En ajoutant l'entité type Chambre, l'attribut nb-chambre de l'entité type Hôtel est devenu inutile. L'opérateur *SupprimerComposant* permet de supprimer cet attribut.

Les huit opérateurs qui suivent ne font pas partie de l'ensemble minimal des opérateurs génériques mais permettent de définir une typologie générique sémantiquement riche.

Renommer

Motivation : L'opérateur *Renommer* permet de modifier le nom d'un *Elément* sans changer la sémantique de cet *Elément*.

Définition :

Renommer : Elément, String → Elément, String
Renommer (E, N) : E.nom(N_1) | N, N_1 ∈ String, E ∈ Elément

Description : L'opérateur *Renommer* affecte seulement la façon dont l'organisation se réfère à un élément. Cet élément existe et a un nom avant et après l'application de l'opérateur. Le prédicat de l'opérateur permet de préciser que dans le modèle To-Be, l'élément E a pour nom N_1.

Exemple : Dans le cas de la gestion de réservations de chambres d'hôtel, l'opérateur *Renommer* permet de changer le nom de l'entité type Station en Destination afin que cette entité reflète mieux la réalité. En effet, dans la situation To-Be, les dirigeants souhaitent que le système ne gère plus seulement des hôtels dans des stations de skis ou des stations balnéaires, mais aussi dans des grandes villes ou à la campagne.

Modifier

Motivation : Cet opérateur a pour but de changer une propriété d'un *Elément* du modèle As-Is par exemple après une fusion, un remplacement ou pour répondre à de nouvelles exigences.

Définition :

Modifier : Elément, Propriété → Elément, Propriété
Modifier (E, P) = E.a(P) | P ∈ Propriété, E ∈ Elément

Description : L'opérateur *Modifier* ne change pas la structure de *l'Elément* qui dans les modèles As-Is et To-Be a le même nombre de propriétés. Seule la propriété P varie.

Exemple : Il est, par exemple, possible de changer le domaine de l'attribut 'Prix-du-petit-déjeuner' d'entier à réel. En effet, si le prix en franc était toujours rond, il est indispensable de pouvoir jouer avec les fractions d'euros afin de ne pas augmenter les prix trop rapidement.

Retyper

Motivation : Afin de donner plus d'importance à un concept ou de simplifier le modèle, il est possible de retyper un *Elément*. Il s'agit, par exemple, de réifier une Relation Type en Entité Type ou de considérer qu'une stratégie doit devenir une intention dans le modèle To-Be.

Définition :

Retyper : Elément, Type → Elément, Type
Retyper (E, T) = E.type(T_1) | E ∈ Elément, T, T_1 ∈ Type

Description : L'opérateur *Retyper* permet de changer le type d'un *Elément* présent dans les situations As-Is et To-Be. La fonction type() permet de spécifier que le nouveau type de l'élément E est T_1.

Exemple : Il est, par exemple, possible de retyper l'intention 'paiement de réservation' en une stratégie 'par paiement' permettant d'atteindre l'intention *Arrêter* depuis l'intention *Gérer les réservations*. Cela signifie que le paiement est considéré comme un moyen de terminer le processus. Ce n'est plus un but à part entière.

Remplacer

Motivation : L'opérateur *Remplacer* a pour but de substituer un *Elément* par un autre. Ces deux *Eléments* n'ont pas la même sémantique.

Définition : L'opérateur *Remplacer* se définit de deux façons différentes suivant qu'il s'agisse de remplacer un élément *Lien* ou un élément *Non Lien*.

$$\text{RemplacerLien} : \text{Lien}, \text{NonLien}^2 \rightarrow \text{Lien}, \text{NonLien}^2$$
$$\text{RemplacerLien}(L, NL_1, NL_2) = L \notin M \wedge L_1 \in M \wedge L_1.\text{a-pour-source}(NL_1) \wedge L_1.\text{a-pour-cible}(NL_2) \mid NL_1, NL_2 \in \text{NonLien}, L, L_1 \in \text{Lien}, M \in \text{Modèle}$$

$$\text{RemplacerNonLien} : \text{NonLien} \rightarrow \text{NonLien}$$
$$\text{RemplacerNonLien}(NL) = NL \notin M \wedge NL_1 \in M \mid NL, NL_1 \in \text{Non Lien}, M \in \text{Modèle}$$

Description : L'opérateur *RemplacerLien* permet de substituer au *Lien* L qui a pour source NL_1 et pour cible NL_2, le lien L_1 qui a la même source et la même cible. Le *Lien* L n'existe plus dans le modèle M après application de l'opérateur.

L'opérateur *RemplacerNonLien* permet de préciser que l'élément *Non Lien* NL qu'on remplace, n'existe plus dans le modèle M après application de l'opérateur. Il est substitué par l'élément *Non Lien* NL_1.

Exemple : L'opérateur *Remplacer* permet, par exemple, de substituer l'intention 'Gérer la relation Client', à l'intention du modèle As-Is 'Gérer les réservations'. Cette dernière disparaît donc du modèle. Ce changement permet de traduire la volonté de l'organisation de passer d'une politique orientée produit à une politique orientée client.

Fusionner

Motivation : L'opérateur *Fusionner* permet de rassembler au sein d'un même *Elément* deux *Eléments* distincts dans le modèle As-Is.

Définition : L'opérateur *Fusionner* se définit de deux façons différentes suivant qu'il s'agisse de fusionner deux éléments *Lien* ou deux éléments *Non Lien*.

$$\text{FusionnerLien} : \text{Lien}^2 \rightarrow \text{Lien}, \text{NonLien}^2$$
$$\text{FusionnerLien}(L_1, L_2) = L \in M \wedge L.\text{a-pour-source}(NL_1) \wedge L.\text{a-pour-cible}(NL_2) \wedge L_1 \notin M \wedge L_2 \notin M \mid NL_1, NL_2 \in \text{NonLien}, L, L_1, L_2 \in \text{Lien}, M \in \text{Modèle}$$

$$\text{FusionnerNonLien} : \text{NonLien}^2 \rightarrow \text{NonLien}$$
$$\text{FusionnerNonLien}(NL_1, NL_2) = NL \in M \wedge NL_1 \notin M \wedge NL_2 \notin M \mid NL, NL_1, NL_2 \in \text{NonLien}, M \in \text{Modèle}$$

Description : L'opérateur *FusionnerLien* permet de réunir en un seul et même *Lien* L deux *Liens* L_1 et L_2. Ces deux liens n'existent donc plus dans le modèle M après application de l'opérateur. En revanche, il existe dans le modèle To-Be, deux éléments *Non Lien* NL_1 et NL_2 tels que L a pour source NL_1 et pour cible NL_2.

L'opérateur *FusionnerNonLien* permet de rassembler au sein d'un unique élément *Non Lien* NL deux éléments *Non Lien* du As-Is, NL_1 et NL_2. Ces derniers n'existent plus dans le modèle M après application de l'opérateur.

Exemple : Il est, par exemple, possible de fusionner les entités Restaurant, Service de chambre et Activité en une unique entité type Service correspondant aux différents types de services que proposent les hôtels. Les trois entités types Restaurant, Service de Chambre et Activité n'apparaissent pas dans le modèle To-Be.

Diviser

Motivation : L'opérateur *Diviser* permet de scinder un unique *Elément* en deux *Eléments* distincts.

Définition : L'opérateur *Diviser* se définit de façon différente suivant que l'élément à scinder est un élément *Lien* ou un élément *Non Lien*.

$$\text{DiviserLien} : \text{Lien} \rightarrow \text{Lien}^2,\ \text{NonLien}^4$$
$$\text{DiviserLien}(L) = L \notin M \wedge L_1 \in M \wedge L_2 \in M \wedge L_1.\text{a-pour-source}(NL_1) \wedge L_1.\text{a-pour-cible}(NL_2) \wedge (L_2.\text{a-pour-source}(NL_3) \wedge L_2.\text{a-pour-cible}(NL_4)) \mid NL_1, NL_2, NL_3, NL_4 \in \text{NonLien}, L_1, L_2 \in \text{Lien}, M \in \text{Modèle}$$

$$\text{DiviserNonLien} : \text{NonLien} \rightarrow \text{NonLien}^2$$
$$\text{DiviserNonLien}(NL) = NL \notin M \wedge NL_1 \in M \wedge NL_2 \in M \mid NL, NL_1, NL_2 \in \text{NonLien}, M \in \text{Modèle}$$

Description : L'opérateur *DiviserLien* permet de préciser que l'élément *Lien* L est scindé en deux éléments L_1 et L_2. L n'appartient plus au modèle M après l'application de l'opérateur. En revanche, il existe des éléments *Non Lien* NL_1, NL_2, NL_3, NL_4, (pas obligatoirement deux à deux distincts), tels que L_1 et L_2 ont respectivement pour source NL_1 et NL_3 et pour cible NL_2 et NL_4.

L'opérateur *DiviserNonLien* spécifie que l'élément NL du modèle As-Is est scindé en deux éléments *Non Lien* NL_1 et NL_2 dans le modèle To-Be. NL n'appartient plus au modèle M après application de l'opérateur.

Exemple : L'opérateur *Diviser* permet par exemple de scinder l'entité type Réservation en deux entités types Demande et Réservation, la première permet de gérer tout ce qui se passe avant que le client n'accepte l'offre qui lui est faîte, la seconde correspond à tout ce qui survient après.

ChangerOrigine

Motivation : Suite à un ajout, un remplacement, une fusion ou une division d'un élément ; ou pour prendre en compte de nouvelles exigences, il est possible de modifier l'origine d'un élément lien. L'opérateur *ChangerOrigine* est alors appliqué.

Définition : L'opérateur *ChangerOrigine* se définit de façon différente suivant qu'il s'agisse de modifier la source ou la cible de l'élément Lien.

ChangerSource : Elément, Lien → Elément, Lien
ChangerSource (E, L) = ¬ L.a-pour-source (E) ∧ L.a-pour-source (E_1) | (E, E_1) ∈ NonLien, L ∈ Lien

ChangerCible : Elément, Lien → Elément, Lien
ChangerCible (E, L) = ¬ L.a-pour-cible (E) ∧ L.a-pour-cible (E_1) | (E, E_1) ∈ Elément, L ∈ Lien

Description : Les deux opérateurs *ChangerSource* et *ChangerCible* ont la même structure. Ils spécifient que l'élément *Lien* L après application de l'opérateur a pour source (respectivement pour cible) E_1 et non plus E. E et E_1 sont le plus souvent des éléments *NonLien*. Cependant, afin de pouvoir tenir compte des *Liens Composés* (par exemple les relations n-aires), un *Lien* peut avoir pour source (ou cible) un autre *Lien*.

Exemple : Dans la situation As-Is, la relation 'porte sur' a pour source l'entité type Réservation et pour cible l'entité type Hôtel. L'opérateur *ChangerOrigine* permet de modifier l'entité type cible de cette relation pour l'entité type Chambre.

DéplacerComposant

Motivation : L'opérateur *DéplacerComposant* permet de repositionner un élément dans la structure de l'élément Composé du modèle To-Be.

Définition : L'opérateur *DéplacerComposant* se définit de façon différente suivant que l'élément à repositionner est un élément *Lien* ou un élément *Non Lien*.

DéplacerComposantLien : Lien, NonLien², Elément → Lien, NonLien², Elément
DéplacerComposantLien (L, NL_1, NL_2, E) = L.a-pour-source(NL_3) ∧ L.a-pour-cible(NL_4) ∧ E.est-composé-de (L) ∧ E.est-composé-de (NL_3) ∧ E.est-composé-de (NL_4) ∧ E.est-composé-de (NL_1) ∧ E.est-composé-de (NL_2) | NL_1, NL_2, NL_3, NL_4 ∈ NonLien, L ∈ Lien, E ∈ Elément

DéplacerComposantNonLien : NonLien, Elément → NonLien, Elément
DéplacerComposantNonLien (NL, E) = E.est-composé-de (NL) | E ∈ Elément, NL ∈ NonLien

Description : L'opérateur *DéplacerComposantLien* permet de préciser que l'élément Lien L qui est repositionné dans l'élément composé E reste un élément de E, mais il a désormais pour source NL_3 et pour cible NL_4. Les éléments NL_1, NL_2, NL_3, NL_4 ne sont pas forcément deux à deux distincts.

L'opérateur *DéplacerComposantNonLien* spécifie que l'élément *Non Lien* NL est toujours un composant de l'élément composé E après application de l'opérateur.

4.3. Propriétés de la typologie générique d'écarts

Nous avons mentionné précédemment que la typologie générique d'opérateurs d'écarts devait être *complète*, *correcte*, *consistante*, *minimale* et *sémantiquement riche*. A partir de la

définition des opérateurs, nous montrons dans cette section que la typologie satisfait ces qualités. Rappelons qu'un même ensemble ne pouvant à la fois être sémantiquement riche et minimal, nous avons précisé parmi les quatorze opérateurs génériques ceux qui forment l'ensemble minimal. Les autres permettent à la typologie d'être sémantiquement riche.

4.3.1 Complétude

Rappelons qu'un ensemble d'opérateurs de changement est considéré comme *complet* s'il subsume tout type de changement possible.

La typologie générique d'écarts semble intuitivement capturer tous les types 'intéressants' de changements ; démontrons qu'elle est complète. Pour cela, nous nous inspirons de la démonstration faite dans [Banerjee87] pour les opérateurs d'évolution de schémas ORION.

A partir des opérateurs définis précédemment, considérons les opérateurs suivants :

- *Joindre* : cet opérateur permet d'ajouter une propriété à un élément ;
- *Enlever* : cet opérateur permet de supprimer une propriété existante d'un élément ;
- *Ajouter* : cet opérateur permet d'ajouter à un modèle existant tout type d'élément et donc aussi bien un élément lien qu'un élément non lien ;
- *Supprimer* : cet opérateur permet de supprimer tout type d'élément d'un modèle existant ;
- *AjouterComposant* : cet opérateur permet d'ajouter un élément à un élément composé existant ;
- *SupprimerComposant* : cet opérateur permet de supprimer un élément d'un élément composé existant.

Lemme 1 : Pour n'importe quel modèle M, il existe une séquence finie de {*Enlever*, *Supprimer*, *SupprimerComposant*} qui permet de transformer ce modèle en un modèle réduit à sa racine.

Preuve Il apparaît clairement que *Enlever*, *Supprimer* et *SupprimerComposant* peuvent être appliqués de façon répétitive pour supprimer tous les éléments jusqu'à la racine du modèle M. La présence, ici, de l'opérateur *Enlever* permet de remédier au cas où, pour certains méta-modèles, la suppression des différentes propriétés d'un élément constitue une étape préalable à la suppression de l'élément lui-même.

Lemme 2 : Il existe une séquence finie de {*Joindre*, *Ajouter*, *AjouterComposant*} qui génère n'importe quel modèle.

Preuve Considérons deux modèles M et M'. M est un modèle arbitraire avec un nombre fini d'éléments. M' est un modèle réduit à sa plus simple expression c'est-à-dire sa racine. Le processus de construction qui suit permet de générer un modèle équivalent à M, à partir de M'.

Parcourir M et :

a. pour chaque élément composé *x* de M : en utilisant les opérateurs *Joindre*, *Ajouter* et *AjouterComposant*, ajouter un élément composé *x*' à M' tel que *x*'

ait les mêmes éléments qui le composent que *x* et chaque élément les mêmes propriétés que ceux composant *x*.

b. pour chaque élément simple Non Lien *y* de M n'appartenant pas à un autre élément composé autre que M lui-même : ajouter à M' un élément simple *y'* ayant les mêmes propriétés que *y* en utilisant les opérateurs *Joindre* et *Ajouter*.

c. pour chaque élément lien ayant pour cible (respectivement pour source) *x* ou *y*, ajouter un lien correspondant, ayant pour cible (respectivement pour source) *x*' ou *y*' et ayant les mêmes propriétés que le lien entrant dans (respectivement sortant de) *x* ou *y* en utilisant les opérateurs *Joindre* et *Ajouter*.

Le modèle M' résultant de ce processus est équivalent à M. En effet, M' a les mêmes éléments non liens et les mêmes éléments liens que M (aucun élément lien ou non lien additionnel) et les mêmes composants et propriétés pour chacun des éléments.

Théorème 1 : (Complétude) Etant donnés deux modèles arbitraires M1 et M2, il existe toujours une séquence finie F d'opérateurs {*Joindre*, *Enlever*, *Ajouter*, *Supprimer, AjouterComposant*, *SupprimerComposant*} telle que F(M1) = M2.

Preuve On peut obtenir M2 en réduisant d'abord M1 à sa racine en appliquant le lemme 1 et ensuite en construisant M2 à partir de la racine selon le lemme 2.

La complétude de l'ensemble des opérateurs génériques étant démontrée, il ne sera pas nécessaire de démontrer la complétude de l'ensemble des opérateurs spécifiques. En effet, si chacun des opérateurs {*Joindre*, *Enlever*, *Ajouter*, *Supprimer*, *AjouterComposant*, *SupprimerComposant*} est instancié pour tous les éléments clé du méta-modèle spécifique, la complétude de la typologie spécifique est subsumée par celle de la typologie générique.

4.3.2 Minimalité

Rappelons qu'un ensemble d'opérateurs est minimal s'il est complet et s'il ne contient aucun opérateur qui ne puisse être obtenu par composition d'autres opérateurs.

Nous avons montré dans la démonstration de la complétude (i) que chacun des opérateurs de l'ensemble {*Joindre*, *Enlever*, *Ajouter*, *Supprimer*, *AjouterComposant*, *SupprimerComposant*} est nécessaire pour satisfaire la propriété de complétude et (ii) qu'aucun autre opérateur n'est nécessaire.

De plus, aucun de ces six opérateurs ne peut être obtenu par composition d'autres opérateurs de l'ensemble ou du reste de la typologie générique.

Cet ensemble formé des opérateurs *Joindre*, *Enlever*, *Ajouter*, *Supprimer*, *AjouterComposant* et *SupprimerComposant* correspond donc à l'ensemble minimal. La typologie générique n'est pas minimale dans la mesure où elle contient d'autres opérateurs. Cependant, nous avons réussi à mettre en évidence au sein de cette typologie l'ensemble minimal.

La typologie spécifique sera minimale si (i) chacun de ces six opérateurs est instancié pour les éléments clé du méta-modèle spécifique et (ii) aucun autre opérateur générique n'est instancié.

4.3.3 Correction

On dit qu'un opérateur est correct si l'état dans lequel il amène le modèle vérifie tous les invariants définis pour le méta-modèle en question. Les invariants sont des propriétés du méta-modèle spécifique qui doivent être satisfaites avant et après toute modification du modèle.

La correction est une propriété qui doit être vérifiée au niveau spécifique en tenant compte des invariants définis sur le méta-modèle spécifique. Il est néanmoins possible d'affirmer que les opérateurs génériques tels qu'ils ont été définis précédemment satisfont tous les contraintes imposées par le méta-modèle générique à savoir :

- un élément *Lien* a une source et une cible ;
- une *Propriété* est attachée à un *Elément* ;
- un élément *Composé* est composé de plusieurs *Eléments*.

4.3.4 Consistance

Comme cela a été précisé auparavant, un ensemble d'opérateurs est consistant si les définitions des différents opérateurs n'entrent pas en conflit les unes par rapport aux autres. Des définitions sont en conflit si un même écart peut être interprété comme se référant à plusieurs opérateurs différents.

Au niveau générique les opérateurs sont consistants et non ambigus. En effet, d'une part chaque opérateur s'applique sur un type d'objet particulier (*Elément Lien*, *Elément*, *Propriété*…), d'autre part, pour un même élément, il n'existe pas deux opérateurs avec des définitions identiques ou similaires.

4.3.5 Richesse Sémantique

Une typologie d'opérateurs est considérée comme sémantiquement riche si chaque type de changement peut être exprimé en n'utilisant qu'un seul opérateur.

La typologie générique est sémantiquement riche dans la mesure où chaque type de changement correspondant à l'ajout, la suppression, la division, la fusion d'éléments… peut être exprimé en n'utilisant qu'un seul opérateur.

La typologie spécifique sera sémantiquement riche si chacun des quatorze opérateurs de la typologie générique est instancié pour chacun des éléments du méta-modèle spécifique mis en évidence en instanciant le méta-modèle générique.

Dans cette section, nous avons montré que l'ensemble générique des opérateurs d'écarts défini précédemment satisfait un certain nombre de propriétés. La section suivante propose un processus pour générer une typologie d'opérateurs d'écarts associée à un méta-modèle spécifique à partir de la typologie générique. Les propriétés de l'ensemble des opérateurs sont subsumées par celle de la typologie générique.

5. Génération d'une typologie spécifique d'écarts

Le processus de génération d'une typologie spécifique peut avoir un but différent suivant qu'il existe ou non au préalable une typologie d'opérateurs d'écarts associée au méta-modèle spécifique [Etien03b]. Dans le premier cas, le processus permet de modifier la typologie existante afin par exemple, qu'elle satisfasse certaines des propriétés énoncées plus haut. Dans le second cas, il s'agit de construire une typologie d'opérateurs d'écarts associée au méta-modèle spécifique. Le processus est différent dans chacun de ces deux cas.

5.1. Construire une typologie d'écarts associée à un méta-modèle spécifique

La Figure 66 schématise le processus qui permet de générer une typologie d'écarts associée à un méta-modèle spécifique alors qu'il n'en existe pas d'autre.

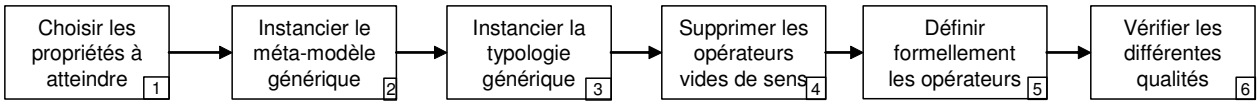


Figure 66 : Processus de génération d'une typologie spécifique d'écarts

Ce processus est constitué de six étapes :

1. *le choix des propriétés à atteindre*, en particulier la minimalité ou la richesse sémantique de la typologie spécifique. En effet, comme nous l'avons vu précédemment, (i) ces deux propriétés ne peuvent être satisfaites en même temps, (ii) l'ensemble des opérateurs à instancier dans l'un ou l'autre des deux cas n'est pas le même. Pour atteindre la minimalité, seuls les opérateurs génériques *Joindre*, *Enlever*, *Ajouter*, *Supprimer*, *AjouterComposant* et *SupprimerComposant* sont instanciés. Pour satisfaire le critère richesse sémantique, tous les opérateurs de la typologie générique sont instanciés lors la deuxième étape.

2. *l'instanciation du méta-modèle générique*. Cette étape a pour but de construire le méta-modèle spécifique par instanciation du méta-modèle générique. Il peut s'avérer nécessaire d'utiliser la connaissance conceptuelle du méta-modèle spécifique pour préciser, ou modifier, le cas échéant, sa formalisation. La formalisation des concepts du méta-modèle spécifique est un pré-requis pour définir les opérateurs d'écarts, ce qui explique le flux de l'étape 1 '*Instancier le méta-modèle générique*' vers l'étape 2 '*Instancier la typologie générique*'.

3. *l'instanciation de la typologie générique*. Cette étape utilise la typologie générique d'écarts pour générer, par instanciation, une typologie spécifique. Suivant le choix effectué à l'étape 1, tous les opérateurs, ou seuls ceux formant l'ensemble minimal, sont instanciés pour chaque concept, en fonction de son type générique *Lien*, *Non Lien*, *Composé*, *Simple* ou *Propriété*.

4. *la suppression des opérateurs vide de sens*. Une fois la collection d'opérateurs d'écarts définie, cette étape permet de supprimer des opérateurs qui n'auraient pas de sens ou ne pourraient être mis en œuvre dans le contexte du méta-modèle spécifique retenu.

5. *la définition formelle des opérateurs*. Cette étape s'appuie sur la définition formelle des opérateurs génériques et sur la connaissance du méta-modèle spécifique pour définir formellement chacun des opérateurs spécifiques.

6. *la vérification des différentes propriétés*. Cette étape correspond à l'évaluation des cinq propriétés identifiées précédemment. Durant cette étape, la typologie spécifique peut être modifiée dans le but d'atteindre les différents critères de qualité.

Il est important de noter que l'ordre dans lequel les propriétés sont vérifiées n'a pas d'importance.

- La vérification de la propriété de correction est la seule qui peut engendrer des modifications directement sur les définitions des opérateurs, sans en changer le sens global, mais afin que celles-ci satisfassent les invariants définis pour le méta-modèle spécifique. Les vérifications des autres propriétés provoquent des ajouts ou des suppressions d'opérateurs, mais pas de modification. Vérifier la correction n'a donc aucune incidence sur les autres propriétés et vice versa.

- La vérification de la minimalité peut engendrer des suppressions d'opérateurs mais cela n'a aucune incidence sur les autres propriétés car (i) la propriété de complétude est toujours satisfaite (compte tenu de la définition de ces deux propriétés), (ii) si la typologie était consistante, elle le reste après vérification de la minimalité, si elle ne l'était pas, au mieux elle le devient.

- La vérification de la complétude peut engendrer uniquement des ajouts d'opérateurs mais cela n'a aucune incidence sur (i) la minimalité car si la complétude n'était pas satisfaite, la minimalité non plus (ii) la richesse sémantique ou la consistance car si, au départ, la typologie n'était pas complète c'est qu'il manquait des opérateurs pour répondre aux besoins d'ajouter ou de supprimer certains types d'éléments ou leur propriétés, la typologie n'était donc pas sémantiquement riche et la typologie résultante ne sera pas inconsistante car aucun opérateur ne permettait de satisfaire ces besoins.

- La vérification de la richesse sémantique peut entraîner uniquement des ajouts d'opérateurs répondant à de nouvelles exigences d'évolution. Un ensemble complet d'opérateurs reste complet après l'ajout de nouveaux opérateurs. Il en est de même pour un ensemble consistant d'opérateurs auquel on ajoute des opérateurs qui répondent à de nouvelles exigences.

- La vérification de la consistance implique, comme seule modification, des suppressions d'opérateurs pour lesquels il existe un autre opérateur dans la typologie qui répond à la même exigence. Les deux typologies avant et après vérifications permettent de satisfaire les mêmes exigences d'évolution. Toutes les propriétés vérifiées par la typologie avant vérification de cette propriété le sont également après.

Le processus se termine lorsque la typologie spécifique satisfait les propriétés.

5.2. Modifier une typologie existante

Le processus de génération d'une typologie spécifique d'écarts ayant pour but de modifier une typologie existante est une variante du processus précédent. Il compte deux étapes

supplémentaires correspondant à la formalisation de la typologie existante et à la mise en correspondance des deux typologies.

L'étape de *formalisation de la typologie existante* permet de structurer la typologie existante d'opérateurs spécifiques en fonction du formalisme du méta-modèle générique et de la typologie générique d'écarts. Elle a donc lieu après la formalisation du méta-modèle spécifique par instanciation du méta-modèle générique, parallèlement à la construction d'une typologie spécifique d'écarts par instanciation de la typologie générique d'écarts.

L'étape de *mise en correspondance* permet de relier les deux typologies (celle préexistant et celle construite par instanciation de la typologie générique). En effet, il peut, par exemple, s'avérer nécessaire de fusionner un opérateur préexistant avec un opérateur obtenu par instanciation de la typologie générique d'écarts. Les différentes qualités sont vérifiées sur la typologie obtenue après la mise en correspondance des deux typologies.

La Figure 67 présente le processus permettant de modifier une typologie existante.

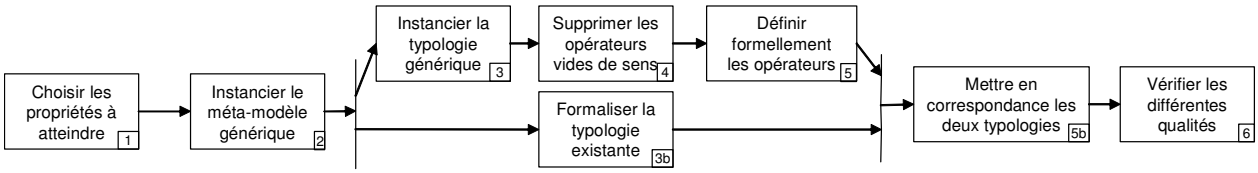


Figure 67 : Variante du processus de génération permettant de modifier une typologie existante

Dans la section suivante, nous utilisons le premier processus pour générer une typologie d'opérateurs associée au méta-modèle pivot.

6. Génération d'une typologie spécifique d'écarts pour le méta-modèle pivot

Cette section montre comment définir une typologie spécifique d'écarts. Elle applique le processus de génération (Figure 66) au méta-modèle pivot.

6.1. Etape 1 : Choix des propriétés à atteindre

Dans la mesure où il nous parait nécessaire de répondre au mieux aux besoins des utilisateurs et des décideurs, la typologie spécifique associée au méta-modèle pivot doit être sémantiquement riche en définissant des opérateurs complexes.

6.2. Etape 2 : Instanciation du méta-modèle générique

Dans cette section, nous formalisons le méta-modèle pivot en instanciant le méta-modèle générique. Ceci correspond à la réalisation de la deuxième étape du processus de génération.

La Figure 68 montre la formalisation du méta-modèle pivot obtenue après réalisation de cette étape. Elle met en évidence les concepts suivants :

Une *Carte pivot* est un élément *Composé* de *Sections*. Une carte pivot possède deux propriétés : un code et une désignation.

Une *Intention* est un élément *Simple* de type *Non lien* qui a pour propriété les *états désirables* que l'on cherche à atteindre par la réalisation de cette intention ainsi qu'une couleur permettant de préciser si l'intention correspond à un objectif du système et/ou des processus.

Une *Stratégie* est un élément *Simple* de type *Lien* qui a pour source et pour cible deux intentions, celles-ci pouvant être distinctes ou non.

Une *Section* est un élément *Composé* de type *Non Lien* dont la structure est immuable. En effet, une section est toujours composée d'une intention source, d'une intention cible et d'une stratégie. De plus, une section a huit propriétés : une règle de gestion, une pré-condition, une post-condition, des ressources, un acteur déclencheur, un code, une désignation et une couleur.

Un *Chemin*, un *Paquet* et un *Segment* sont des éléments *Lien* qui permettent de lier deux sections. Un chemin permet de préciser qu'une section en précède une autre. Par défaut, deux sections ayant les mêmes intentions sources et cibles forment un segment (c'est-à-dire sont reliées par un lien OR). Il est cependant possible qu'elles forment un paquet ce qui permet de préciser que les deux sections sont mutuellement exclusives (c'est-à-dire sont reliées par un lien XOR).

Un *Affinement* est un élément *Lien* qui a pour source une section et pour cible une carte.

Une *Classe* est un élément *Simple* ayant pour propriété des états et des attributs.

Une *Relation* est un élément *Lien* permettant de relier des classes entre elles.

Dans le contexte d'une carte pivot, l'élément *racine* correspond à une carte pivot composée des deux intentions *Démarrer* et *Arrêter* (qui existent dans toute carte) et d'une stratégie permettant d'atteindre cette dernière à partir de la première.

Mais le méta-modèle pivot, grâce au concept d'affinement permet également de construire un modèle pivot (c'est-à-dire un arbre dont les nœuds sont des *Cartes pivot* et les liens des *Affinements*) et de naviguer entre ces cartes pivot. Dans ce contexte, la *racine* est la carte pivot racine aussi appelée carte pivot de plus haut niveau.

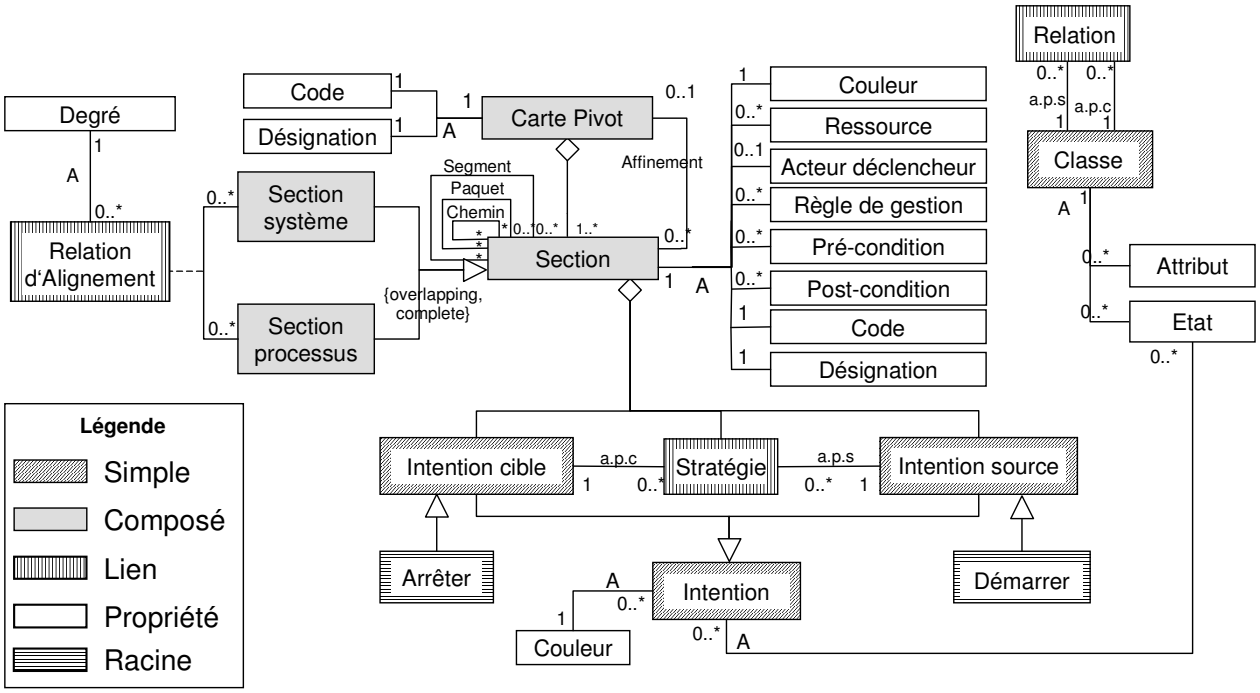


Figure 68: Instanciation du méta-modèle générique

6.3. Etape 3 : Instanciation de la typologie générique

L'instanciation des quatorze opérateurs génériques (Tableau 20) pour les éléments spécifiques des cartes permet d'aboutir au Tableau 21. Comme le méta-modèle pivot comprend onze éléments Intention, Stratégie, Section, Carte pivot, Affinement, Chemin, Paquet, Segment, Classe, Relation et Relation d'alignement, le Tableau 21 est organisé en onze colonnes.

Opérateur	Intention	Stratégie	Section	Carte pivot	Affinement	Paquet	Segment	Chemin	Classe	Relation	Relation alignement
Renommer	RenommerIntention	RenommerStratégie	RenommerSection	RenommerCarte	RenommerAffinement	RenommerPaquet	RenommerSegment	RenommerChemin	RenommerClasse	RenommerRelation	RenommerRelationA1
Ajouter	AjouterIntention	AjouterStratégie	AjouterSection	AjouterCarte	AjouterAffinement	AjouterPaquet	AjouterSegment	AjouterChemin	AjouterClasse	AjouterRelation	AjouterRelationA1
Supprimer	SupprimerIntention	SupprimerStratégie	SupprimerSection	SupprimerCarte	SupprimerAffinement	SupprimerPaquet	SupprimerSegment	SupprimerChemin	SupprimerClasse	SupprimerRelation	SupprimerRelationA1
Fusionner	FusionnerIntention	FusionnerStratégie	FusionnerSection	FusionnerCarte	FusionnerAffinement	FusionnerPaquet	FusionnerSegment	FusionnerChemin	FusionnerClasse	FusionnerRelation	FusionnerRelationA1
Diviser	DiviserIntention	DiviserStratégie	DiviserSection	DiviserCarte	DiviserAffinement	DiviserPaquet	DiviserSegment	DiviserChemin	DiviserClasse	DiviserRelation	DiviserRelationA1
Remplacer	RemplacerIntention	RemplacerStratégie	RemplacerSection	RemplacerCarte	RemplacerAffinement	RemplacerPaquet	RemplacerSegment	RemplacerChemin	RemplacerClasse	RemplacerRelation	RemplacerRelationA1
Changer Origine	N/A (Non Applicable)	ChangerIntentionSource ChangerIntentionCible	N/A.	N/A.	ChangerSectionSource ChangerCarteCible	ChangerSectionSource ChangerSectionCible	ChangerSectionSource ChangerSectionCible	ChangerSectionSource ChangerSectionCible	N/A	ChangerClasseSource ChangerClasseCible	ChangerSourceRelationA1 ChangerCibleRelationA1
Retyper	RetyperIntention	RetyperStratégie	RetyperSection	RetyperCarte	RetyperAffinement	RetyperPaquet	RetyperSegment	RetyperChemin	RetyperClasse	RetyperRelation	RetyperRelationA1
Ajouter Composant	N/A	N/A	AjouterComposant Section	AjouterSection Carte	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Supprimer Composant	N/A.	N/A.	SupprimerComposant Section	SupprimerSection Carte	N/A.	N/A.	N/A.	N/A.	N/A.	N/A.	N/A.
Déplacer Composant	N/A.	N/A.	DéplacerComposant Section	DéplacerSection Carte	N/A.	N/A.	N/A.	N/A.	N/A.	N/A.	N/A.
Joindre	JoindreEtatIntention JoindreCouleurIntention	N/A	JoindrePreCondition JoindrePostCondition JoindreActeur JoindreRessource JoindreRègleGestion JoindreCodeSection JoindreDésignationSection JoindreCouleurSection	JoindreCodeCarte JoindreDésignation	N/A	N/A	N/A	N/A	JoindreEtatClasse JoindreAttribut	N/A	JoindreDegré
Enlever	EnleverEtatIntention EnleverCouleurIntention	N/A	EnleverPreCondition EnleverPostCondition EnleverActeur EnleverRessource EnleverRègleGestion EnleverCodeSection EnleverDésignationSection EnleverCouleurSection	EnleverCodeCarte EnleverDésignation	N/A	N/A	N/A	N/A	EnleverEtatClasse EnleverAttribut	N/A	EnleverDegré
Modifier	ModifierEtatIntention ModifierCouleurIntention	N/A	ModifierPreCondition ModifierPostCondition ModifierActeur ModifierRessource ModifierRègleGestion ModifierCodeSection ModifierDésignationSection ModifierCouleurSection	ModifierCodeCarte ModifierDésignation	N/A	N/A	N/A	N/A	ModifierEtatClasse ModifierAttribut	N/A	ModifierDegré

Tableau 21 : Typologie spécifique obtenue par instanciation

La nature même des éléments (*Lien*, *Non Lien*, *Simple* ou *Composé*) permet de restreindre le nombre d'opérateurs spécifiques dans la typologie. Ceci explique les « N/A » (non applicable) pour les opérateurs *AjouterComposant*, *SupprimerComposant*, *DéplacerComposant*, *ChangerOrigine*, *Joindre*, *Enlever* et *Modifier* qui ne s'appliquent qu'à un seul type d'élément :

- Les opérateurs *AjouterComposant*, *SupprimerComposant* ou *DéplacerComposant* qui permettent de modifier la structure d'un élément Composé ne sont pas applicables à des éléments simples.
- Les opérateurs *Joindre*, *Enlever* et *Modifier* qui permettent de faire évoluer les propriétés d'un élément ne sont instanciés que pour les éléments possédant des propriétés.
- Enfin, l'opérateur *ChangerOrigine* ne s'applique qu'aux éléments *Lien*.

6.4. Etape 4 : Suppression des opérateurs vides de sens

Appliquer les opérateurs *AjouterComposant*, *SupprimerComposant* et *DéplacerComposant* à l'élément *Section* n'a pas de sens puisque la structure d'une section est immuable. Une section est toujours composée d'une intention source, une intention cible et une stratégie.

L'opérateur *Renommer* ne s'applique pas à élément *Affinement* car celui-ci n'a pas de nom.

Renommer une section ou une carte n'a pas de sens puisque le nom de la première est obtenu par concaténation des noms des intentions et de la stratégie qui la composent. Le nom d'une carte pivot, quant à lui, correspond au nom de la section qu'elle affine. Dans ces deux cas les noms de ces éléments ne peuvent être modifiés directement. Les opérateurs *RenommerCarte* et *RenommerSection* sont supprimés de la typologie spécifique.

L'opérateur *Retyper* n'a de sens que s'il est appliqué sur (i) les éléments Intention et Stratégie ; une Intention pouvant être retypée en Stratégie et vice-versa (ii) les éléments Classe et Relation ; une Classe pouvant être retypée en Relation et vice-versa et (iii) sur les liens Paquet et Segment pour transformer un Paquet en Segment et vice-versa. Nous avons donc supprimé les opérateurs *RetyperSection*, *RetyperCarte*, *RetyperAffinement*, *RetyperChemin* et *RetyperRelationAl*.

Les opérateurs *FusionnerAffinement* et *DiviserAffinement* n'ont pas de sens. En effet, une section ne peut pas être affinée par deux cartes différentes et une carte ne peut affiner qu'une seule section. La troisième étape du processus de génération d'une typologie spécifique associée au méta-modèle pivot entraîne la suppression de ces deux opérateurs.

L'opérateur *RemplacerAffinement* ne peut exister. Un tel opérateur aurait pour but de changer la façon dont la section Se de la carte pivot C_1 est affinée par la carte pivot C_2. Or il n'existe qu'une et une seule façon de relier par un lien d'affinement une section donnée à une carte pivot particulière. L'affinement n'exprime rien d'autre que le fait qu'une section peut être considérée comme une carte pivot entière, contrairement, par exemple, à l'élément *Lien* Stratégie qui, lui, permet d'exprimer non seulement que deux intentions sont reliées, mais précise aussi la façon dont cela est fait. Un raisonnement analogue permet de supprimer les opérateurs *FussionnerAffinement* et *DiviserAffinement* de la typologie.

L'opérateur *ChangerSectionSource* sur l'élément *Affinement* ne peut être conservé dans la typologie spécifique d'opérateurs associée au méta-modèle pivot. En effet, un lien d'affinement permet avant tout de préciser la section de carte qui est affinée. Un lien d'affinement puise sa sémantique dans la section qu'il affine. Il ne peut donc pas garder la même sémantique si ce n'est plus la même section qui est affinée. De façon similaire, l'opérateur *ChangerCarteCible* est retiré de la typologie car il n'a pas de sens.

Concernant les éléments Paquet et Segment, seul l'opérateur *Retyper* a du sens :

- L'opérateur *Renommer* n'a pas de sens car les éléments Paquet et Segment n'ont pas de nom.
- Les opérateurs *Ajouter* et *Supprimer* n'ont pas de sens car par défaut deux sections ayant les mêmes intentions source et cible sont reliées par un Segment. Il n'est donc pas possible d'ajouter un lien ni de le supprimer.
- Les opérateurs *Fusionner*, *Diviser* et *Remplacer* n'ont aucun sens. En effet, tous les Paquets entre deux intentions ont la même signification. Il n'est donc pas possible de les *Fusionner* entre eux. Il en est de même pour les Segments. De la même façon, il n'est pas possible de diviser (ou de remplacer) un Paquet ou un Segment.
- Les opérateurs *ChangerSectionSource* et *ChangerSectionCible* sont sémantiquement pauvres. En effet, le Paquet et le Segment sont des liens non orientés qui ne peuvent exister qu'entre deux sections ayant les mêmes intentions source et cible.

Les opérateurs *RenommerPaquet*, *RenommerSegment AjouterPaquet*, *AjouterSegment*, *SupprimerPaquet*, *SupprimerSegment*, *FusionnerPaquet*, *FusionnerSegment*, *DiviserPaquet*, *DiviserSegment*, *RemplacerPaquet*, *RemplacerSegment*, *ChangerSectionSource* et *ChangerSectionCible* ont donc été supprimés de la typologie. Pour des raisons de place, les opérateurs *RetyperPaquet* et *RetyperSegment* ont été mis dans la même colonne dans le Tableau 22.

Aucun opérateur ne peut s'appliquer sur l'élément Chemin. En effet, il n'est pas possible de renommer un tel élément car celui-ci n'a pas de nom. Il n'est pas possible d'ajouter, de supprimer, de remplacer, de fusionner ou de diviser des chemins, ce sont les sections qui sont ajoutées, supprimées, remplacées, fusionnées ou divisées. Changer la source ou la cible d'un chemin n'a pas de sens car deux sections dont l'intention source de l'une est la cible de l'autre sont forcément liées par un chemin. Retyper un chemin n'a pas non plus de sens. Nous avons donc supprimé tous les opérateurs relatifs à l'élément Chemin et ainsi que la colonne Chemin.

Nous avons également supprimé les opérateurs *AjouterSection* et *SupprimerSection* car ils feraient double usage avec les opérateurs *AjouterSectionCarte* et *SupprimerSectionCarte* qui permettent d'ajouter et de supprimer une section d'une carte. L'opérateur *DéplacerSectionCarte* a également été retiré de la typologie car il n'a pas de sens.

Nous avons supprimé les opérateurs *JoindreCodeSection*, *JoindreCodeCarte*, *EnleverCodeSection*, *EnleverCodeCarte*, *ModifierCodeSection* et *ModifierCodeCarte*. En effet, le méta-modèle présenté Figure 68 précise qu'une carte pivot (ou une section) a un et un seul code. Il n'est donc pas possible d'ajouter ni de supprimer un code de carte (ou de section). De plus, dans la mesure où ces codes sont générés automatiquement, il n'est pas possible de les modifier. Un raisonnement analogue peut être fait concernant les propriétés *DésignationCarte* et *DésignationSection*. En revanche, l'opérateur *ModifierDésignation* qui

s'applique sur une carte pivot est maintenu dans la typologie, car si les désignations des sections (il s'agit du triplet intention source, intention cible et stratégie) et des cartes affinées sont générées automatiquement, il n'en est rien pour la désignation de la carte pivot de plus haut niveau, qui peut donc être modifiée.

Aucun opérateur ne peut s'appliquer sur l'élément Relation d'alignement. En effet, un tel élément n'a pas de nom, il n'est donc pas possible de le renommer. Il est également impossible d'ajouter, de supprimer, de remplacer, de fusionner, de diviser, de retyper ou de changer l'origine d'une relation d'alignement. Ce sont les sections système et processus qui sont ajoutées, supprimées, remplacées, fusionnées, divisées ou modifiées (dans ce cas l'un des composants ou l'une des propriétés sont changées). La relation d'alignement est définie en fonction des sections de la carte pivot. Son degré est généré de façon systématique en fonction de la sémantique et de la couleur des sections.

Enfin, nous avons supprimé les opérateurs *JoindreCouleurIntention*, *JoindreCouleurSection*, *ModifierCouleurIntention*, *ModifierCouleurSection*, *EnleverCouleurIntention* et *EnleverCouleurSection* qui s'applique sur les éléments Intention et Section car une section (ou une intention) ne peuvent avoir qu'une et une seule couleur et que celle-ci ne peut être modifiée. C'est le sens et la provenance de la section ou de l'intention (carte système ou carte processus) qui en détermine la couleur.

Le Tableau 22 présente, après suppression, les opérateurs spécifiques associés au méta-modèle pivot.

Opérateur	Intention	Stratégie	Section	Carte	Affinement	Paquet/Segment	Classe	Relation
Renommer	RenommerIntention	RenommerStratégie	N/A	RenommerCarte	N/A	N/A	RenommerClasse	RenommerRelation
Ajouter	AjouterIntention	AjouterStratégie	N/A	AjouterCarte	AjouterAffinement	N/A	AjouterClasse	AjouterRelation
Supprimer	SupprimerIntention	SupprimerStratégie	N/A	SupprimerCarte	SupprimerAffinement	N/A	SupprimerClasse	SupprimerRelation
Fusionner	FusionnerIntention	FusionnerStratégie	FusionnerSection	FusionnerCarte	N/A	N/A	FusionnerClasse	FusionnerRelation
Diviser	DiviserIntention	DiviserStratégie	DiviserSection	DiviserCarte	N/A	N/A	DiviserClasse	DiviserRelation
Remplacer	RemplacerIntention	RemplacerStratégie	RemplacerSection	RemplacerCarte	N/A	N/A	RemplacerClasse	RemplacerRelation
Changer Origine	N/A (Non Applicable)	ChangerIntentionSource ChangerIntentionCible	N/A.	N/A.	N/A.	N/A.	N/A	ChangerSourceRelation ChangerCibleRelation
Retyper	RetyperIntention	RetyperStratégie			N/A	RetyperPaquet RetyperSegment	RetyperClasse	RetyperRelation
Ajouter Composant	N/A	N/A	N/A	AjouterSection Carte	N/A	N/A	N/A	N/A
Supprimer Composant	N/A.	N/A.	N/A.	SupprimerSection Carte	N/A.	N/A.	N/A.	N/A.
Déplacer Composant	N/A.	N/A.	N/A.	DéplacerSection Carte	N/A.	N/A.	N/A.	N/A.
Joindre	JoindreEtatIntention	N/A	JoindrePreCondition JoindrePostCondition JoindreActeur JoindreRessource JoindreRègleGestion		N/A	N/A	JoindreEtatClasse JoindreAttribut	N/A
Enlever	EnleverEtatIntention	N/A	EnleverPreCondition EnleverPostCondition EnleverActeur EnleverRessource EnleverRègleGestion		N/A	N/A	EnleverEtatClasse EnleverAttribut	N/A
Modifier	ModifierEtatIntention	N/A	ModifierPreCondition ModifierPostCondition ModifierActeur ModifierRessource ModifierRègleGestion	Modifier Désignation	N/A	N/A	ModifierAttribut ModifierEtatClasse	N/A

Tableau 22 : Typologie spécifique d'écarts associée au méta-modèle pivot

La connaissance du méta-modèle ne permet pas de modifier d'avantage la typologie spécifique d'écarts associée au méta-modèle pivot. La section suivante correspond à la définition précise de chacun de ces opérateurs.

6.5. Etape 5 : Définition formelle des opérateurs associés au méta-modèle pivot

La définition formelle des opérateurs de la typologie spécifique associée au méta-modèle pivot s'appuie sur celle des opérateurs génériques et sur la définition des invariants et corollaires associés à ce méta-modèle. Les invariants et les corollaires, respectivement rassemblés dans le Tableau 23 et le Tableau 24, correspondent aux propriétés que doit satisfaire le modèle pivot pour être correcte. Ces invariants et corollaires ont été définis au chapitre 6 présentant le méta-modèle pivot.

Invariants
I1. Toute carte pivot a une et une seule intention qui n'est la cible d'aucune stratégie ; l'intention *Démarrer*.
I2. Toute carte pivot a une et une seule intention qui n'est la source d'aucune stratégie ; l'intention *Arrêter*.
I3. Toute intention dans une carte pivot doit être quasi-vivace. Une intention est quasi-vivace si elle peut être atteinte au moins une fois, c'est-à-dire si il existe un chemin de *Démarrer* à cette intention.
I4. Au sein d'une carte pivot, toute intention est unique.
I5. Toute carte pivot a une et une seule intention qui peut être vue comme un ensemble vide d'états, l'intention *Démarrer*.
I6. Dans un modèle pivot, une section est la source d'au plus un lien d'affinement.
I7. L'ensemble des classes est un graphe connecté.
I8. Toute classe est unique.

Tableau 23 : Invariants associés au méta-modèle pivot

A partir de ces cinq invariants dix corollaires ont été définis.

Corollaires
C1. Une carte pivot est un graphe connecté ; il n'y a pas d'intention ni de stratégie isolée.
C2. Toute intention dans une carte pivot est la source d'au moins une stratégie sauf l'intention *Arrêter*.
C3. Toute intention dans une carte est la cible d'au moins une stratégie sauf l'intention *Démarrer*.
C4. Il existe toujours un chemin de *Démarrer* à *Arrêter*.
C5. Toute section appartient à au moins un chemin entre *Démarrer* et *Arrêter*.
C6. Toute intention correspond à un but différent et se voit attribuer un nom différent de celui des autres intentions de la carte.
C7. Toute intention est constituée d'au moins un état sauf l'intention Démarrer
C8. Toute section a sa pré-condition qui diffère de l'ensemble vide sauf les sections ayant pour source l'intention Démarrer
C9. Toute intention correspond à un ensemble différent d'états.
C10. Toute carte pivot de niveau $i+1$ ($i \neq 0$) est la cible d'un lien d'affinement ayant pour source une section du niveau i.
C11. Il n'y a pas de classe ni de relation isolée.

Tableau 24 : Corollaires associés au méta-modèle pivot

Les opérateurs spécifiques associés au méta-modèle pivot adoptent la même structure que les opérateurs génériques. Ils possèdent une signature et un prédicat. La signature précise les éléments du méta-modèle qui sont présents dans les situations As-Is et To-Be c'est-à-dire avant et après application de l'opérateur. Le prédicat décrit la situation To-Be et non la démarche pour l'atteindre.

Un certain nombre de conditions ont été ajoutées dans les prédicats afin (i) de tenir compte des spécificités du méta-modèle pivot et (ii) de définir des opérateurs corrects c'est-à-dire tels que le modèle pivot vérifie les invariants et les corollaires après leur exécution.

Les sections 6.5.1 à 6.5.6 correspondent aux opérateurs spécifiques formant l'ensemble minimal. Les sections suivantes définissent formellement les autres opérateurs.

6.5.1 Ajouter

AjouterIntention

Les invariants définis pour le modèle pivot (chapitre 6) précisent qu'aucune intention n'est isolée (**C1**). Ainsi, quand on ajoute une intention I, celle-ci doit être reliée au reste de la carte pivot par, au moins, une stratégie.

Afin de préserver le corollaire **C2**, une stratégie St_1 doit avoir l'intention ajoutée I comme intention cible et une intention I_1 comme source.

Pour préserver **C3**, une autre stratégie St_2 doit avoir l'intention ajoutée I comme intention source et une intention I_2 comme cible. Il est possible de préciser quelles sont ces intentions I_1 et I_2. Par défaut I_1 et I_2 sont respectivement les intentions *Démarrer* et *Arrêter*. I_1 et I_2 apparaissent comme paramètres optionnels dans le prédicat de l'opérateur (c'est-à-dire entre crochets). L'opérateur spécifie donc comment ajouter une intention entre deux autres intentions qui, par défaut, sont *Démarrer* et *Arrêter*.

Afin de satisfaire **C7**, l'intention I doit avoir au moins un état.

I4 est préservé s'il n'existe aucune autre intention dans la carte pivot identique à I.

> AjouterIntention : $\text{Intention}^2 \rightarrow \text{Stratégie}^2$, Intention
> AjouterIntention $([I_1], [I_2]) = I \in C \wedge St_1.\text{a-pour-source}(I_1) \wedge St_1.\text{a-pour-cible}(I) \wedge St_2.\text{a-pour-source}(I) \wedge St_2.\text{a-pour-cible}(I_2) \wedge I.a\ (E) \wedge \nexists\ I_i\ (I = I_i)$ | $(St_1, St_2) \in$ Stratégie, $E \in$ Etat, $I, I_1, I_2, I_i \in$ Intention, $C \in$ Carte pivot

AjouterStratégie

Toute stratégie dans une carte pivot doit être définie entre deux intentions, l'une jouant le rôle de source l'autre celui de cible. Il en est de même pour la stratégie qu'on ajoute. Par défaut ces deux intentions correspondent respectivement à *Démarrer* et *Arrêter*. I_1 et I_2 apparaissent comme paramètres optionnels dans le prédicat de l'opérateur (c'est-à-dire entre crochets).

> AjouterStratégie : $\text{Intention}^2 \rightarrow$ Stratégie
> AjouterStratégie $([I_1], [I_2]) = St \in C \wedge St.\text{a-pour-source}(I_1) \wedge St.\text{a-pour-cible}(I_2)$ | $St \in$ Stratégie, $I_1, I_2 \in$ Intention, $C \in$ Carte pivot

AjouterAffinement

Un lien d'affinement permet de lier une section Se et une carte pivot C, dans un modèle pivot. Afin de satisfaire **I6**, il ne doit pas exister un autre lien d'affinement A_i ayant pour source Se.

> AjouterAffinement : Section, Carte pivot $\rightarrow$ Section, Affinement, Carte pivot
> AjouterAffinement $(Se, C) = A \in M \wedge A.\text{a-pour-source}(Se) \wedge A.\text{a-pour-cible}(C) \wedge (\nexists\ A_i, A_i \neq A \wedge A_i.\text{a-pour-source}(Se))$ | $(A, A_i) \in$ Affinement, $C \in$ Carte pivot, $Se \in$ Section, $M \in$ Modèle pivot

AjouterCarte

L'opérateur *AjouterCarte* a pour but d'ajouter une carte pivot C dans un modèle pivot M. Afin de satisfaire le corollaire **C10** et **I6**, si C affine Se, une section de C_1, il ne doit pas exister un autre lien d'affinement ayant Se pour source.

> AjouterCarte : Modèle pivot → Carte pivot
> AjouterCarte (C) = C ∈ M ∧ A.a-pour-source(Se) ∧ C_1.est-composé-de(Se) ∧ (∄ A_i, $A_i \neq A$ ∧ A_i.a-pour-source(Se)) | (A, A_i) ∈ Affinement, (C, C_1) ∈ Carte pivot, Se ∈ Section, M ∈ Modèle pivot

AjouterClasse

Afin de vérifier le corollaire **C11**, la classe Cl qu'on ajoute doit être reliée au reste des classes. Il doit exister une relation R ayant pour source (respectivement pour cible) Cl et pour cible (respectivement pour source) Cl_1. L'invariant **I8** nous amène à préciser, dans la définition de l'opérateur *AjouterClasse*, qu'il n'existe pas d'autres classes Cl_i qui soit identique à Cl.

> AjouterClasse : Classe → Relation^2, Classe
> AjouterClasse (Cl_1) = Cl ∈ Mc ∧ (R.a-pour-source(Cl_1) ∧ R.a-pour-cible(Cl)) ∨ (R.a-pour-source(Cl) ∧ R.a-pour-cible(Cl_1)) ∧ ∄ Cl_i (Cl = Cl_i) | R ∈ Relation, (Cl, Cl_1, Cl_i) ∈ Classe, Mc ∈ Modèle de Classes

AjouterRelation

L'opérateur *AjouterRelation* permet d'ajouter une relation entre deux classes (Cl_1 et Cl_2 dans la définition de l'opérateur).

> AjouterRelation : Classe^2 → Relation,
> AjouterRelation (Cl_1, Cl_2) = R ∈ Mc ∧ R.a-pour-source(Cl_1) ∧ R.a-pour-cible(Cl_2) | R ∈ Relation, (Cl_1, Cl_2)∈ Classe, Mc ∈ Modèle de Classes

6.5.2 Supprimer

SupprimerIntention

L'exécution de l'opérateur *SupprimerIntention* met en jeu les corollaires **C1**, **C2** et **C3**. Pour les préserver, il faut que chaque stratégie St^s_i (respectivement St^c_j) ayant l'intention supprimée I comme intention source (respectivement intention cible) ait désormais une autre intention source I_k (respectivement intention cible I_m). Par défaut, l'intention *Démarrer* (respectivement *Arrêter*) peut être utilisée.

> SupprimerIntention : Intention → Carte pivot
> SupprimerIntention (I) = I ∉ C ∧ [∀ i, ∃ k, St^s_i.a-pour-source(I_k)] ∧ [∀ j, ∃ m St^c_j.a-pour-cible(I_m)] | (I, I_k, I_m) ∈ Intention, (St^s_i, St^c_j) ∈ Stratégie, C ∈ Carte pivot

SupprimerStratégie

Considérons que la section Se = (<I_s, I_c, St>) existe dans la carte pivot du As-Is. Afin de préserver les corollaires **C1**, **C2** et **C3**, l'opérateur *SupprimerStratégie* appliqué à la stratégie St impose l'existence de deux sections Se_1 = <I_s, I_1, St_1> et Se_2 = <I_2, I_c, St_2>. I_s et I_c reste donc respectivement intention source et intention cible de sections dans la carte pivot. Par

défaut I_1 et I_2 peuvent être respectivement Démarrer et Arrêter. I_s et I_c sont des paramètres optionnels et sont précisés entre crochets.

SupprimerStratégie : Stratégie, Intention^2 → Stratégie², Intention^2
SupprimerStratégie (St, [I_s], [I_c]) = St ∉ C ∧ St_1.a-pour-source(I_s) ∧ St_1.a-pour-cible(I_1) ∧ St_2.a-pour-source(I_2) ∧ St_2.a-pour-cible(I_c) | (St, St_1, St_2) ∈ Stratégie, (I_s, I_c, I_1, I_2) ∈ Intention, C ∈ Carte pivot

SupprimerAffinement

Cet opérateur permet de préciser que la section Se n'est plus affinée par la carte pivot C.

SupprimerAffinement: Affinement → Modèle pivot
SupprimerAffinement (A) = A ∉ M ∧ C ∉ M | A∈ Affinement, M ∈ Modèle pivot

SupprimerCarte

L'opérateur SupprimerCarte permet d'enlever une carte pivot C du modèle pivot M. Afin de satisfaire le corollaire **C10** et **I6**, toute carte section affinée Se_k doit être la source d'un unique lien d'affinement.

SupprimerCarte: Carte pivot → Modèle pivot
SupprimerCarte (C) = C ∉ M ∧ (∀ i, ∃ j, k, A_j.a-pour-source(Se_k) ∧ A_j.a-pour-cible(C_i) ∧ C_k.est-composé-de(Se_k)) ∧ (∄ A_m, A_m ≠ A_j ∧ A_m.a-pour-source(Se_k)) | (C, C_i, C_k) ∈ Carte pivot, (A_j, A_m) ∈ Affinement, M ∈ Modèle pivot

SupprimerClasse

La définition de l'opérateur *SupprimerClasse* précise que la classe Cl qui est supprimée n'appartient plus au modèle de classe après l'exécution de l'opérateur.

SupprimerClasse : Classe → Modèle de Classes
SupprimerClasse (Cl) = Cl ∉ Mc | (Cl) ∈ Classe, Mc ∈ Modèle de Classes

SupprimerRelation

L'exécution de l'opérateur *SupprimerRelation* met en jeu l'invariant **I7**. Pour le préserver, il faut que les classes Cl_1 et Cl_2 qui étaient source ou cible de cette relation soient, après application de l'opérateur source ou cible d'une autre relation. Cl_1 et Cl_2 sont des paramètres optionnels et sont précisés entre crochets.

SupprimerRelation : Relation → Modèle de Classes
SupprimerRelation (R, [Cl_1], [Cl_2]) = R ∉ M ∧ (R_1.a-pour-source(Cl_1) ∨ R_1.a-pour-cible(Cl_1)) ∧ (R_2.a-pour-source(Cl_2) ∨ R_2.a-pour-cible(Cl_2))] | (Cl_1, Cl_2) ∈ Classe, (R, R_1, R_2) ∈ Relation, Mc ∈ Modèle de Classes

6.5.3 AjouterComposant

AjouterSectionCarte

Afin de préserver l'invariant **I3** et être conforme au corollaire **C5**, l'opérateur *AjouterSectionCarte* doit garantir que la section ajoutée (Se = <I_s, I_c, St>) est connectée au reste de la carte. Ceci est réalisé en forçant l'existence de :

- une stratégie St_1 ayant I_s comme intention cible et une intention I_1 existant préalablement dans la carte pivot comme intention source et
- une stratégie St_2 ayant I_c comme intention source et une intention I_2 existant préalablement dans la carte pivot comme intention cible. Par défaut les intentions I_1 et I_2 peuvent respectivement correspondre à Démarrer et Arrêter.

L'invariant **C7** est respecté si I_s et I_c sont constituées d'au moins un état chacune.

Afin d'être conforme à **I4**, il ne doit pas exister dans la carte pivot d'intention I identique à I_s ou I_c avec I_s différente de I_c.

Enfin, afin de satisfaire l'invariant **C8**, la section Se a une pré-condition en effet, l'intention source I_s de cette section ne peut pas être l'intention *Démarrer* car l'intention *Démarrer* existe dans la carte pivot C avant l'application de l'opérateur.

I_1 et I_2 sont des paramètres optionnels et sont précisés entre crochets.

AjouterSectionCarte : Carte, Intention² → Section, Stratégie², Intention²
AjouterSectionCarte (C, [I_1], [I_2]) = Se ∈ C ∧ St_1.a-pour-source(I_1) ∧ St_1.a-pour-cible(I_s) ∧ St_2.a-pour-source(I_c) ∧ St_2.a-pour-cible(I_2) ∧ $I_1 \neq I_2$ ∧ ∄ I_i ($I_s = I_i \vee I_c = I_i$) ∧ I_s.a (E_1) ∧ I_s.a (E_2) ∧ Se.a(Pr) ∣ Se.est-composée-de(St) ∧ Se.est-composée-de(I_s) ∧ Se.est-composée-de(I_c), (St, St_1, St_2) ∈ Stratégie, (I_s, I_c, I_1, I_2, I_i) ∈ Intention, Se ∈ Section, (E_1, E_2) ∈ Etat, C ∈ Carte, Pr ∈ Pré-condition

6.5.4 Supprimer Composant

SupprimerSectionCarte

Quand une section Se (<I_s, I_c, St>) est supprimée d'une carte pivot C, ses trois composants (l'intention source I_s, l'intention cible I_c et la stratégie St) sont supprimés. Afin de préserver les invariants des cartes, l'opérateur *SupprimerSectionCarte* doit s'assurer que toute stratégie St^s_i ayant I_s ou I_c pour source, a une autre intention I_k de la carte pivot du To-Be comme source (à défaut l'intention *Démarrer*). De la même façon, les stratégies St^c_j ayant I_s ou I_c comme intention cible doivent, après le retrait de la section, avoir une intention I_m comme intention cible (à défaut *Arrêter*). {St^s_i} et {St^c_j} sont des paramètres optionnels et sont précisés entre crochets.

SupprimerSectionCarte : Section→ Carte
SupprimerSectionCarte (Se, [{St^s_i}], [{St^c_j}]) = Se ∉ C ∧ [∀ i, ∃ k, St^s_i.a-pour-source(I_k)] ∧ [∀ j, ∃ m St^c_j.a-pour-cible(I_m)] ∣ C ∈ Carte, Se ∈ Section, (I_k, I_m) ∈ Intention, (St^s_i, St^c_j) ∈ Stratégie

6.5.5 Joindre

JoindreEtatIntention

Cet opérateur a pour but d'ajouter un état associé à l'intention I. Afin de valider l'invariant **I4** et le corollaire **C13**, il est important qu'après l'ajout de l'état, I ne soit pas identique à une autre intention.

JoindreEtatIntention : Intention → Etat, Intention

JoindreEtatIntention (I) = I.a(E) $\wedge \nexists I_i$ ($I_i = I$) | (I, I_i) $\in$ Intention, E $\in$ Etat

JoindrePreCondition

Cet opérateur permet d'ajouter une pré-condition à une section Se. Après application de l'opérateur, le corollaire **C12** est satisfait dans la mesure où on ajoute une pré-condition, celle-ci n'est donc pas vide.

JoindrePrécondition : Section $\rightarrow$ PréCondition, Section
JoindrePrécondition (Se) = Se.a(Pr) | Se $\in$ Section, Pr $\in$ PréCondition

JoindrePostCondition

Cet opérateur permet d'ajouter une post-condition à une section Se.

JoindrePostcondition : Section $\rightarrow$ PostCondition, Section
JoindrePostcondition (Se) = Se.a(Po) | Se $\in$ Section, Po $\in$ PostCondition

JoindreRègleGestion

Cet opérateur permet d'ajouter une règle de gestion à une section Se.

JoindreRègleGestion : Section $\rightarrow$ RègleGestion, Section
JoindreRègleGestion (Se) = Se.a(Rg) | Se $\in$ Section, Rg $\in$ RègleGestion

JoindreActeur

Cet opérateur permet d'ajouter un acteur Ac à une section Se.

JoindreActeur : Section $\rightarrow$ Acteur, Section
JoindreActeur (Se) = Se.a(Ac) | Se $\in$ Section, Ac $\in$ Acteur

JoindreRessource

Cet opérateur permet d'ajouter une ressource Re à une section Se.

JoindreRessource : Section $\rightarrow$ Ressource, Section
JoindreRessource (Se) = Se.a(Re) | Se$\in$ Section, Re $\in$ Ressource

JoindreEtatClasse

Cet opérateur permet d'associer un nouvel état E à la classe Cl. Afin de satisfaire l'invariant **I8**, nous précisons également qu'il ne doit pas exister une autre classe Cl_i identique à Cl.

JoindreEtatClasse : Classe $\rightarrow$ Etat, Classe
JoindreEtatClasse (Cl) = Cl.a(E) $\wedge \nexists Cl_i$ ($Cl_i = Cl$) | (Cl, Cl_i) $\in$ Classe, E $\in$ Etat

JoindreAttribut

Cet opérateur permet d'associer un nouvel attribut At à la classe Cl. Afin de satisfaire l'invariant **I8**, nous précisons également qu'il ne doit pas exister une autre classe Cl_i identique à Cl.

JoindreAttribut : Classe $\rightarrow$ Attribut, Classe
JoindreAttribut (Cl) = Cl.a(At) $\wedge \nexists Cl_i$ ($Cl_i = Cl$) | (Cl, Cl_i) $\in$ Classe, At $\in$ Attribut

6.5.6 Enlever

EnleverEtatIntention

Cet opérateur permet d'enlever un état associé à l'intention I. Pour préserver le corollaire **C7**, il doit exister un autre état E_i associé à I après l'application de cet opérateur.

EnleverEtatIntention : Etat, Intention → Intention
EnleverEtatIntention (I, E) = ¬ I.a(E) ∧ I.a(E_i) | I ∈ Intention, (E, E_i) ∈ Etat

EnleverPreCondition

Cet opérateur permet de supprimer une pré-condition associée à une section Se. Après l'application de cet opérateur, pour satisfaire le corollaire **C8**, l'ensemble des pré-conditions doit être différent de l'ensemble vide sauf s'il s'agit d'une section ayant pour source l'intention *Démarrer*.

EnleverPrécondition : PréCondition, Section → Section
EnleverPrécondition (Se, Pr) = ¬ Se.a(Pr) ∧ (Se.a(Pr_i) ∨ Se.est-composé-de(Démarrer)) |
Se∈ Section, Pr_i ∈ PréCondition, St ∈ Stratégie, (I, Démarrer) ∈ Intention

EnleverPostCondition

Cet opérateur permet de supprimer une post-condition associée à une section Se.

EnleverPostcondition : PostCondition, Section → Section
EnleverPostcondition (Se, Po) = ¬ Se.a(Po) | Se ∈ Section, Po ∈ PostCondition

EnleverRègleGestion

Cet opérateur permet de supprimer une règle de gestion associée à une section Se.

EnleverRègleGestion : RègleGestion, Section → Section
EnleverRègleGestion (Se, Rg) = ¬ Se.a(Rg) | Se ∈ Section, Rg ∈ RègleGestion

EnleverActeur

Cet opérateur permet de supprimer un acteur associé à une section Se.

EnleverActeur : Acteur, Section → Section
EnleverActeur (Se, Ac) = ¬ Se.a(Ac) | Se ∈ Section, Ac ∈ Acteur

EnleverRessource

Cet opérateur permet de supprimer une ressource associée à une section Se.

EnleverRessource : Ressource, Section → Section
EnleverRessource (Se, Re) = ¬ Se.a(Re) | Se ∈ Section, Re ∈ Ressource

EnleverEtatClasse

Cet opérateur permet d'enlever un état associé à la classe Cl. Pour préserver l'invariant **I8**, il ne doit pas exister une autre classe Cl_i telle que Cl = Cl_i après l'application de cet opérateur.

EnleverEtatClasse : Etat, Classe → Classe
EnleverEtatClasse (Cl, E) = ¬ Cl.a(E) ∧ ∄ Cl_i (Cl_i = Cl) | (Cl, Cl_i) ∈ Classe, E ∈ Etat

EnleverAttribut

Cet opérateur permet d'enlever un attribut associé à la classe Cl. Pour préserver l'invariant **I8**, il ne doit pas exister une autre classe Cl_i telle que $Cl = Cl_i$ après l'application de cet opérateur.

EnleverAttribut : Attribut, Classe → Classe
EnleverAttribut (Cl, At) = $\neg$ Cl.a(At) $\wedge \nexists$ Cl_i ($Cl_i = Cl$) | (Cl, Cl_i) $\in$ Classe, At $\in$ Attribut

6.5.7 Renommer

RenommerIntention

Une intention a un nom qui est de type String. Renommer une intention consiste à changer la valeur de la chaîne de caractères qui définit son nom. Afin de satisfaire l'invariant **I4**, le nom de cette intention doit être unique.

RenommerIntention : Intention, String → Intention, String
RenommerIntention (I, N) = I.nom(N) $\wedge \nexists$ I_i ($N = N_i \wedge I_i$.nom(N_i) $\wedge I \neq I_i$) | N, $N_i \in$ String, (I, I_i) $\in$ Intention

RenommerStratégie

Une stratégie a un nom qui est de type String. Renommer une stratégie consiste à changer la valeur de la chaîne de caractères qui définit son nom.

RenommerStratégie : Stratégie, String → Stratégie, String
RenommerStratégie (St, N) = St.nom(N) | N $\in$ String, St $\in$ Stratégie

RenommerClasse

Une classe est associée à une chaîne de caractère (String) qui correspond à son nom. Afin de satisfaire l'invariant **I8**, le nom de cette classe doit être unique.

RenommerClasse : Classe, String → Classe, String
RenommerClasse (Cl, N) = Cl.nom(N) $\wedge \nexists$ Cl_i ($N = N_i \wedge Cl_i$.nom(N_i) $\wedge Cl \neq Cl_i$) | N, $N_i \in$ String, (Cl, Cl_i) $\in$ Classe

RenommerRelation

Une relation a un nom qui est de type String. Renommer une relation consiste à changer la valeur de la chaîne de caractères qui définit son nom.

RenommerRelation : Relation, String → Relation, String
RenommerRelation (R, N) = R.nom(N) | N $\in$ String, R $\in$ Relation

6.5.8 Remplacer

RemplacerIntention

Cet opérateur permet de remplacer une intention I par une autre intention I_n. Toute stratégie ayant pour intention source (respectivement cible) I a désormais I_n pour intention source (respectivement intention cible). Ceci permet de préserver les corollaires **C1**, **C2** et **C3**.

Afin de satisfaire **C7**, l'intention I_n doit avoir au moins un état.

I4 est préservé s'il n'existe pas une autre intention dans la carte pivot identique à I_n.

RemplacerIntention : Intention, {Stratégie}² → Intention
RemplacerIntention (I, [{St^s_i}], [{St^c_j}]) = I ∉ C ∧ I_n ∈ C ∧ [∀ i, St^s_i.a-pour-source(I_n)] ∧ [∀ j, St^c_j.a-pour-cible(I_n)] ∧ I_n.a(E) ∧ ∄ I_i (I_n = I_i) | (I, I_n, I_i) ∈ Intention, E ∈ Etat, C ∈ Carte

RemplacerStratégie

En appliquant cet opérateur, on remplace la stratégie St par une nouvelle stratégie St_n. Ainsi, pour satisfaire le corollaire **C1**, l'intention source de St devient l'intention source de St_n. Il en est de même pour l'intention cible de St qui devient l'intention cible de St_n.

RemplacerStratégie : Stratégie, Intention² → Stratégie
RemplacerStratégie (St, [I_s], [I_c]) = St ∉ C ∧ St_n ∈ C ∧ St_n.a-pour-source(I_s) ∧ St_n.a-pour-cible(I_c) | (St, St_n)∈ Stratégie, C ∈ Carte

RemplacerSection

En remplaçant la section Se (<I_s, I_c, St>) par Se_n (<I^s_n, I^c_n, St_n>), il faut pour préserver les invariants **I1**, **I2** et **I3** que toute stratégie St^s_i ayant I_s (respectivement I_c) pour source, ait l'intention I^s_n (respectivement I^c_n) de la carte pivot du To-Be comme source. Les stratégies St^c_j ayant I_s (respectivement I_c) comme intention cible doivent avoir l'intention I^s_n (respectivement I^c_n) comme intention cible. Les stratégies St^s_i et St^c_j sont des paramètres optionnels

De plus, I^s_n et I^c_n doivent être constituées chacune d'au moins un état et il ne doit pas exister d'intention I_i identique à I^s_n ou I^c_n.

RemplacerSection : Section, {Stratégie}² → Section
RemplacerSection (Se, [{St^s_i}], [{St^c_j}]) = Se ∉ C ∧ Se_n ∈ C ∧ [∀ i, St^s_i.a-pour-source(I^s_n) ∨ St^s_i.a-pour-source(I^c_n)] ∧ [∀ j, St^c_j.a-pour-cible(I^s_n) ∨ St^c_j.a-pour-cible(I^c_n)] ∧ I^s_n.a (E_1) ∧ ∄ I_i - (I^s_n = I_i) ∧ I^c_n.a (E_2) ∧ ∄ I_i (I^c_n = I_i) | Se_n.est-composée-de(St_n) ∧ Se_n.est-composée-de(I^s_n) ∧ Se_n.est-composée-de(I^c_n), St_n ∈ Stratégie, (I^c_n, I^s_n, I_i) ∈ Intention, (Se,Se_n) ∈ Section, (E_1, E_2) ∈ Etat, C ∈ Carte

RemplacerCarte

Cet opérateur permet de substituer la carte pivot C_n à C, dans un modèle pivot M. Après application de l'opérateur, afin de satisfaire **C10** et **I6**, toute carte pivot C_i autre que la carte pivot de haut niveau doit être la cible d'un unique lien d'affinement

RemplacerCarte : Carte pivot → Carte pivot
RemplacerCarte (C_n) = C ∉ M ∧ Cl_n ∈ M ∧ (∀ i, ∃ j, k, A_j.a-pour-source(Se_k) ∧ A_j.a-pour-cible(C_i) ∧ C_k.est-composé-de(Se_k)) ∧ (∄ A_m, ∧ A_m ≠ A_j ∧ A_m.a-pour-cible(C_i)) | (C, C_n, C_i, C_k) ∈ Carte, (A_j, A_m) ∈ Affinement, M ∈ Modèle pivot

RemplacerClasse

Cet opérateur permet de remplacer une classe Cl par une autre classe Cl_n. Afin de préserver l'invariant **I7**, Cl_n doit être reliée au reste du modèle de classes Mc. En d'autres termes, il doit exister une relation R ayant Cl_n pour source (ou pour cible) et une autre classe Cl_1 pour cible

(ou pour source). De plus, pour satisfaire l'invariant **I8**, le nom de la classe Cl_n doit être unique.

RemplacerClasse : Classe → Classe
RemplacerClasse (Cl) = Cl ∉ Mc ∧ C_n ∈ Mc ∧ (R.a-pour-source(Cl_1) ∧ R.a-pour-cible(Cl_n)) ∨ (R.a-pour-source(Cl_n) ∧ R.a-pour-cible(Cl_1)) ∧ ∄ Cl_i (Cl_n = Cl_i) | (Cl, Cl_n, Cl_i) ∈ Classe, R ∈ Relation, Mc ∈ Modèle de classes

RemplacerRelation

En appliquant cet opérateur, on remplace la relation R qui a pour source Cl_1 et pour cible Cl_2 par une nouvelle relation R_n.

RemplacerRelation : Relation, Classe² → Relation
RemplacerRelation (R, [Cl_1], [Cl_2]) = R ∉ Mc ∧ R_n.a-pour-source(Cl_1) ∧ R_n.a-pour-cible(Cl_2) | (R, R_n)∈ Relation, (Cl_1, Cl_2) ∈ Classe, Mc ∈ Modèle de classes

6.5.9 Fusionner

FusionnerIntention

Quand on fusionne deux intentions I_1 et I_2, l'intention I remplace I_1 et I_2 dans toutes les sections ayant initialement I_1 ou I_2 comme intention source ou intention cible. Ceci permet de préserver les invariants **I1**, **I2** et **I3**.

De plus, afin de satisfaire **C7** et **I4**, I est possède au moins un état et ne doit pas être identique à une autre intention de la carte.

FusionnerIntention : Intention², {Stratégie}² → Intention
FusionnerIntention (I_1, I_2, [{St^s_i}], [{St^c_j}]) = I_1 ∉ C ∧ I_2 ∉ C ∧ I ∈ C ∧ [∀ i, St^s_i.a-pour-source(I)] ∧ [∀ j, St^c_j.a-pour-cible(I)] ∧ I.a (E) ∧ ∄ I_i (I = I_i) | (I, I_1, I_2, I_i) ∈ Intention, E ∈ Etat, (St^s_i, Stcj) ∈ Stratégie, C ∈ Carte

FusionnerStratégie

La stratégie St est considérée comme le résultat d'une fusion de deux stratégies St_1 et St_2 si ces deux stratégies ont la même intention source I_s et la même intention cible I_c. Si cette condition n'est pas remplie, un autre opérateur devra être utilisé. Par exemple, si on est en présence de trois intentions, c'est l'opérateur *FusionnerSection* qui doit être appliqué.

FusionnerStratégie : Stratégie², Intention² → Stratégie
FusionnerStratégie (St_1, St_2, [I_s], [I_c]) = St_1 ∉ C ∧ St_2 ∉ C ∧ St ∈ C ∧ St.a-pour-source(I_s) ∧ St.a-pour-cible(I_c) | C ∈ Carte, (St, St_1, St_2) ∈ Stratégie, (I_s, I_c) ∈ Intention, C ∈ Carte

FusionnerSection

La fusion de deux sections Se_1 (<I^s_1, I^c_1, St_1>) et Se_2 (<I^s_2, I^c_2, St_2>) résulte en une section Se_r (<I^s_n, St_n, I^c_n>). Il existe une pré-condition à l'application de cet opérateur : I^s_1 doit être différente de I^s_2 ou I^c_1 doit être différente de I^c_2. Après application de l'opérateur :

- I^s_1 et I^s_2 sont remplacées par I^s_n
- I^c_1 et I^c_2 sont remplacées par I^c_n

dans chaque section ayant initialement I^s_1, I^s_2, I^c_1 ou I^c_2 comme intention source ou cible.

De cette façon, les invariants **I1**, **I2** et **I3** sont satisfaits.

De plus, afin de satisfaire **C7**, I^s_r et I^c_r sont chacune contituées d'au moins un état.

I4 est respecté s'il n'existe pas d'autre intention I_i identique à I^s_r ou I^c_r.

> FusionnerSection : Section², {Stratégie}² → Section, Intention², Stratégie
> FusionnerSection (Se_1, Se_2, [{St^s_i}], [{St^c_j}]) = $Se_1 \notin C \wedge Se_2 \notin C \wedge Se_n \in C \wedge [\forall i, St^s_i$.a-pour-source($I^s_n$)] $\wedge$ [$\forall j, St^c_j$.a-pour-cible(I^c_n)] $\wedge$ I^s_n.a (E_1) $\wedge \nexists I_i$ ($I^s_n = I_i$) $\wedge$ I^c_n.a (E_2) $\wedge \nexists I_i$ - ($I^c_n = I_i$) | Se_n.est-composé-de(I^s_n) $\wedge$ Se_n.est-composé-de(I^c_n) $\wedge$ Se_n.est-composé-de(St_n), (St_n, St^s_i, St^c_j) ∈ Stratégie, (I_i, I^s_n, I^c_n) ∈ Intention, (Se_n) ∈ Section, C ∈ Carte, (E_1, E_2) ∈ Etat

FusionnerCarte

La fusion de deux cartes C_1 et C_2 appartenant au modèle pivot M permet d'obtenir une unique carte pivot C_n. Afin de satisfaire l'invariant **I4** toute intention dans la carte pivot C_n doit être unique (c'est en particulier le cas pour les intentions *Démarrer* et *Arrêter*).

> FusionnerCarte : Carte pivot² → Carte pivot
> FusionnerCarte (C_1, C_2) = $C_1 \notin M \wedge C_2 \notin M \wedge C_n \in M \wedge [\forall i, (\nexists j, i \neq j \wedge I_i = I_j)] \wedge [\forall i, \exists j, k\ A_j$.a-pour-source($Se_k$) $\wedge$ A_j.a-pour-cible(C_i) $\wedge$ C_k.est-composé-de(Se_k) $\wedge$ ($\nexists A_m, A_m \neq A_j \wedge A_m$.a-pour-cible($C_i$))] | ($I_i$, I_j) ∈ Intention, (C_n, C_1, C_2, C_i, C_k) ∈ Carte, (A_j, A_m) ∈ Affinement, M ∈ Modèle pivot

FusionnerClasse

Quand on fusionne deux classes Cl_1 et Cl_2 par une classe Cl, il est important, afin de préserver les invariants **I7** et **I8**, que la classe Cl soit reliée au reste du graphe et qu'il n'existe pas une autre classe (Cl_i) identique à Cl.

> FusionnerClasse : Classe² → Classe
> FusionnerClasse (Cl_1, Cl_2) = $Cl_1 \notin$ Mc $\wedge$ $Cl_2 \notin$ Mc $\wedge$ Cl ∈ Mc $\wedge$ (R.a-pour-source(Cl_3) $\wedge$ R.a-pour-cible(Cl)) $\vee$ (R.a-pour-source(Cl) $\wedge$ R.a-pour-cible(Cl_3)) $\wedge \nexists$ Cl_i (Cl = Cl_i) | (Cl, Cl_1, Cl_2, Cl_3, Cl_i) ∈ Classe, R ∈ Relation, Mc ∈ Modèle de Classe.

FusionnerRelation

Pour que la relation R soit considérée comme le résultat d'une fusion de deux relations R_1 et R_2 il faut que ces deux relations aient la même classe source Cl_1 et la même classe cible Cl_2. Si cette condition n'est pas remplie, un autre opérateur devra être utilisé.

> FusionnerRelation : Relation², classe² → Relation
> FusionnerRelation (R_1, R_2, [Cl_1], [Cl_2]) = $R_1 \notin$ Mc $\wedge$ $R_2 \notin$ Mc $\wedge$ R ∈ Mc $\wedge$ R.a-pour-source(Cl_1) $\wedge$ R.a-pour-cible(Cl_2) | Mc ∈ Modèle de classes, (R, R_1, R_2) ∈ Relation, (Cl_1, Cl_2) ∈ Classe

6.5.10 Diviser

DiviserIntention

L'opérateur DiviserIntention permet de remplacer une intention I par deux intentions I_1 et I_2. Afin de satisfaire les invariants, toute stratégie St^c_j ayant pour cible, dans la situation initiale, l'intention I doit avoir, après application de l'opérateur, I_1 ou I_2 comme cible. Il en est de même pour les stratégies St^s_i qui avait initialement I comme intention source et doivent avoir désormais I_1 ou I_2 comme source. I_1 et I_2 doivent être différentes et chacune composée d'au moins un état.

DiviserIntention : Intention, {Stratégie}² → Intention²
DiviserIntention (I, {St^s_i}, {St^c_j}) = $I \notin C \wedge I_1 \in C \wedge I_2 \in C \wedge [\forall i, St^s_i.\text{a-pour-source}(I_1) \vee St^s_i.\text{a-pour-source}(I_2)] \wedge [\forall j, St^c_j.\text{a-pour-cible}(I_1) \vee St^c_j.\text{a-pour-cible}(I_2)] \wedge (St_k.\text{a-pour-source}(I_1) \wedge St_l.\text{a-pour-cible}(I_1)) \wedge (St_m.\text{a-pour-source}(I_2) \wedge St_n.\text{a-pour-cible}(I_2))\ I_1.a\ (E_1) \wedge I_2.a\ (E_2) \wedge \nexists I_i\ (I_1 = I_i \vee I_2 = I_i)$ | $(I_1, I_2, I_i) \in$ Intention, $(E_1, E_2) \in$ Etat, (St^s_i, St^c_j, St_k, St_l, St_m, St_n) ∈ Stratégie, C ∈ Carte

DiviserStratégie

La division d'une stratégie St d'une section Se (<I_s, I_c, St>) donne deux stratégies St_1 et St_2, qui permettent de définir deux sections Se_1 (<I_s, I_c, St_1>) et Se_2 (<I_s, I_c, St_2>).

DiviserStratégie : Stratégie, Intention² → Stratégie²
DiviserStratégie (St, I_s, I_c) = $St \notin C \wedge St_1 \in C \wedge St_2 \in C \wedge St_1.\text{a-pour-source}(I_s) \wedge St_2.\text{a-pour-source}(I_s) \wedge St_1.\text{a-pour-cible}(I_c) \wedge St_2.\text{a-pour-cible}(I_c)$ | (St_1, St_2) ∈ Stratégie, C ∈ Carte

DiviserSection

Une section Se (<I_s, I_c, St>) peut être divisée en deux sections Se_1 (<I^s_1, I^c_1, St_1>) et Se_2 (<I^s_2, I^c_2, St_2>). Pour préserver les invariants, il faut que :

- toute stratégie St^c_j ayant pour cible, dans la situation initiale, l'intention I_s (respectivement I_c), doit avoir après application de l'opérateur, dans la situation finale, I^s_1 ou I^s_2 (respectivement I^c_1 ou I^c_2) comme cible.
- toute stratégie St^s_i ayant pour source, dans la situation initiale, l'intention I_s (respectivement I_c), doit avoir après application de l'opérateur, dans la situation finale, I^s_1 ou I^s_2 (respectivement I^c_1 ou I^c_2) comme source.

DiviserSection : Section, {Stratégie}² → Section², Intention², Stratégie²
DiviserSection (Se, {St^s_i}, {St^c_j}) = $Se \notin C \wedge Se_1 \in C \wedge Se_2 \in C \wedge [\forall i\ (St^s_i.\text{a-pour-source}(I^s_1) \vee St^s_i.\text{a-pour-source}(I^s_2) \vee St^s_i.\text{a-pour-source}(I^c_1) \vee St^s_i.\text{a-pour-source}(I^c_2))] \wedge [\forall j\ (St^c_j.\text{a-pour-cible}(I^s_1) \vee St^c_j.\text{a-pour-cible}(I^s_2) \vee St^c_j.\text{a-pour-cible}(I^c_1) \vee St^c_j.\text{a-pour-cible}(I^c_2))]$ | $Se_1.\text{est-composé-de}(I^s_1) \wedge Se_1.\text{est-composé-de}(I^c_1) \wedge Se_1.\text{est-composé-de}(St_1) \wedge Se_2.\text{est-composé-de}(I^s_2) \wedge Se_2.\text{est-composé-de}(I^c_2) \wedge Se_2.\text{est-composé-de}(St_2)$, ($St_1$, St_2, St^s_i, St^c_j) ∈ Stratégie, (I^s_1, I^c_1, I^s_2, I^c_2) ∈ Intention, (Se_1, Se_2) ∈ Section, C ∈ Carte

DiviserCarte

Une carte pivot C peut être divisée en deux cartes C_1 et C_2. Pour préserver les invariants, il faut que C_1 et C_2 soient la cible d'un unique lien d'affinement.

DiviserCarte : Carte pivot → Carte pivot^2
DiviserCarte : $C \notin M \wedge C_1 \in M \wedge C_2 \in M \wedge A_i$.a-pour-source($Se_k$) $\wedge A_i$.a-pour-cible(C_1) $\wedge C_k$.est-composé-de(Se_k) $\wedge A_j$.a-pour-source(Se_m) $\wedge A_j$.a-pour-cible(C_2) $\wedge C_m$.est-composé-de(Se_m) $\wedge (\nexists A_p, A_p \neq A_i \wedge A_p$.a-pour-cible($C_1$)) $\wedge (\nexists A_q, A_q \neq A_j \wedge A_q$.a-pour-cible($C_2$)) | ($C, C_1, C_2, C_m, C_k$) ∈ Cartes, ($A_i, A_j, A_p, A_q$) ∈ Affinement, M ∈ Modèle pivot

DiviserClasse

Pour préserver les invariants **I7** et **I8**, il est nécessaire qu'après l'exécution de l'opérateur *DiviserClasse*, les classes Cl_1 et Cl_2 soient reliées au reste du graphe et qu'il n'existe pas une autre classe (Cl_k) identique à Cl_1 ou à Cl_2.

DiviserClasse : Classe → Classe²
DiviserClasse (Cl) = $Cl \notin Mc \wedge Cl_1 \in Mc \wedge Cl_2 \in Mc \wedge (R_1$.a-pour-source($Cl_1$) $\wedge R_1$.a-pour-cible(Cl_i)) $\vee (R_1$.a-pour-source(Cl_i) $\wedge R_1$.a-pour-cible(Cl_1)) $\wedge (R_2$.a-pour-source(Cl_2) $\wedge R_2$.a-pour-cible(Cl_j)) $\vee (R_2$.a-pour-source(Cl_j) $\wedge R_2$.a-pour-cible(Cl_2)) $\wedge Cl_1 \neq Cl_2 \wedge \nexists Cl_k$ ($Cl_k = Cl_1 \vee Cl_k = Cl_2$) | ($Cl, Cl_1, Cl_2, Cl_i, Cl_j, Cl_k$) ∈ Classe, ($R_1, R_2$) ∈ Relation, Mc ∈ Modèle de classes.

DiviserRelation

La division d'une relation R donne deux relations R_1 et R_2 ayant pour source Cl_1 et pour cible Cl_2.

DiviserRelation : Relation, Classe² → Relation²
DiviserRelation (R, Cl_1, Cl_2) = $R \notin Mc \wedge R_1 \in Mc \wedge R_2 \in Mc \wedge R_1$.a-pour-source($Cl_1$) $\wedge R_2$.a-pour-source(Cl_1) $\wedge R_1$.a-pour-cible(Cl_2) $\wedge R_2$.a-pour-cible(Cl_2) | (R_1, R_2) ∈ Association, (Cl_1, Cl_2) ∈ Classe, Mc ∈ Modèle de classes

6.5.11 ChangerOrigine

L'opérateur ChangerOrigine se décompose en deux opérateurs ChangerSource et ChangerCible qui permettent respectivement de changer la source ou la cible d'un élément lien.

ChangerIntentionSource

Cet opérateur permet de changer l'intention source I d'une stratégie St en la remplaçant par une autre I_n existant déjà dans la carte. En changeant l'origine de la stratégie, il est possible que l'invariant **I3** et les corollaires **C1**, **C2** et **C3** ne soient plus satisfaits. Ce problème est résolu en s'assurant qu'il existe dans cette carte pivot une stratégie St_1 ayant I comme source.

ChangerIntentionSource : Stratégie, Intention² → Stratégie
ChangerIntentionSource (St, [I], [I_n]) = ¬ St.a-pour-source(I) $\wedge$ St.a-pour-source(I_n) $\wedge$ St_1.a-pour-source(I) | (St, St_1) ∈ Stratégie, (I, I_n) ∈ Intention

ChangeIntentionCible

Cet opérateur permet de changer l'intention cible I d'une stratégie St en la remplaçant par une autre I_n existant déjà dans la carte. En changeant la cible de la stratégie, il est possible que

l'invariant **I3** et les corollaires **C1**, **C2** et **C3** ne soient plus satisfaits. Ce problème est résolu en s'assurant qu'il existe dans cette carte pivot une stratégie St_1 ayant I comme cible.

ChangerIntentionCible : Stratégie, Intention² → Stratégie
ChangerIntentionCible (St, [I], [I_n]) = ¬ St.a-pour-cible(I) ∧ St.a-pour-cible(I_n) ∧ St_1.a-pour-cible(I) | (St, St_1) ∈ Stratégie, , (I, I_n) ∈ Intention

ChangerClasseSource

Cet opérateur permet de changer la classe source Cl d'une relation R en la remplaçant par une autre Cl_n existant déjà dans le modèle de classe Mc. L'exécution de cet opérateur peut entraîner une violation de l'invariant **I7**. Ce problème est résolu en s'assurant qu'il existe dans ce modèle de classe une relation R_1 ayant Cl comme source ou cible.

ChangerClasseSource : Relation, Classe² → Relation
ChangerClasseSource (R, Cl, Cl_n) = ¬ R.a-pour-source(Cl) ∧ R.a-pour-source(Cl_n) ∧ [(R_1.a-pour-source(Cl) ∨ R_1.a-pour-cible(Cl)] | (Cl, Cl_n) ∈ Classe, (R, R_1) ∈ Relation

ChangerClasseCible

Cet opérateur permet de changer la classe cible Cl d'une relation R en la remplaçant par une autre Cl_n existant déjà dans le modèle de classes Mc. L'exécution de cet opérateur peut entraîner une violation de l'invariant **I7**. Ce problème est résolu en s'assurant qu'il existe dans ce modèle de classe une relation R_1 ayant Cl comme source ou cible.

ChangerClasseCible : Relation, Classe² → Relation
ChangerClasseCible (R, Cl, C_n) = ¬ R.a-pour-cible(Cl) ∧ R.a-pour-cible(Cl_n) ∧ [(R_1.a-pour-source(Cl_n) ∨ R_1.a-pour-cible(Cl_n)] | (Cl, Cl_n) ∈ Classe, (R, R_1) ∈ Relation

6.5.12 Retyper

RetyperIntention

Une Intention I dans la carte pivot As-Is peut être retypée en une stratégie St dans la carte pivot To-Be. Le prédicat de l'opérateur *RetyperIntention* assure que toute stratégie St^s_i (respectivement St^c_j) ayant initialement I pour source (respectivement pour cible) doivent avoir une autre intention source I_k (respectivement intention cible I_m). La nouvelle stratégie St doit avoir une intention source I_1 et une intention cible I_2.

RetyperIntention : Intention, {Stratégie}², {Intention}² → Stratégie, Intention²
RetyperIntention(I, {St^s_i}, {St^c_j}, {I^s_k}, {I^c_l}) = I ∉ C ∧ St ∈ C ∧ [∀ i, ∃ k, St^s_i.a-pour-source(I_k)] ∧ [∀ j, ∃ m St^c_j.a-pour-cible(I_m)] ∧ St.a-pour-source(I_1) ∧ St.a-pour-cible(I_2) | St ∈ Stratégie, (I_1, I_2, I_k, I_m) ∈ Intention, C ∈ Carte

RetyperStratégie

L'écart entre une carte pivot As-Is et une carte pivot To-Be peut correspondre au retypage d'une stratégie St en une intention I.

Afin de satisfaire les corollaires **C2** et **C3**, l'intention qui était la source (respectivement la cible) de St doit être l'intention source (respectivement l'intention cible) d'au moins une stratégie dans la carte pivot To-Be.

De plus, afin de préserver **C1**, la nouvelle intention I ne doit pas être isolée. Deux stratégies St_3 et St_4 ayant I comme source et comme cible doivent exister dans la carte pivot To-Be.

RetyperStratégie : Stratégie, Intention^2 → Intention, Stratégie^4
RetyperStratégie (St, I_s, I_c) = St ∉ C ∧ I ∈ C ∧ St_1.a-pour-source(I_s) ∧ St_1.a-pour-cible(I_1) ∧ St_2.a-pour-cible(I_c) ∧ St_2.a-pour-source(I_2) ∧ St_3.a-pour-source(I) ∧ St_3.a-pour-cible(I_3) ∧ St_4.a-pour-cible(I) ∧ St_4.a-pour-source(I_4) | (I, I_1, I_2, I_3, I_4) ∈ Intention, C ∈ Carte, (St, St_1, St_2, St_3, St_4) ∈ Stratégie

RetyperPaquet

Il est possible de retyper le Paquet, qui existe entre deux sections Se_1 <I_s, I_c, St_1> et Se_2 <I_s, I_c, St_2>, en Segment.

RetyperPaquet : Section^2, Paquet → Section^2, Segment
RetyperPaquet (Se_1, Se_2, P) = P ∉ C ∧ Sg ∈ C ∧ Sg.a-pour-source(Se_1) ∧ Sg.a-pour-cible(Se_2) | (Se_1, Se_2) ∈ Section, C ∈ Carte, P ∈ Paquet, Sg ∈ Segment

RetyperSegment

Il est possible de changer le lien Segment qui existe entre deux sections Se_1 <I_s, I_c, St_1> et Se_2 <I_s, I_c, St_2>, c'est-à-dire le changer en Paquet.

RetyperSegment : Section^2, Segment → Section^2, Paquet
RetyperSegment (Se_1, Se_2, Sg) = Sg ∉ C ∧ P ∈ C ∧ P.a-pour-source(Se_1) ∧ P.a-pour-cible(Se_2) | (Se_1, Se_2) ∈ Section, C ∈ Carte, P ∈ Paquet, Sg ∈ Segment

RetyperClasse

L'opérateur *RetyperClasse* permet de transformer une classe en relation. Après l'exécution de cet opérateur, la classe Cl est retypée en relation R. R doit avoir pour source un une classe Cl_1 et pour cible une classe Cl_2.

RetyperClasse : Relation, Classe → Relation
RetyperClasse (Cl) = Cl ∉ Mc ∧ R ∈ Mc ∧ R.a-pour-source(Cl_1) ∧ R.a-pour-cible(Cl_2) | R ∈ Relation, (Cl, Cl_1, Cl_2) ∈ Classe, Mc ∈ Modèle de classes

RetyperRelation

L'opérateur *RetyperRelation* permet de transformer une relation en classe. Après l'exécution de cet opérateur, il est important, afin de préserver les invariants **I7** et **I8**, que la classe Cl soit reliée au reste du graphe et qu'il n'existe pas une autre classe (Cl_i) identique à Cl.

RetyperRelation : Classe → Relation
RetyperRelation (Cl) = R ∉ Mc ∧ Cl ∈ Mc ∧ [∀ i, ∃(j, k) (R_j.a-pour-source(Cl_i) ∧ R_j.a-pour-cible(Cl_k)) ∨ (R_j.a-pour-source(Cl_k) ∧ R_j.a-pour-cible(Cl_i))] ∧ ∄ Cl_m (Cl = Cl_m) | (Cl, Cl_i, Cl_k, Cl_m) ∈ Classe, (R, R_j) ∈ Relation, Mc ∈ Modèle de Classe.

6.5.13 Modifier

ModifierEtatIntention

Cet opérateur a pour but de modifier des états associés à l'intention I. Afin de valider le corollaire **C13**, il est important qu'après cette modification d'états, I ne soit pas identique à une autre intention.

ModifierEtatIntention : Intention, Etat → Intention, Etat
ModifierEtatIntention (I, E) = I.a(E) ∧ ∄ I_i (I_i = I), | (I, I_i) ∈ Intention, E ∈ Etat

ModifierPreCondition

Cet opérateur modifie une pré-condition associée à la section Se. Après l'application de cet opérateur, pour satisfaire le corollaire **C8**, l'ensemble des pré-conditions doit être différent de l'ensemble vide sauf s'il s'agit d'une section ayant pour source l'intention *Démarrer*.

ModifierPrécondition : Section, Pré-condition → Section, PréCondition
ModifierPrécondition (Se, Pr) = Se.a(Pr) ∧ ∃ Pr_i, Se.a(Pr_i) ∨ Se.est-composé-de(Démarrer) |
Se∈ Section, Pr_i ∈ PréCondition, St ∈ Stratégie, (I, Démarrer) ∈ Intention

ModifierPostCondition

Cet opérateur modifie une post-condition de la section Se.

ModifierPostcondition : Section, PostCondition → Section, PostCondition
ModifierPostcondition (Se, Po) = Se.a(Po) | Se∈ Section, Po ∈ PostCondition

ModifierRègleGestion

Cet opérateur modifie une règle de gestion associée à la section Se.

ModifierRègleGestion : Section, RègleGestion → Section, RègleGestion
ModifierRègleGestion (Se, Rg) = Se.a(Rg) | Se∈ Section, Rg ∈ RègleGestion

ModifierActeur

Cet opérateur modifie un acteur de la section Se.

ModifierActeur : Section, Acteur → Section, Acteur
ModifierActeur (Se, Ac) = Se.a(Ac) | Se∈ Section, Ac ∈ Acteur

ModifierRessource

Cet opérateur modifie une ressource associée à la section Se.

ModifierRessource : Section, Ressource → Section, Ressource
ModifierRessource (Se, Re) = Se.a(Re) | Se∈ Section, Re ∈ Ressource

ModifierDésignationCarte

Cet opérateur modifie la désignation associée à la carte pivot C, si C est la carte pivot de haut niveau c'est-à-dire si C n'est la cible d'aucun lien d'affinement.

ModifierDésignationCarte : Carte pivot, DésignationCarte → Carte pivot, DésignationCarte

ModifierDésignationCarte (C, Dc) = C.a(Dc) ∧ ¬ A.a-pour-cible(C) | C ∈ Carte pivot, Dc ∈ DésignationCarte, A ∈ Affinement

ModifierEtatClasse

Cet opérateur a pour but de modifier des états associés à la classe Cl. Afin de valider l'invariant **I8**, il est important qu'après cette modification d'états Cl ne soit pas identique à une autre classe.

ModifierEtatClasse : Classe, Etat → Classe, Etat
ModifierEtatClasse (Cl, E) = Cl.a(E) ∧ ∄ Cl_i (Cl_i = Cl) | (Cl, Cl_i) ∈ Classe, E ∈ Etat

ModifierAttribut

Cet opérateur a pour but de modifier des attributs associés à la classe Cl. Afin de valider l'invariant **I8**, il est important qu'après cette modification d'états Cl ne soit pas identique à une autre classe.

ModifierAttribut : Classe, Attribut→ Classe, Attribut
ModifierAttribut (Cl, At) = Cl.a(At) ∧ ∄ Cl_i (Cl_i = Cl) | (Cl, Cl_i) ∈ Classe, At ∈ Attribut

6.6. Etape 6 : Vérification des différentes qualités

La typologie spécifique pour le méta-modèle pivot est consistante, complète, correcte et sémantiquement riche. Nous le démontrons dans cette section.

6.6.1 Consistance

Dans cette section, nous prouvons que la typologie spécifique du Tableau 22 est consistante.

Vérifier la consistance, c'est s'assurer qu'un même écart ne se traduit pas par deux opérateurs différents. L'analyse des opérateurs permet d'affirmer que la typologie présentée au Tableau 22 est consistante.

6.6.2 Complétude

On a vu, au niveau générique, que la typologie générique est complète car elle subsume tout type de changement. Les six opérateurs Joindre, Enlever, Ajouter, Supprimer, AjouterComposant, SupprimerComposant nécessaires pour démontrer la complétude au niveau générique ont été instanciés pour chacun des éléments clé du méta-modèle pivot. Certains des opérateurs spécifiques résultants ont été retirés de la typologie générique, soit parce qu'ils n'avaient pas de sens soit parce qu'un autre opérateur permettait de satisfaire le même écart. Dans ce dernier cas, un des opérateurs a été conservé dans la typologie afin que l'exigence d'évolution puisse être exprimée.

Ainsi, (1) les opérateurs *SupprimerIntention*, *SupprimerStratégie*, *SupprimerSection*, *SupprimerCarte*, *SupprimerClasse,* *SupprimerRelation,* *EnleverEtatIntention*, *EnleverPreCondition*, *EnleverPostCondition*, *EnleverActeur*, *EnleverRessource*, *EnleverRègleGestion*, *EnleverEtatClasse* et *EnleverAttribut* permettent de réduire une carte pivot ou un modèle pivot à leur racine et (2) les opérateurs *AjouterIntention*,

AjouterStratégie, *AjouterSection*, *AjouterCarte*, *AjouterClasse,* *AjouterRelation,* *JoindreEtatIntention*, *JoindrePreCondition*, *JoindrePostCondition*, *JoindreActeur*, *JoindreRessource*, *JoindreRègleGestion*, *JoindreEtatClasse* et *JoindreAttribut* permettent de construire une carte pivot et un modèle pivot à partir de leur racine.

La typologie spécifique associée au méta-modèle pivot est donc complète.

6.6.3 Minimalité

La typologie du Tableau 22 n'est pas minimale. Elle contient, par exemple les opérateurs *FusionnerIntention* ou *DiviserStratégie*. Cependant, nous avons mis en exergue les opérateurs formant l'ensemble minimal à savoir : *SupprimerIntention*, *SupprimerStratégie*, *SupprimerSection*, *SupprimerCarte*, *SupprimerClasse,* *SupprimerRelation,* *EnleverEtatIntention*, *EnleverPreCondition*, *EnleverPostCondition*, *EnleverActeur*, *EnleverRessource*, *EnleverRègleGestion*, *EnleverEtatClasse,* *EnleverAttribut,* *AjouterIntention*, *AjouterStratégie*, *AjouterSection*, *AjouterCarte*, *AjouterClasse,* *AjouterRelation,* *JoindreEtatIntention*, *JoindrePreCondition*, *JoindrePostCondition*, *JoindreActeur*, *JoindreRessource*, *JoindreRègleGestion*, *JoindreEtatClasse* et *JoindreAttribut*.

Chacun de ces opérateurs instancie un opérateur générique de l'ensemble minimal et réciproquement tous les opérateurs génériques de l'ensemble minimal sont instanciés (sous réserve que l'opérateur spécifique n'exprime pas le même écart qu'un autre opérateur spécifique).

Cet ensemble formé de 28 opérateurs correspond donc à l'ensemble minimal.

6.6.4 Correction

La propriété de correction est vérifiée pour la typologie d'opérateurs associée au méta-modèle pivot compte tenu de la définition même des opérateurs. En effet, le prédicat de chacun des opérateurs est construit de façon à satisfaire les invariants du méta-modèle pivot et donc à vérifier la propriété de correction. En d'autres termes, le prédicat garantit que l'opérateur laisse la carte pivot To-Be dans un état correct préservant les invariants. Par exemple, l'opérateur *AjouterIntention* est défini comme suit :

> AjouterIntention : Intention^2 $\rightarrow$ Stratégie^2, Intention
> AjouterIntention ($[I_1]$, $[I_2]$) = $I \in C \wedge St_1$.a-pour-source(I_1) $\wedge$ St_1.a-pour-cible(I) $\wedge$ St_2.a-pour-source(I) $\wedge$ St_2.a-pour-cible(I_2) $\wedge$ I.a (E) $\wedge \nexists I_i$ ($I = I_i$) | (St_1, St_2) $\in$ Stratégie, E $\in$ Etat, I, I_1, I_2, $I_i \in$ Intention, C $\in$ Carte

L'addition d'une nouvelle intention I dans la future carte pivot assure qu'il existe au moins une section (<I_1,I,St_1 >) dans laquelle I est l'intention cible et une section (<I, I_2, St_2 >) dans laquelle *I* est l'intention source. De plus I doit avoir au moins un état et il ne doit pas exister dans la carte pivot une autre intention I_i identique à I. Ainsi, le prédicat préserve tous les invariants susceptibles d'être insatisfaits après l'application de cet opérateur. Il en est de même pour tous les autres opérateurs que nous avons définis.

La propriété de correction est donc vérifiée par construction.

6.6.5 Richesse Sémantique

La typologie générique des écarts est considérée comme sémantiquement riche car elle permet d'exprimer tout type de changement en n'utilisant qu'un seul opérateur. La typologie spécifique associée aux cartes est également sémantiquement riche. En effet, seuls les opérateurs qui n'ont pas de sens ou qui expriment la même exigence d'évolution qu'un autre opérateur ont été retirés de la liste initiale obtenue par instanciation de la typologie générique. Ainsi, seul les opérateurs qui ont du sens et représentent un changement ont été conservés. Chaque type de changement pouvant avoir lieu dans une carte pivot peut donc s'exprimer par un unique opérateur. La typologie spécifique du Tableau 22 est donc sémantiquement riche.

Il est important de noter que l'ordre dans lequel les propriétés ont été vérifiées n'a pas d'importance. En effet, les opérateurs supprimés lors de la vérification de la consistance n'ont aucun impact sur la complétude, la correction ou la richesse sémantique.

Chacune des qualités retenues étant satisfaite, le processus de génération d'une typologie spécifique d'écarts associée à la partie intentionnelle du méta-modèle pivot est terminé. Il nous a permis d'obtenir la typologie du Tableau 22.

7. Discussion et conclusion

Dans ce chapitre, nous avons proposé une approche pour identifier des opérateurs permettant d'exprimer des exigences d'évolution. Cette approche repose sur l'existence d'un méta-modèle et d'une typologie génériques. Elle permet de construire une nouvelle typologie d'opérateurs associée à un méta-modèle spécifique ou de modifier une typologie existante.

Le processus que nous avons défini dans ce chapitre permet de générer de façon systématique une typologie spécifique satisfaisant les critères de qualité de complétude, correction, consistance et de richesse sémantique. De cette façon :

1. tout changement peut être exprimé par l'ensemble des opérateurs de la typologie ;
2. l'application de chaque opérateur laisse le système d'information dans un état cohérent sans introduire de nouvelles erreurs ;
3. les définitions des opérateurs sont claires et non ambiguës et
4. chaque type de changement peut être exprimé en n'utilisant qu'un seul opérateur.

L'illustration de ce processus pour définir une typologie spécifique associée au méta-modèle pivot a montré sa relative simplicité et son caractère systématique.

Nous verrons au chapitre suivant comment la typologie spécifique d'opérateurs associée au méta-modèle pivot est utilisée dans le processus d'ingénierie d'alignement.

CHAPITRE 8 : UNE DEMARCHE GUIDEE D'INGENIERIE D'ALIGNEMENT

1. Introduction

La démarche ACEM (**A**lignment **C**orrection and **E**volution **M**ethod) a pour objet de faire évoluer le système d'information et les processus d'entreprise pour prendre en compte de nouveaux besoins ou mettre en œuvre des actions correctives permettant d'améliorer le degré d'alignement courant. Cette méthode repose sur l'utilisation d'un modèle pivot pour matérialiser la relation d'alignement.

La Figure 69[2] donne une vue générale de la méthode ACEM. Celle-ci distingue les modèles de système, les modèles de processus et le modèle pivot. Deux moments sur l'axe du temps sont également identifiés :

- le présent, qui fait référence à la situation de l'organisation avant la prise en compte des exigences d'évolution. Cette situation rassemble les modèles As-Is de système et de processus qui ont un certain degré d'alignement.
- le futur, qui représente les modèles de système et de processus après la prise en compte des exigences d'évolution.

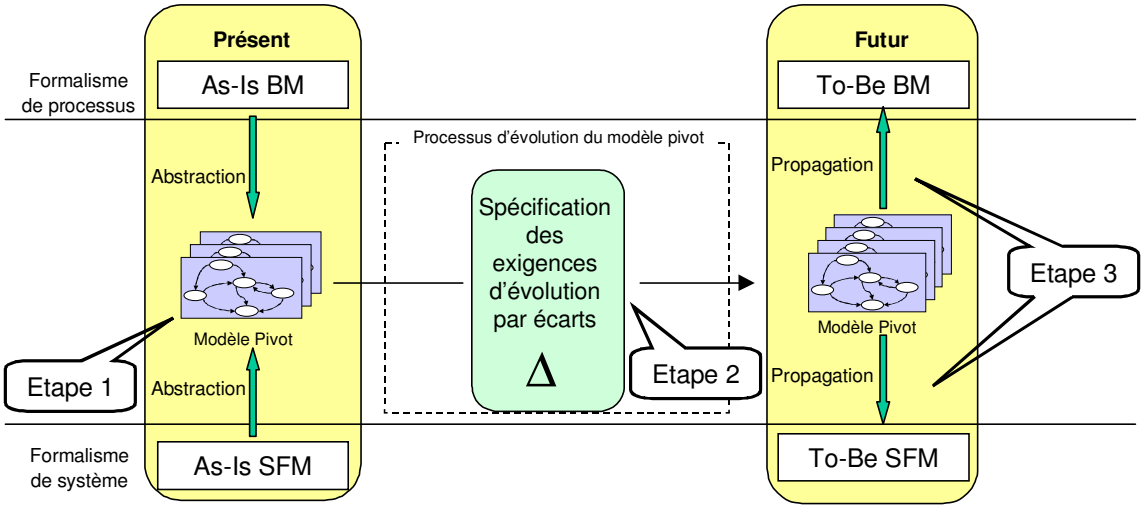


Figure 69 : Vue générale de l'approche

Le processus de la méthode ACEM se décompose en trois étapes :

1. la construction du modèle pivot ;
2. l'évolution du modèle pivot par spécification des exigences d'évolution ;
3. la propagation des écarts sur les modèles de système et de processus.

[2] Cette figure est identique à la Figure 45 du chapitre 5.

Le processus de la méthode ACEM est vu comme une succession de décisions qui conduisent à la transformation du produit qu'est le modèle pivot. Ce processus est décrit en utilisant le méta-modèle de Carte permettant de guider cette prise de décision. Une carte est alors considérée comme une structure de navigation contenant un nombre fini de chemins où aucun n'est recommandé « a priori », mais où chacun est choisi de manière *dynamique*. La sélection d'une stratégie se fait au fur et à mesure de la réalisation des intentions, en fonction de l'état du produit. L'ingénieur d'alignement sélectionne une intention pour progresser dans le processus et réalise l'intention sélectionnée à l'aide des *directives* associées à la carte. Ce chapitre présente l'ensemble des directives qui guident le processus ACEM.

Le reste du chapitre est organisé de la façon suivante : la section 2 présente le méta-modèle de Carte comme méta-modèle de processus de la méthode ACEM. La section 3 décrit la carte ACEM de haut niveau. La section 4 présente comment démarrer le processus de la méthode ACEM. La section 5 présente les différentes directives qui correspondent à la construction du modèle pivot. Les sections 6 et 7 présentent les directives correspondant à la découverte des exigences d'évolution et à l'arrêt du processus. La section 8 correspond à la conclusion du chapitre.

2. Méta-modèle de Carte comme méta-modèle de Processus

Le méta-modèle de Carte (Figure 70) permet de représenter le processus de la méthode ACEM. Une telle utilisation du méta-modèle de Carte a été décrite dans [Ralyte01]. Nous faisons ici un rappel.

Le méta-modèle de Carte est composé de deux parties, une *partie intentionnelle* (représentée en gris à la Figure 70) et *une partie directive*. Grâce aux concepts d'intention et de stratégie, la partie intentionnelle permet de différencier le but à atteindre de la façon de l'atteindre. Une section correspond alors au triplet constitué d'une intention source, d'une intention cible et d'une stratégie exprimant la façon d'atteindre un but précis à partir d'une situation donnée. Cette partie est semblable à la partie intentionnelle du méta-modèle pivot. La partie directive aide à naviguer dans la carte et à réaliser les différentes sections.

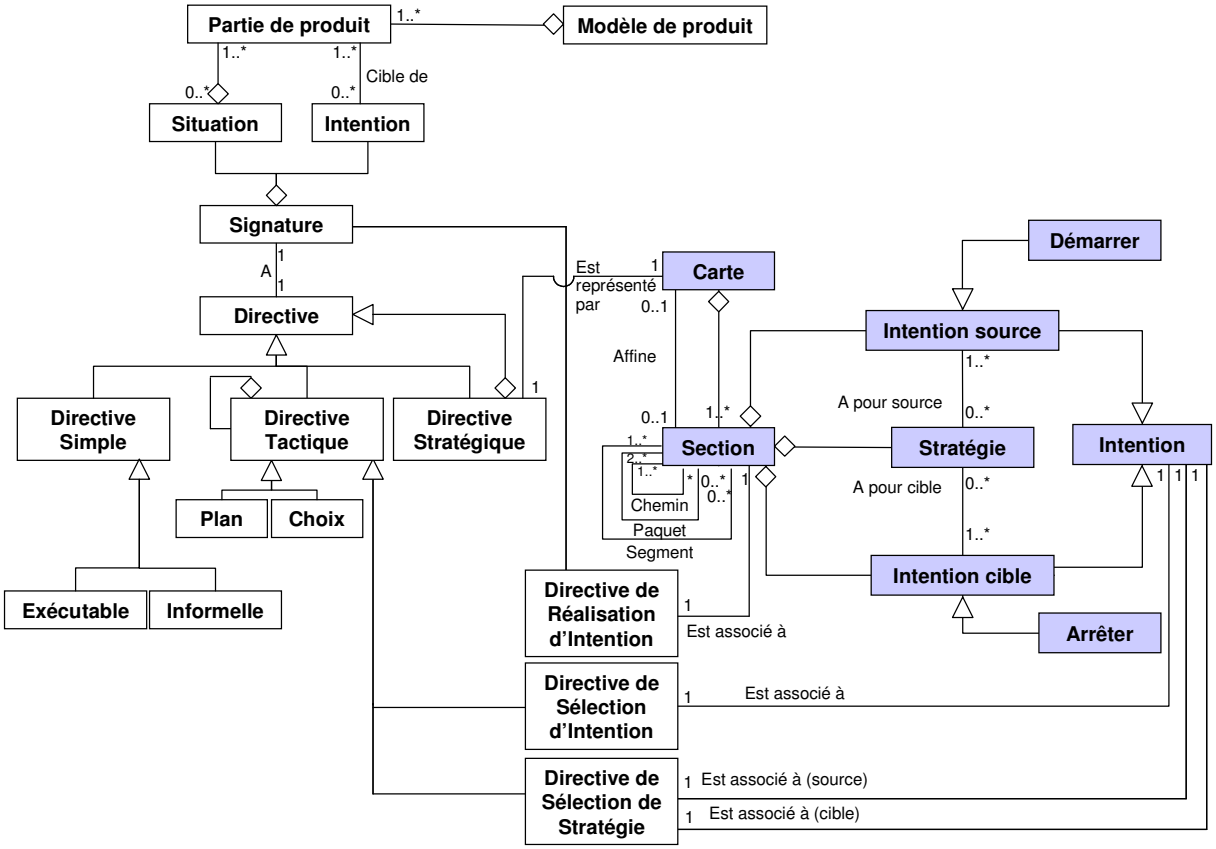


Figure 70 : Méta-modèle de Carte vu comme méta-modèle de processus

Le méta-modèle de Carte permet de représenter des processus d'ingénierie sous forme de cartes composées de sections. Ces processus d'ingénierie que l'ingénieur d'alignement exécute pour réaliser ses objectifs sont vus comme une succession de décisions qui conduisent à la transformation du produit. Le rôle de la carte comme modèle de processus est de guider cette prise de décision grâce à des directives.

Dans la suite de cette section, nous présentons les différentes directives et leurs rôles respectifs.

2.1. Notion de directive

Une directive est définie comme « un ensemble d'indications sur la façon de procéder pour réaliser un objectif ou exécuter une activité » [Robert00].

Plus précisément, selon [Ralyté01] une directive (i) définit la connaissance du domaine pour aider l'ingénieur d'alignement à réaliser une intention dans une situation donnée et (ii) préconise un processus à suivre.

Toute directive possède une *signature* et un *corps*. La *signature* caractérise les conditions dans lesquelles la directive peut être appliquée et le résultat qu'elle permet d'obtenir. Le *corps* définit la démarche à suivre pour satisfaire ce résultat.

La *signature* de la directive est définie par un couple <situation, intention>. Les conditions d'application de la directive sont précisées dans sa *situation*. Le résultat auquel elle permet

d'aboutir est indiqué dans son *intention*. Par conséquent, chaque directive s'applique dans une situation particulière pour satisfaire une intention particulière.

La situation dans la signature d'une directive identifie une partie de produit en cours de développement nécessaire à la satisfaction de l'intention de la directive. Chaque partie de produit référencée dans la situation est un élément du modèle de produit de la méthode (Figure 70). Il peut s'agir d'un élément de produit atomique, d'une association de plusieurs éléments de produit ou même du modèle de produit de la méthode en entier.

Une intention exprime un but que l'ingénieur des besoins souhaite atteindre en appliquant la directive. La cible de ce but est composée d'une ou plusieurs parties de produit. Par exemple, « Construire le modèle pivot » est une des intentions que l'on peut exprimer dans le processus ACEM.

Le *corps* d'une directive explicite le guidage fourni par celle-ci. Il contient un ensemble de recommandations définissant comment procéder pour satisfaire l'intention définie dans la signature de la directive. Il propose une ou plusieurs démarches à suivre pour aboutir au résultat attendu.

2.2. Types de directives

Il existe différents types de directives. D'un côté la taille des directives peut varier de la description d'une action atomique jusqu'à celle d'une démarche complète. D'un autre côté, le but d'une directive peut être d'aider à naviguer dans la carte ou d'exécuter une section.

La première caractérisation conduit à distinguer *directive simple*, *directive tactique*, et *directive stratégique*. La deuxième permet de différencier *directive de réalisation d'intention*, *directive de sélection d'intention* et *directive de sélection de stratégie*.

2.2.1 Typologie des directives selon la taille

On distingue, selon la taille, trois types de directives : *directive simple*, *directive tactique*, et *directive stratégique* [Ralyté01]. Dans chacun de ces cas, la complexité et la manière dont les directives sont exprimées varient.

2.2.1.1 Directive simple

Une directive simple est une directive qui ne se décompose pas en sous-directives. Elle constitue donc un élément atomique dans la démarche d'une méthode. Elle est soit informelle soit exécutable. Une directive informelle explique de manière narrative comment procéder pour obtenir le produit cible tandis qu'une directive exécutable propose une action à exécuter, soit avec un outil soit de manière manuelle. La Figure 71 présente un exemple de directive simple.

<(modèle As-Is='à jour'), Identifier carte à partir du produit> :

Cette directive s'appuie sur la structure et le contenu du modèle du système ou des processus pour identifier, à partir du produit, un objectif que le système permet de satisfaire. L'identification de la carte consiste à attribuer une désignation à la carte. Cette désignation précise l'objectif que la carte doit satisfaire dans son ensemble.

Figure 71 : Exemple de directive simple

2.2.1.2 Directive tactique

Une directive tactique est une directive complexe utilisant une structure d'arbre pour relier ses sous-directives. Il existe deux types de directives tactiques : les *directives choix* et les *directives plan*.

- Une directive choix est un ensemble de sous-directives tactiques ou simples liées par un lien OU. Chaque sous-directive représente une manière différente pour satisfaire l'objectif de la directive tactique. L'exécution d'une directive tactique de ce type consiste à choisir l'une de ses sous-directives, la plus adaptée à la situation donnée, et à l'exécuter. Ces sous-directives alternatives peuvent être tactiques ou simples. Des *critères de choix* aident l'ingénieur d'alignement à choisir l'alternative la plus appropriée aux caractéristiques de la situation courante.

  La Figure 72 présente un exemple de directive choix. Dans cet exemple, la directive <(*modèle pivot = 'à jour'*), *Identifier les dysfonctionnements du système*> est affinée par quatre alternatives :

  <(*modèle pivot = 'à jour'*), *Interviewer les parties prenantes*>,
  <(*modèle pivot = 'à jour'*), *Analyser l'utilisation du système*>,
  <(*modèle pivot = 'à jour'*), *Analyser le modèle pivot*> ou
  <(*modèle pivot = 'à jour'*), *Analyser le modèle du système*>.

  On associe un argument à chaque alternative pour motiver le choix de cette alternative plutôt que les autres. Pour choisir l'alternative <(*modèle pivot = 'à jour'*), *Analyser le modèle pivot*>, l'argument (a3) (le modèle pivot permet de révéler des dysfonctionnements) doit être évalué ; s'il est vérifié, la directive associée peut être exécutée.

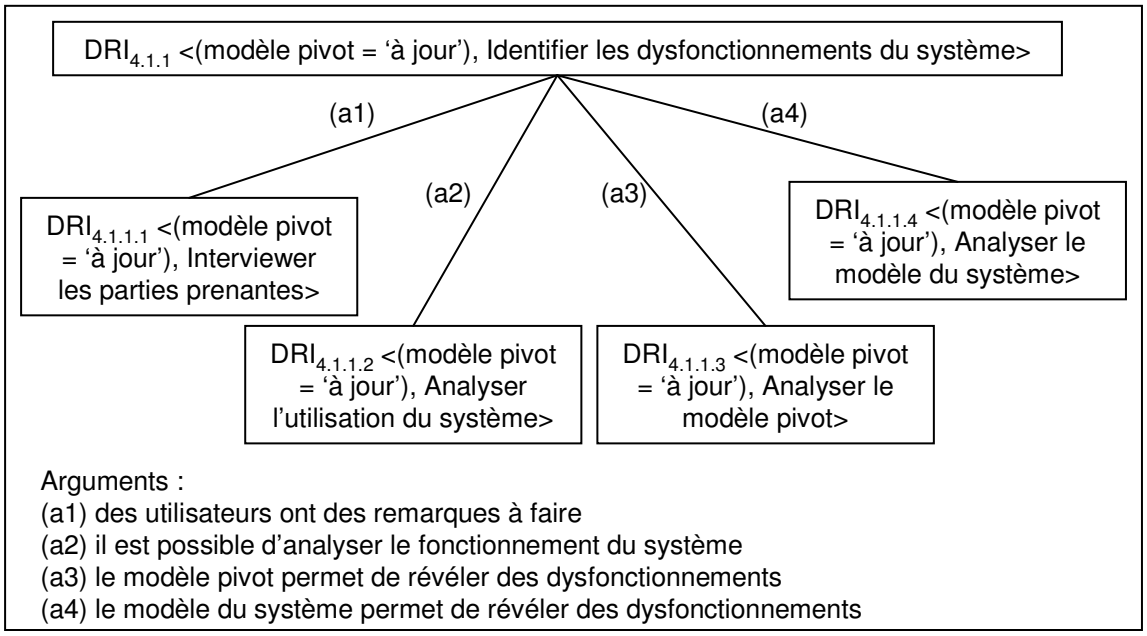


Figure 72 : Exemple de directive choix

- Une directive plan est un ensemble de sous-directives reliées par un lien ET. L'ordre d'exécution des sous-directives est défini dans un *graphe de précédence*. Les nœuds du graphe sont des sous-directives (les composantes du plan), alors que les arcs, appelés liens de précédence, représentent des transitions ordonnées ou parallèles entre directives. Les critères de choix attachés aux liens permettent d'aider l'ingénieur à choisir le chemin à suivre pour l'exécution du plan. Lorsque le graphe est trivial ou séquentiel, il n'est pas représenté.

La Figure 73 présente un exemple de directive plan. La directive $DRI_{4.1.3}$ <(*dysfonctionnement = 'identifié'*), *Eliciter des exigences d'évolution à partir des dysfonctionnements*> se décompose en cinq directives dont l'ordre d'exécution est précisé dans le graphe de précédence.

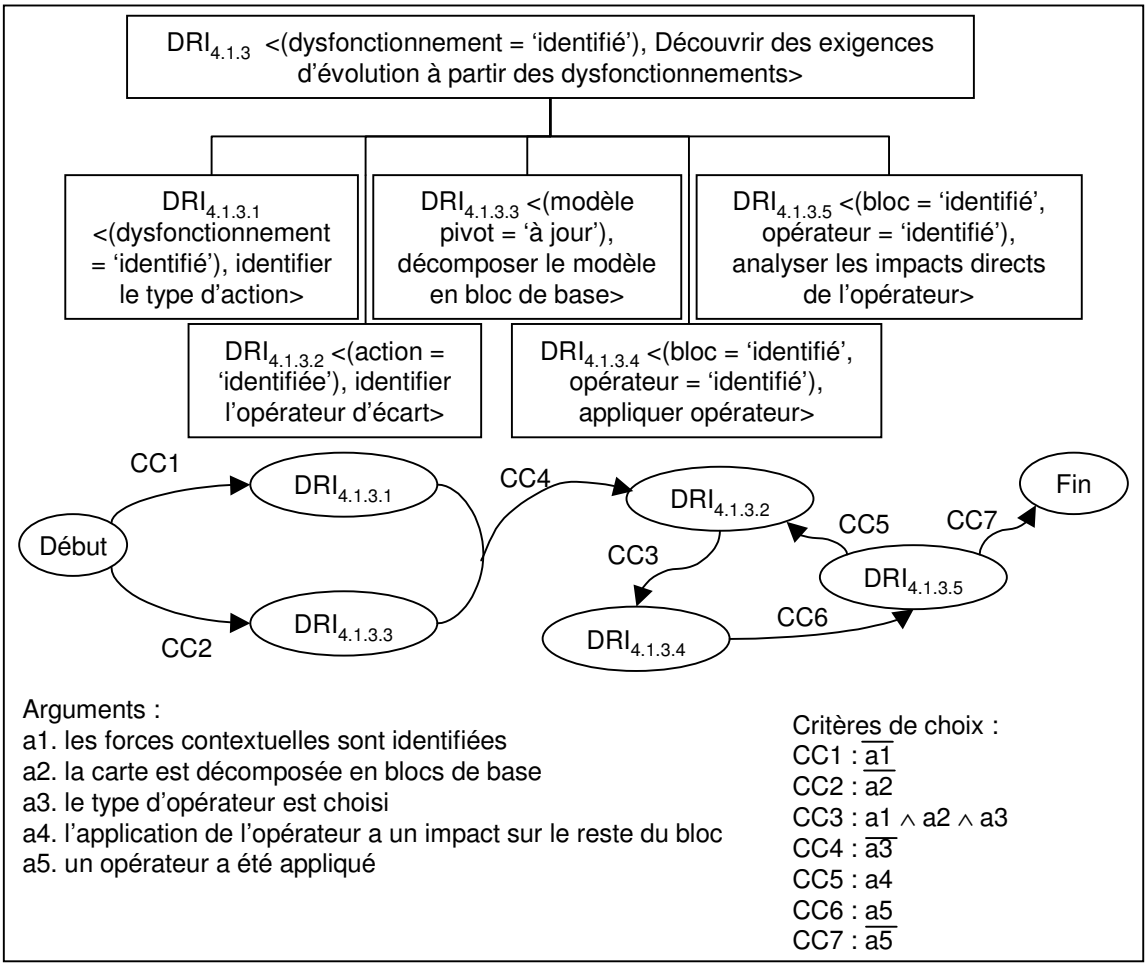


Figure 73 : Exemple de directive plan

2.2.1.3 Directive stratégique

Une directive stratégique est une directive complexe utilisant une structure de graphe pour relier ses sous-directives. Une directive stratégique permet de représenter un processus de développement multi-démarche en utilisant une carte. Les directives composant ce processus sont reliées par des liens ET/OU (topologie segment) et par des liens de précédence (topologie chemin). La carte représente alors une vue stratégique en précisant ce qui peut être réalisé (quelle intention) suivant quelle stratégie. La Figure 74 présente un exemple de directive stratégique permettant de construire un ensemble de cartes modélisant les processus d'entreprise ou le système.

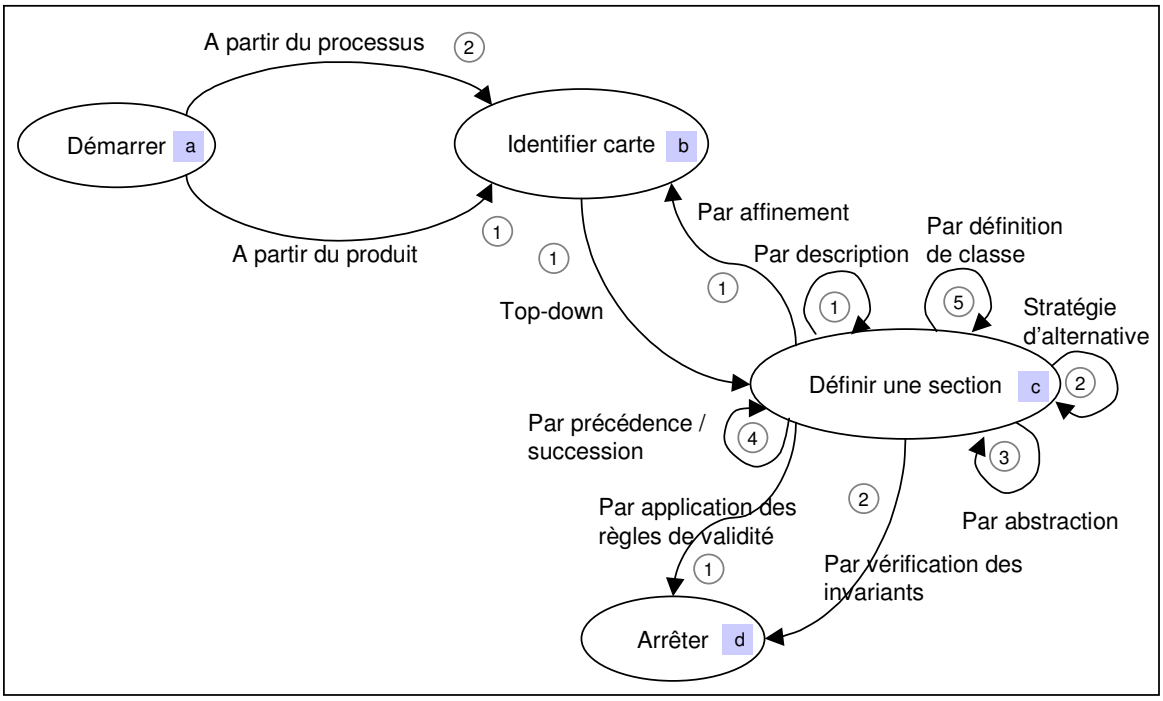


Figure 74 : Exemple de directive stratégique

2.2.2 Typologie des directives suivant le but

On distingue, suivant le but, deux types de directives :

1. celles qui aident à la réalisation des intentions. On parle alors de Directive de Réalisation d'Intention (DRI).
2. celles qui permettent de progresser dans la carte. Il peut s'agir de Directive de Sélection de Stratégie (DSS) ou de Directive de Sélection d'Intention (DSI).

Les Directives de Réalisation d'Intention (DRI) permettent d'expliquer comment réaliser l'intention sélectionnée. Elles précisent le mécanisme d'exécution de la tâche opérationnalisant cette intention. Les Directives de Sélection de Stratégie (DSS) aident à choisir une stratégie parmi un ensemble de stratégies données. Les Directives de Sélection d'Intention (DSI) permettent de découvrir toutes les intentions succédant une intention donnée.

Les sous-sections suivantes présentent successivement les DRI, les DSS et les DSI.

2.2.2.1 Directive de réalisation d'intention

Une Directive de Réalisation d'Intention (DRI) peut être simple, tactique ou stratégique. Elle aide à la réalisation d'une intention selon une stratégie donnée. Cette réalisation d'intention aboutit à la transformation du produit en cours de développement. Dans la carte, il existe une DRI pour chaque section $<I_i, I_j, S_{ij}>$. Elle aide l'ingénieur d'alignement à atteindre l'intention cible I_j en suivant la stratégie S_{ij}.

La signature d'une DRI associée à une section $<I_i, I_j, S_{ij}>$ est un couple <(situation), intention> construit comme suit :

- La situation comporte la partie de produit obtenue en réalisant l'intention I_i et dont l'état peut être précisé par une condition d'occurrence ;
- L'intention est exprimée sous la forme ***I_j avec (ou par) S_{ij}***.

Prenons comme exemple la DRI associée à la section <*Construire le modèle pivot, Découvrir les exigences d'évolution, Par analyse des dysfonctionnements*>. La signature de cette DRI est la suivante : <(*modèle pivot = 'à jour'*), *Eliciter des exigences d'évolution Par analyse des dysfonctionnements*>.

La Figure 75 présente cette DRI. Elle se décompose en trois parties :

- En haut à gauche, dans l'encadré, est représentée la partie de la carte ou la partie de la hiérarchie de directives que la directive de réalisation d'intention permet de décrire. Cette partie permet de situer la directive dans son contexte.
- La deuxième partie présente le corps de la directive. Ici il s'agit d'une hiérarchie de directives (c'est-à-dire une directive tactique composée de directives choix et directives plan). La directive <(*modèle pivot = 'à jour'*), *Eliciter des exigences d'évolution Par analyse des dysfonctionnements*> propose deux alternatives qui se décomposent chacune en trois sous-directives.
- La troisième partie définit les arguments de choix qui aident l'ingénieur d'alignement à choisir parmi les différentes alternatives qui lui sont proposées.

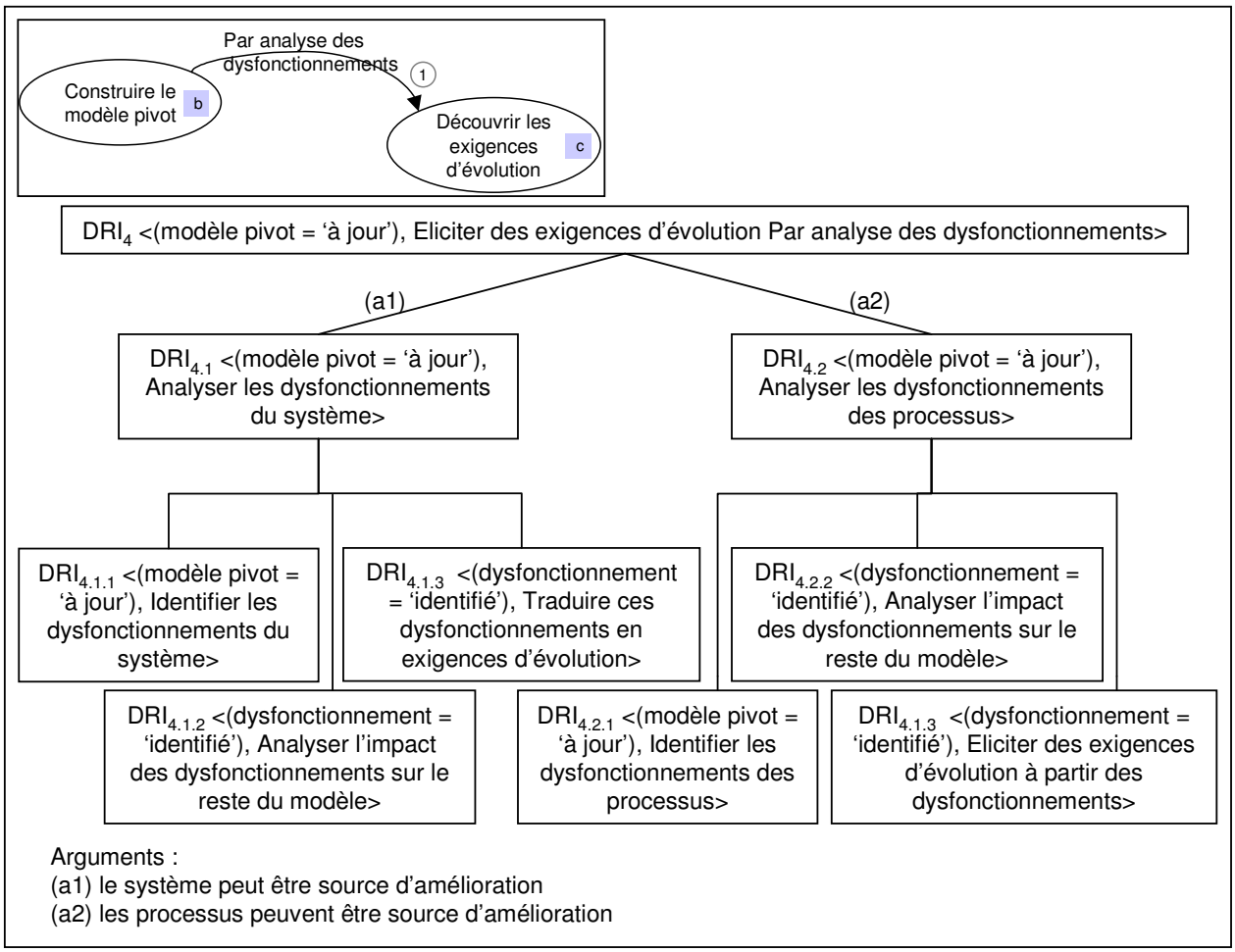


Figure 75 : Exemple de DRI

2.2.2.2 Directive de sélection de stratégie

Une Directive de Sélection de Stratégie (DSS) détermine quelles sont les stratégies connectant deux intentions et aide à choisir l'une d'elles. Elle est appliquée lorsque l'intention source et l'intention cible sont déterminées et qu'il existe plusieurs stratégies possibles pour satisfaire l'intention cible à partir de l'intention source. Le rôle de la DSS est de guider la sélection de la stratégie la plus appropriée à la situation donnée.

Pour un couple d'intentions connectées par plusieurs stratégies de même direction, il existe une DSS. Comme toute directive, la DSS est définie par un couple <(situation), intention>

La signature d'une DSS associée à un couple d'intentions $<I_i, I_j>$ est exprimée de la façon suivante :

- La situation comporte la partie de produit obtenue en réalisant l'intention I_i ; une condition d'occurrence peut préciser l'état de cette partie de produit ;
- L'intention est exprimée sous la forme : ***Progresser vers I_j***.

Le verbe *Progresser* est toujours utilisé pour exprimer les intentions des DSS. Le mot *vers* précise l'intention cible de la progression.

Une DSS est une directive tactique. Elle est composée de directives exécutables, une par stratégie à sélectionner. Chacune de ces directives contient une action de sélection de la DRI associée à la section. Dans notre exemple, la directive de sélection de stratégie illustrée à la Figure 76 est une directive de type choix. Elle aide l'ingénieur d'alignement à choisir parmi les trois stratégies proposées pour progresser dans la carte.

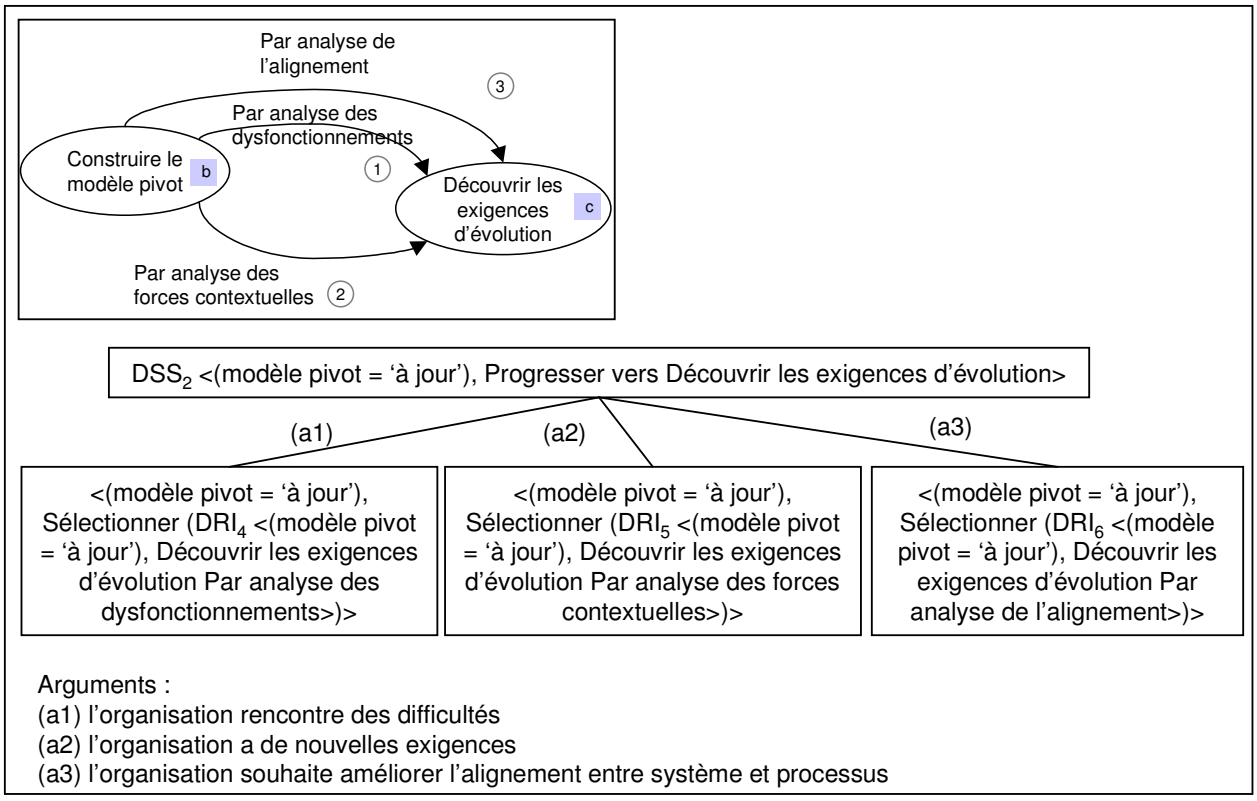


Figure 76 : Exemple de DSS

De la même façon que pour les DRI, la description des DSS se décompose en plusieurs parties (Figure 76) :

- L'encadré en haut à gauche correspond à la partie de carte à laquelle nous nous intéressons,
- la partie intermédiaire décrit la directive elle-même,
- la partie inférieure permet de spécifier les arguments aidant l'ingénieur d'alignement à faire son choix parmi les différentes stratégies proposées.

La structure des directives de sélection de stratégie est toujours la même. Elles se présentent sous la forme de directives tactiques, plus précisément de directives choix proposant différentes alternatives, ici trois. Chacune des ces alternatives est une directive exécutable sous la forme d'action de délégation permettant de sélectionner la directive de réalisation d'intention associée.

Les trois sections <*Construire le modèle pivot, Découvrir les exigences d'évolution, Par analyse des dysfonctionnements*>, <*Construire le modèle pivot, Découvrir les exigences d'évolution, Par analyse des forces contextuelles*> et <*Construire le modèle pivot, Découvrir les exigences d'évolution, Par analyse de l'alignement*> possèdent la même intention source *Construire le modèle pivot* et la même intention cible *Découvrir les exigences d'évolution*. Afin d'aider l'ingénieur d'alignement à progresser vers *Découvrir les exigences d'évolution*, après avoir *Construit le modèle pivot*, une DSS est associée à ce couple d'intentions. La signature de cette DSS est la suivante : <*(modèle pivot = 'à jour'), Progresser vers Découvrir les exigences d'évolution*>.

2.2.3 Directive de sélection d'intention

Une Directive de Sélection d'Intention (DSI) détermine quelles sont les intentions qui peuvent succéder à une intention donnée et aide à choisir l'une d'elles. Une DSI est appliquée lorsqu'une intention vient d'être réalisée et que l'utilisateur doit déterminer quelle sera la prochaine intention à réaliser. Le rôle de la DSI est de guider la sélection de l'intention suivante et de fournir l'ensemble des DRI et DSS correspondantes.

La signature d'une DSI associée à une intention I_i de la carte est exprimée de la façon suivante :

- La situation comporte la partie de produit obtenue en réalisant l'intention I_i ; elle peut éventuellement être précisée par une condition d'occurrence ;
- L'intention est exprimée sous la forme : ***Progresser de (ou depuis)*** $\boldsymbol{I_i}$

L'utilisation du verbe Progresser dans l'intention de la signature de la DSI exprime le fait que la directive aide l'ingénieur d'alignement à progresser dans la carte. Le paramètre accompagné de la préposition *de* précise l'intention qui est la source de progression.

Le corps d'une DSI définit quelles sont les intentions cibles de la progression et aide à choisir l'une d'entre elles. Une DSI est une directive tactique. Comme dans le cas des DSS, une DSI est composée d'un ensemble de directives exécutables comportant des actions de sélection de directives de type DSS ou DRI. Si la progression vers une intention cible est possible par plusieurs chemins, c'est-à-dire s'il y a plusieurs stratégies entre l'intention source et

l'intention cible, l'action sélectionne la DSS qui correspond à cette situation. S'il n'existe qu'une seule section entre ces deux intentions, l'action sélectionne directement la DRI correspondante.

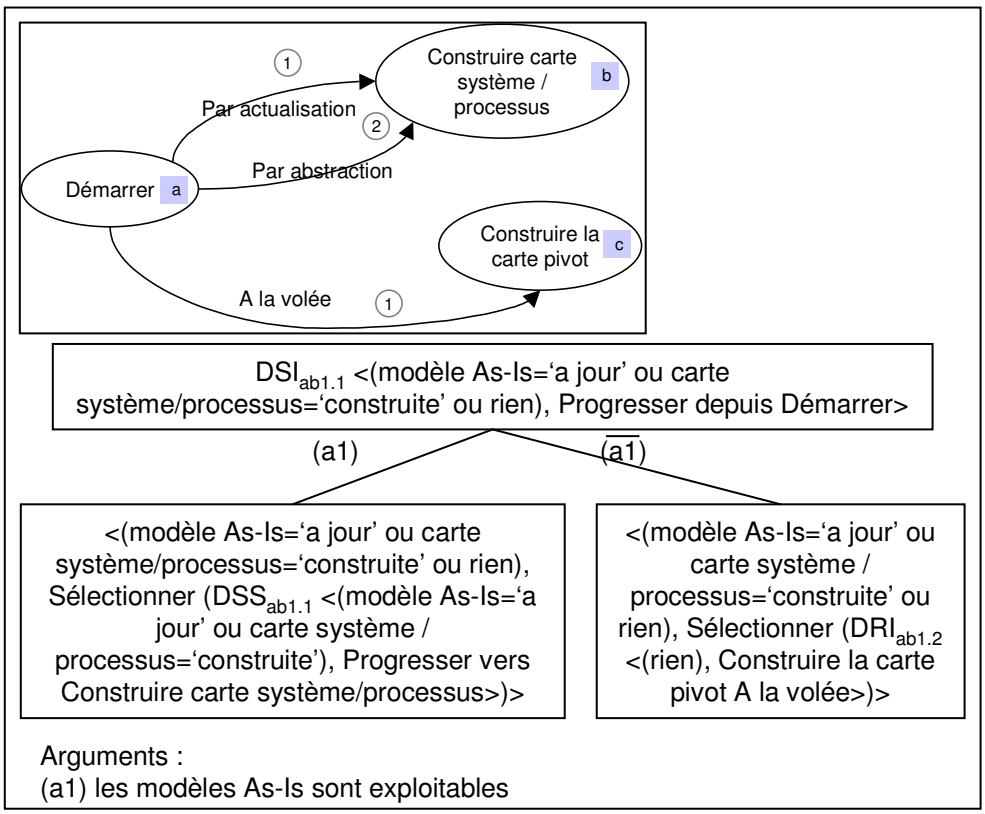


Figure 77 : Exemple de DSI

De façon analogue aux DRI et aux DSS, la description d'une DSI se décompose en trois parties. La Figure 77 montre qu'à partir de l'intention *Démarrer*, l'ingénieur d'alignement peut choisir de (i) construire une carte système ou processus ou (ii) construire la carte pivot. Afin d'aider l'ingénieur dans son choix, un argument en faveur de chacune des deux alternatives est proposé. Si l'ingénieur choisit de construire une carte système ou processus, la directive exécutable associée permet de sélectionner la DSS$_{ab1.1}$. Si l'ingénieur choisit de construire la carte pivot, il devra exécuter la directive *<(rien), Sélectionner (DRI$_{ab1.2}$ <(rien), Construire la carte pivot, A la volée>)>*.

3. Carte de la méthode ACEM

La carte présentée à la Figure 78 représente le processus de la méthode ACEM dans son ensemble. Nous la désignons carte ACEM dans la suite.

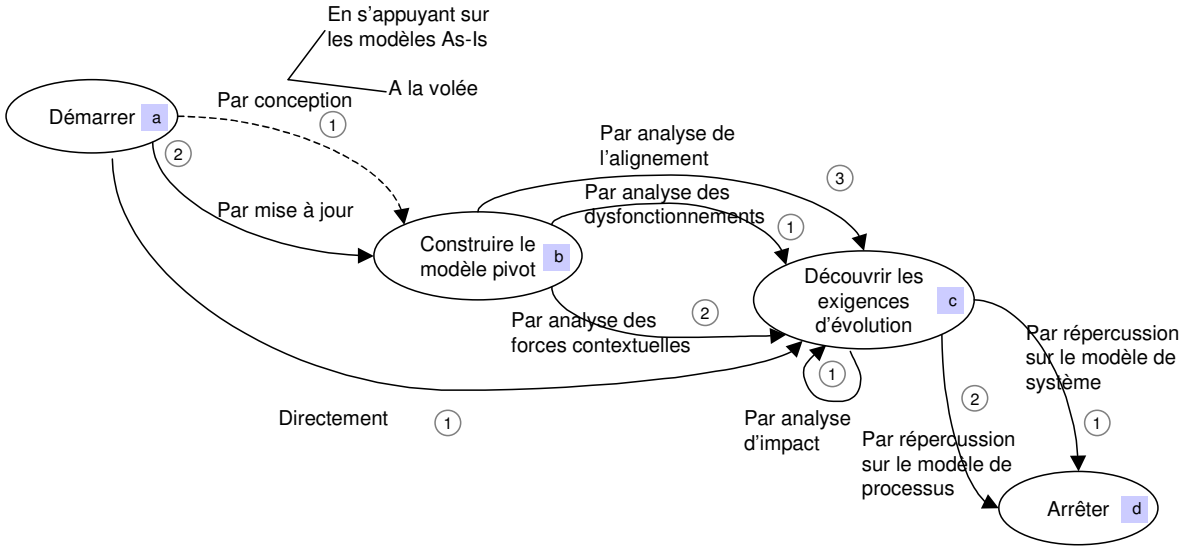


Figure 78: Carte de haut niveau représentant le processus de la méthode ACEM

Outre les intentions *Démarrer* et *Arrêter* qui existent dans toute carte, la carte ACEM possède deux autres intentions *Construire le modèle pivot* et *Découvrir les exigences d'évolution*.

- L'intention *Construire le modèle pivot* concerne la construction du modèle pivot représentant conjointement le système et les processus.
- L'intention *Découvrir les exigences d'évolution* correspond à la spécification d'exigences d'évolution sous forme d'écarts permettant de faire évoluer le système ou les processus ou bien de corriger leur alignement.
- L'intention *Arrêter* recouvre les composantes du processus correspondant à la propagation des écarts sur les modèles de système et de processus.

Les 9 sections qui composent la carte proposent des stratégies pour la réalisation de ces trois intentions et définissent des flux dans leur réalisation. Ainsi, la construction du modèle pivot est une étape préalable à la découverte des exigences d'évolution. En effet, d'une part la réalisation de la section <*Démarrer*, *Découvrir les exigences d'évolution*, *Directement*> suppose qu'il existe un modèle pivot à jour et propose de découvrir des exigences d'évolution sous forme d'écarts pour faire évoluer ce modèle. D'autre part, les autres sections ayant pour cible *Découvrir les exigences d'évolution* ont pour source *Construire le modèle pivot* ou *Découvrir les exigences d'évolution*.

La construction du modèle pivot peut se faire de deux façons différentes :

- *Par conception*, c'est-à-dire en concevant le modèle pivot si celui-ci n'existe pas déjà. Cette stratégie est un paquet proposant deux alternatives : *en s'appuyant sur les modèles As-Is* et *A la volée*. Dans le premier cas, il s'agit d'utiliser les modèles SFM et BM pour découvrir les buts du système et ceux des processus, les façons de les réaliser ainsi que les objets manipulés ; dans le second cas, le modèle pivot n'est pas construit à partir des modèles existants, mais à partir d'interviews avec les parties prenantes…
- *par mise à jour* du modèle pivot existant ; ce modèle ayant été construit lors d'une précédente évolution.

Différentes alternatives sont proposées pour *Découvrir des exigences d'évolution* :

- en analysant l'organisation dans son contexte. Trois approches différentes sont proposées suivant qu'on cherche à découvrir des exigences d'évolution *Par analyse des dysfonctionnements*, *Par analyse des forces contextuelles* ou *Par analyse de l'alignement*.
- soit en analysant l'impact que peuvent avoir des exigences déjà découvertes sur le reste du modèle pivot. En effet, certaines exigences d'évolution peuvent nécessiter ou entraîner la découverte d'autres exigences.

Une alternative composée de deux stratégies permet d'arrêter le processus *Par répercussion sur le modèle de processus* et *Par répercussion sur le modèle de système* qui proposent de traduire les écarts découverts sur le modèle pivot respectivement sur le modèle de processus et sur le modèle de système. Ces deux stratégies permettent de construire le modèle To-Be BM et le modèle To-Be SFM représentant respectivement les processus et le système après évolution.

Les neuf sections de la carte sont numérotées en adoptant la numérotation décrite au chapitre 6 : les intentions portent des lettres, les stratégies des numéros. Les sections sont donc codées ab1, bc2, cc1 etc., ou de façon absolue, C.ab1, C.bc2, C.cc1 etc.

Les sections de la carte ACEM sont listées dans le Tableau 25 ainsi que les DSI et DSS (en grisé dans le tableau) permettant de choisir ces sections (la colonne de droite renvoie à la partie du chapitre dans laquelle la section de la carte est décrite) :

Sections de la carte ACEM	Type	Description
Progresser depuis Démarrer	DSI	4
Progresser vers Construire le modèle pivot	DSS	5.1
C.ab1 : <Démarrer, Construire le modèle pivot, Par conception>	DRI	5.2
C.ab2 : <Démarrer, Construire le modèle pivot, Par mise à jour>	DRI	5.3
Progresser vers Découvrir les exigences d'évolution	DSS	6.1
C.bc1 : <Construire le modèle pivot, Découvrir les exigences d'évolution, Par analyse des dysfonctionnements>	DRI	6.2
C.bc2 : <Construire le modèle pivot, Découvrir les exigences d'évolution, Par analyse des forces contextuelles>	DRI	6.3
C.bc3 : <Construire le modèle pivot, Découvrir les exigences d'évolution, Par analyse de l'alignement >	DRI	6.4
C.ac1 : <Démarrer, Découvrir les exigences d'évolution, Directement>	DRI	6.5
Progresser depuis Découvrir les exigences d'évolution	DSI	6.6
C.cc1 : <Découvrir les exigences d'évolution, Découvrir les exigences d'évolution, Par analyse d'impact>	DRI	6.7
Progresser vers Arrêter	DSS	7.1
C.cd1 : <Découvrir les exigences d'évolution, Arrêter, Par répercussion sur le modèle de système>	DRI	7.2
C.cd2 : <Découvrir les exigences d'évolution, Arrêter, Par répercussion sur le modèle des processus>	DRI	7.2

Tableau 25 : Les neuf sections de la carte ACEM

L'exécution de la démarche ACEM suit la structure de la carte qui la représente. Le parcours de la carte se fait de façon dynamique et contextuelle. A tout moment de l'exécution de la carte, des directives aident l'ingénieur d'alignement à naviguer dans la carte pour décider quelle intention réaliser, quelle stratégie utiliser, mais aussi à exécuter les intentions sélectionnées suivant la stratégie choisie. Le choix de l'une des alternatives dépend de l'état du produit ainsi que de l'objectif que l'ingénieur d'alignement souhaite atteindre. Il est par

exemple, possible de découvrir de nouvelles exigences d'évolution à partir (i) des dysfonctionnements de l'existant, des forces contextuelles et/ou de la relation d'alignement ou (ii) des exigences déjà spécifiées. La carte contient ainsi plusieurs chemins entre *Démarrer* et *Arrêter*.

La présentation des directives est faite dans l'ordre des sections ci-dessus. Pour chaque section, la présentation (1) est faite en profondeur jusqu'aux directives exécutables ou informelles et (2) suit la navigation dans la carte et introduit les DSI et les DSS avant les DRI qu'elles appellent.

4. Progresser depuis Démarrer

Pour démarrer le processus de la méthode ACEM, deux possibilités s'offrent à l'ingénieur d'alignement : *Construire le modèle pivot* ou *Découvrir les exigences d'évolution*. Cette dernière doit être choisie s'il existe un modèle pivot à jour représentant le système et les processus. Dans le cas contraire, la première possibilité permet de *Construire le modèle pivot* en le mettant à jour ou en le concevant de toute pièce. La Figure 79 présente la structure de la DSI permettant de progresser depuis *Démarrer*.

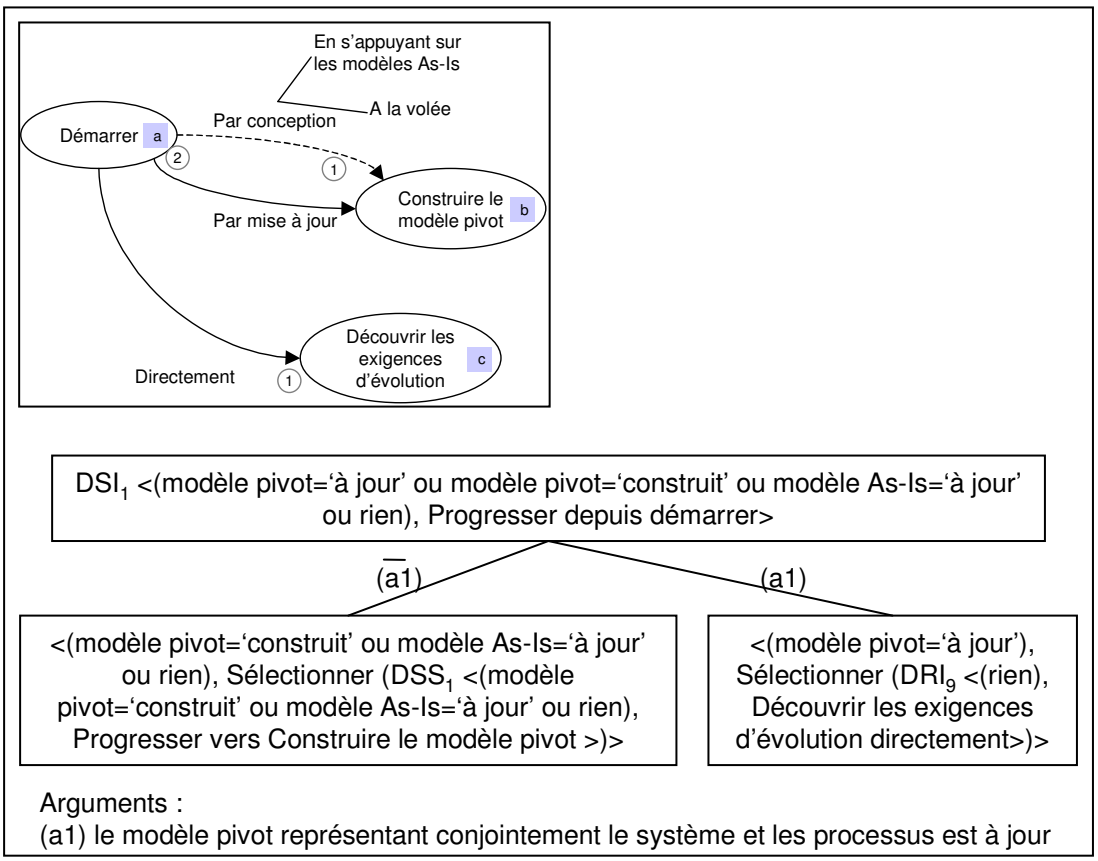


Figure 79 : Directive de sélection d'intention permettant de Progresser depuis Démarrer

Cette directive met en avant le caractère indispensable du modèle pivot dans la réalisation de la méthode ACEM. Il constitue un préalable à la découverte des exigences d'évolution. En effet, parmi les trois sections ayant pour source l'intention *Démarrer*, deux proposent de construire un modèle pivot et la troisième suppose que celui-ci existe. C'est ce modèle qui

évolue pour répondre aux nouvelles exigences. Il est donc indispensable de commencer par construire ce modèle s'il n'existe pas.

La construction du modèle pivot peut s'effectuer de différentes façons qui sont décrites dans la section suivante. La découverte des exigences d'évolution directement est, quant à elle, décrite à la section 6.5 (p. 258).

5. Construction du modèle pivot

Il existe différentes façons de construire le modèle pivot suivant que celui-ci a déjà été construit lors d'une précédente évolution ou qu'il est nécessaire d'en concevoir un de toute pièce.

5.1. Progresser vers Construire le modèle pivot

Cette section correspond à la réalisation de la directive *<(modèle pivot='construit' ou modèle As-Is='à jour' ou rien), Progresser vers Construire le modèle pivot >*.

La construction du modèle pivot peut se faire de deux façons différentes. Afin d'aider les ingénieurs d'alignement à choisir la stratégie la plus appropriée au contexte du projet, nous proposons la directive de sélection de stratégie présentée à la Figure 80.

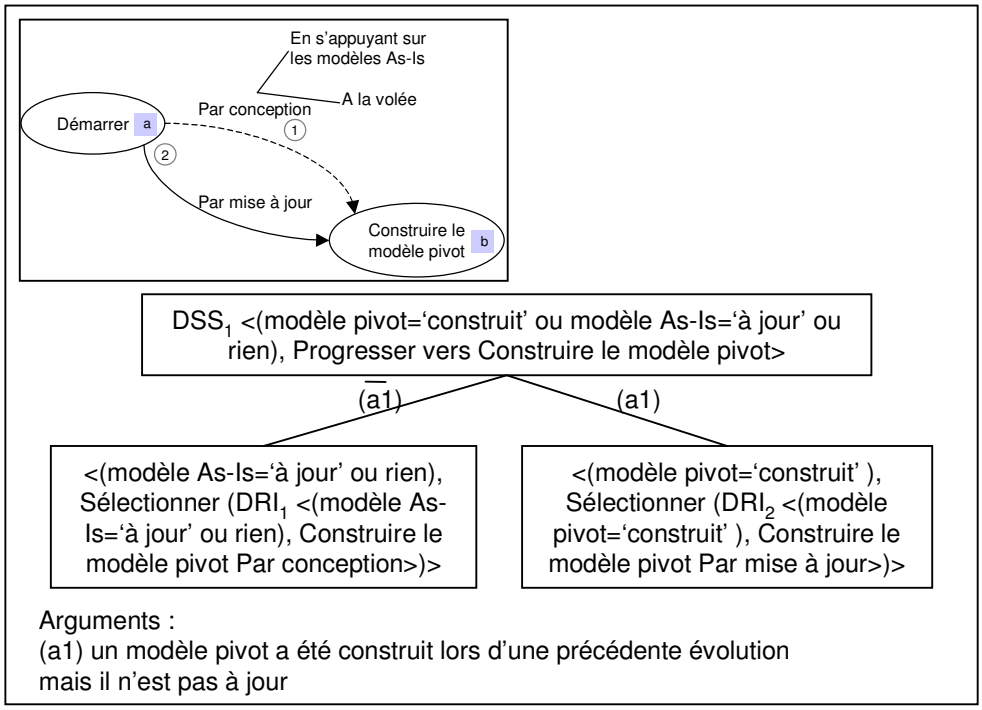


Figure 80 : Directive de sélection de stratégie permettant de progresser vers Construire le modèle pivot

Les deux sous-directives guident la sélection des directives de réalisation d'intention DRI_1, DRI_2. La DRI_2 est sélectionnée si (i) un modèle pivot a déjà été construit par exemple lors d'une précédente évolution et (ii) ce modèle doit être mis à jour. La DRI_1 doit être choisie si le modèle pivot n'a pas encore été construit.

Le reste de la section 5 a pour but de décrire en détail chacune de ces deux directives ainsi que leurs affinements respectifs.

5.2. Construire le modèle pivot par conception

Cette section correspond à la description de la directive <(*modèle As-Is='à jour' ou rien*), *Construire le modèle pivot Par conception*>.

Deux alternatives mutuellement exclusives (représentées au sein d'un paquet) sont proposées pour construire un modèle pivot par conception : *En s'appuyant sur les modèles As-Is* quand ceux-ci sont à jour, c'est-à-dire quand ils sont fidèles à la réalité ou *A la volée* si ces modèles sont inexploitables. L'éventuelle mise à jour des modèles As-Is SFM et BM suit des méthodes de rétro-ingénierie qui ne sont pas présentées dans cette thèse.

La directive permettant de construire le modèle pivot par conception est stratégique. Elle se présente sous la forme d'une carte affinant la section <*Démarrer, Construire le modèle pivot, Par conception*> de la carte ACEM dans l'encadré en haut à gauche de la Figure 81.

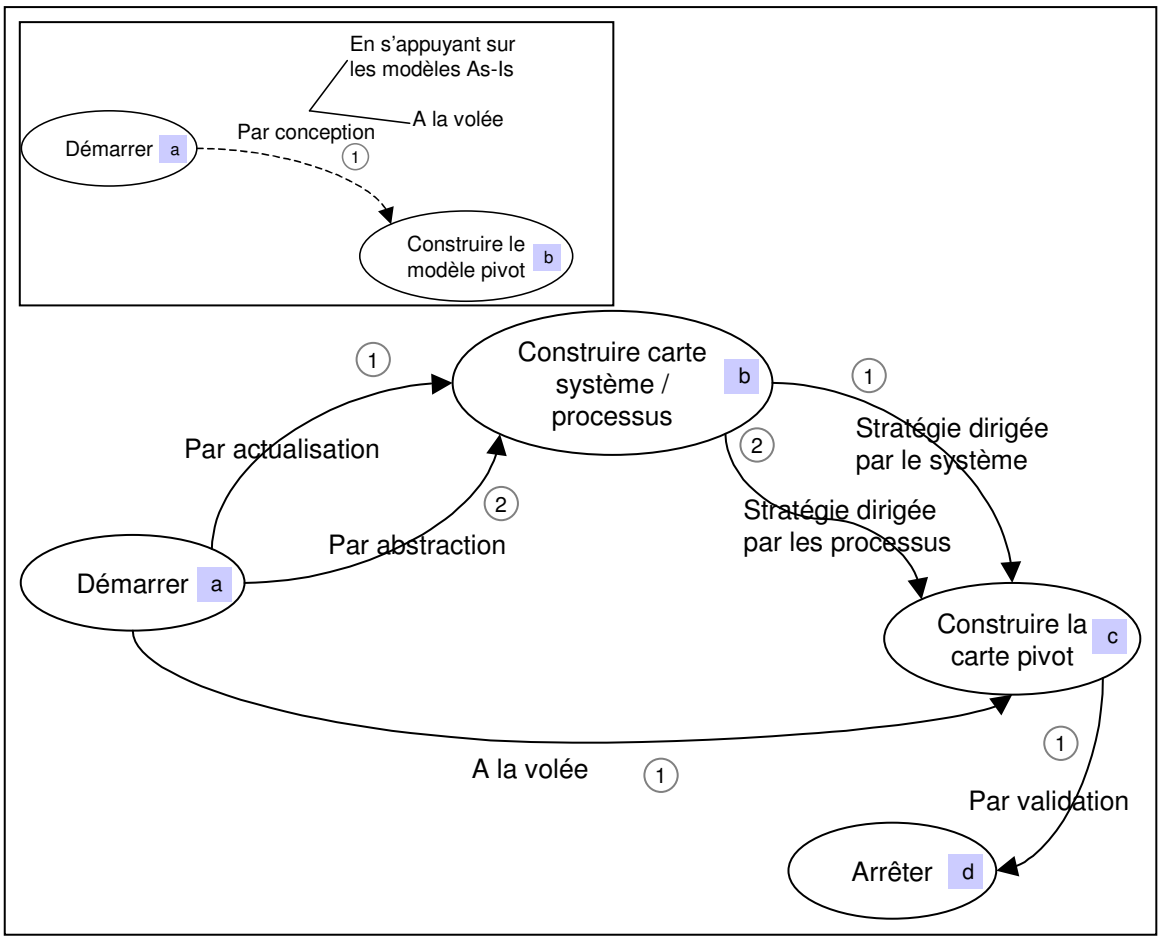


Figure 81 : Directive stratégique permettant de Construire le modèle pivot en s'appuyant sur les modèles As-Is

En plus des intentions *Démarrer* et *Arrêter*, la carte C_{ab1} présentée à la Figure 81 compte deux intentions :

- *Construire carte système / processus* qui permet d'élaborer une carte système et une carte processus. Pour des raisons de lisibilité de la Figure 81 et dans la mesure où la construction de ces deux cartes se fait de façon analogue, nous n'avons utilisé qu'une seule intention *Construire carte système / processus*. Cette intention doit donc être réalisée deux fois, une fois pour construire la carte système (ce qui permet d'amener le produit 'carte système' à l'état 'à jour') et une fois pour construire la carte processus (amenant le produit 'carte processus' à l'état 'à jour').

- *Construire la carte pivot* qui correspond à la construction d'une carte représentant conjointement le système et les processus. Après réalisation de cette intention, l'état de la carte pivot est 'construite'.

Les cartes processus et système peuvent être construites *Par actualisation* si elles ont déjà été construites auparavant ou *Par abstraction* dans le cas contraire. Il est important de noter que la construction de ces deux cartes peut se faire en choisissant des alternatives différentes.

Deux alternatives sont proposées pour construire la carte pivot à partir des cartes du système et des processus : (i) *stratégie dirigée par le système* ou (ii) *stratégie dirigée par les processus* suivant que les cartes système ou les cartes processus servent de référence.

Le processus se termine quand la carte pivot est *validée*.

Cette carte comprend les sections suivantes :

Sections de la carte affinant C.ab1	Description
$C.C_{ab1}$.ab1 : <Démarrer, Construire carte du système / processus, Par mise à jour>	5.2.4
$C.C_{ab1}$.ab2 : <Démarrer, Construire carte du système / processus, Par abstraction>	5.2.3
$C.C_{ab1}$.ac1 : <Démarrer, Construire la carte pivot, A la volée>	5.2.5
$C.C_{ab1}$.bc1 : <Construire carte système / processus, Construire la carte pivot, stratégie dirigée par le système>	5.2.7
$C.C_{ab1}$.bc2 : <Construire carte système / processus, Construire la carte pivot, stratégie dirigée par les processus>	5.2.7
$C.C_{ab1}$.de1 : < Construire la carte pivot, Arrêter, Par validation>	5.2.8

Tableau 26 : Sections de la carte permettant de construire le modèle pivot en s'appuyant sur les modèles As-Is

Chacune des différentes sections correspond à une directive et est détaillée aux sous-sections suivantes.

5.2.1 Progresser depuis Démarrer

La directive <(*modèle pivot='à jour' ou modèle pivot='construit' ou modèle As-Is='à jour' ou rien*), *Progresser depuis démarrer*> propose deux choix pour construire le modèle pivot :

- *Construire une carte du système* et *une carte des processus* si les modèles As-Is sont exploitables ;

- *Construire la carte pivot à la volée* dans le cas contraire. Dans ce cas, la carte pivot correspondant au modèle pivot est construite directement sans construire au préalable une carte du système et une carte du processus.

La Figure 82 présente la directive de sélection d'intention permettant de progresser depuis *Démarrer*.

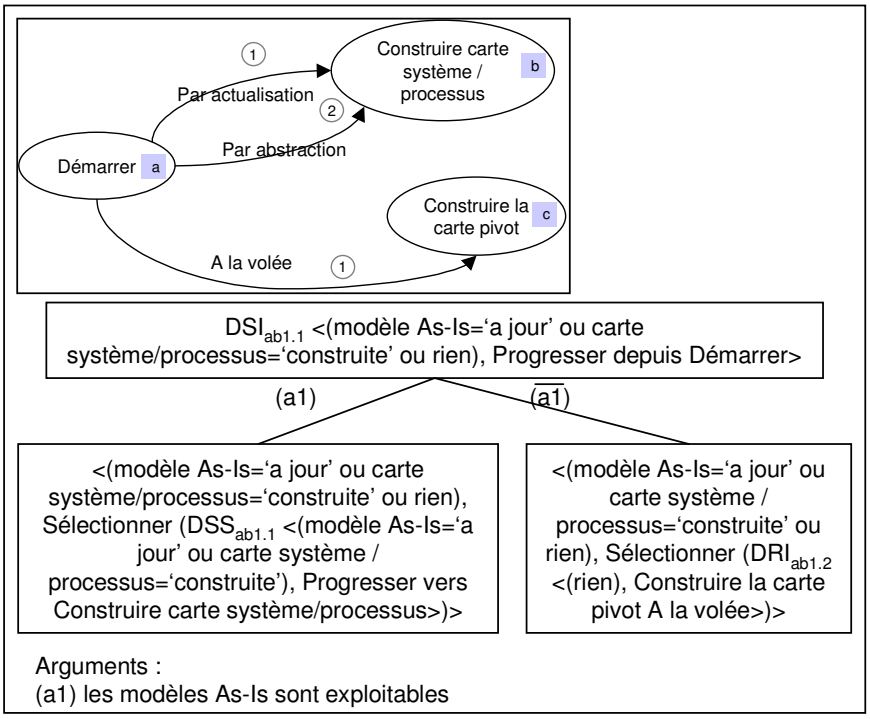


Figure 82 : Directive de sélection d'intention permettant de Progresser depuis Démarrer

Plusieurs alternatives sont proposées pour construire les cartes du système et des processus.

5.2.2 Progresser vers Construire carte système/processus

La directive <(*modèle As-Is='a jour' ou carte système / processus='construite'*), *Progresser vers Construire carte système/processus*> présente la particularité suivante : l'intention *Construire carte système/processus* doit être réalisée deux fois pour :

- *Construire une carte du système* ne représentant que les fonctionnalités du système.
- *Construire une carte des processus* ne décrivant que les intentions que les processus permettent de réaliser et les objets de gestion manipulés.

Ceci se traduit à la Figure 83 par le dédoublement de l'intention *Construire carte système/processus* en *Construire carte système* et *Construire carte processus*. Cette représentation permet de préciser que deux buts distincts doivent être atteints, mais que pour chacun de ces deux buts, les deux mêmes stratégies sont proposées. Ainsi, pour construire une carte (système ou processus), nous proposons deux alternatives :

1. mettre à jour une carte ; cette alternative requiert l'existence d'une carte préalable,
2. abstraire les objectifs de l'étude des modèles existants.

La Figure 83 présente la directive de sélection de stratégie permettant de construire une carte système ou processus. Cette directive est de type choix et est composée de deux sous-directives de délégation permettant de sélectionner les directives de réalisation d'intention associée à chacune des deux sections.

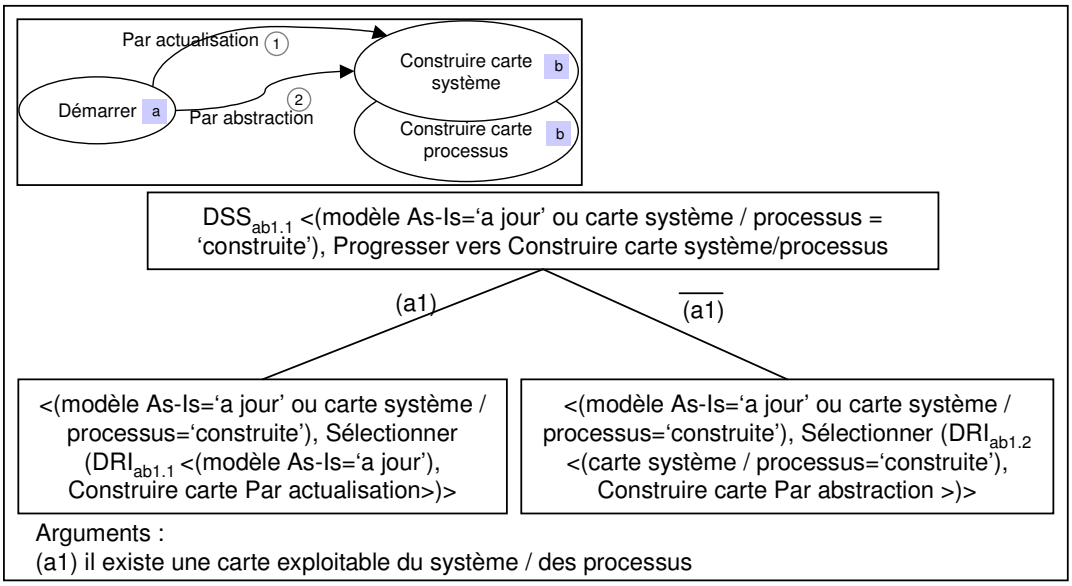


Figure 83 : Directive de sélection de stratégie permettant de progresser vers Construire carte système/processus

5.2.3 Construire carte par abstraction

La directive <(*carte système / processus='construite'*), *Construire carte Par abstraction*> se présente sous la forme d'une carte affinant la section C.C$_{ab1}$.ab3. En effet, la construction d'une carte est un processus long et complexe que nous décrivons Figure 84 par une directive stratégique.

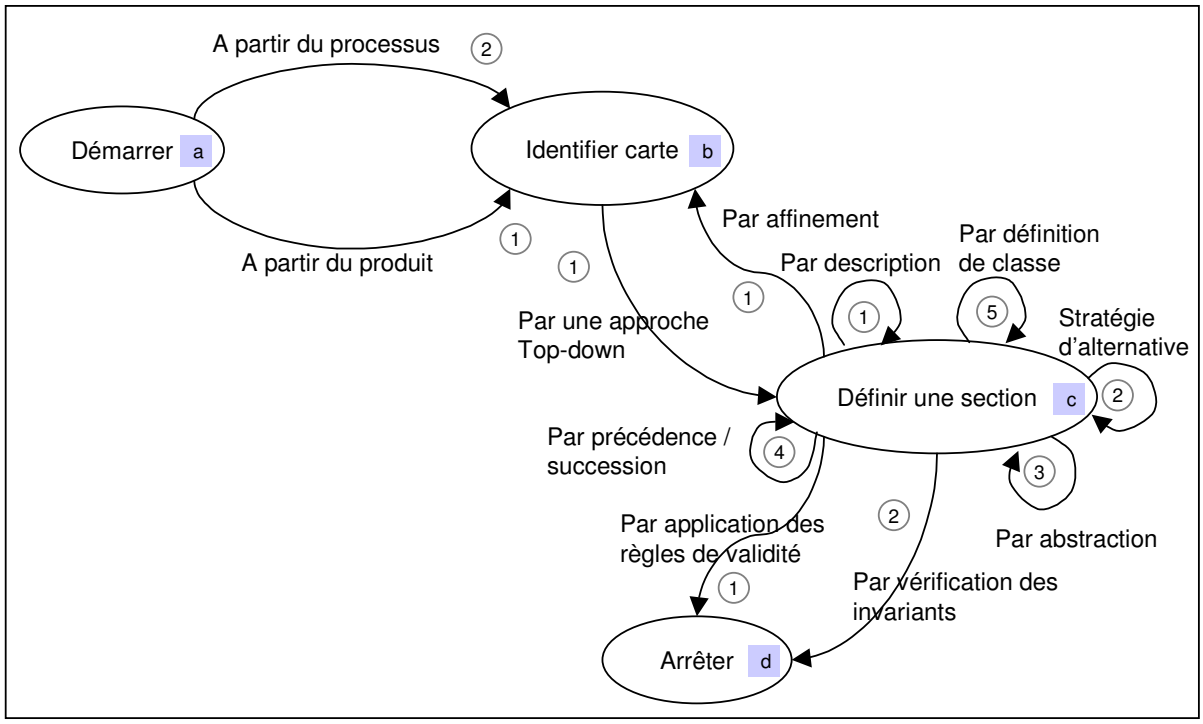


Figure 84 : Directive stratégique aidant à Construire une carte par abstraction

Cette carte compte deux intentions en plus des intentions *Démarrer* et *Arrêter* : *Identifier carte* et *Définir une section*.

- L'intention *Identifier carte* correspond à l'identification du code et de la désignation d'une carte.
- L'intention *Définir une section* permet d'identifier et de décrire une section.

Deux stratégies *A partir du produit* et *A partir des processus* permettent d'atteindre l'intention *Identifier carte*. Ces deux stratégies se différencient en fonction de la vision offerte par le modèle As-Is ; celle-ci pouvant être plutôt accès processus ou plutôt accès produit.

La définition d'une section se fait *Par une approche Top-down* à partir de la carte identifiée. Il s'agit alors de découvrir des intentions et des stratégies permettant de satisfaire la désignation de la carte.

Le guidage offert par les stratégies *Par précédence/succession* et *Stratégie d'alternative* exploite les liens entre les sections de la carte construite. Ainsi la stratégie *Par précédence/succession* cherche à décrire une section pour progresser à partir ou vers une situation issue de la réalisation d'une intention déjà décrite. La *stratégie d'alternative* aide à décrire des sections alternatives pour réaliser les intentions déjà décrites dans la carte. La stratégie *Par abstraction* permet d'abstraire plusieurs sections déjà construites pour définir une seule et même section. La stratégie *Par description* est complémentaire des trois précédentes. Elle guide la documentation d'une section en spécifiant ses propriétés, c'est-à-dire les pré-conditions, les post-conditions, les règles de gestion, éventuellement son acteur déclencheur et ses ressources. La stratégie *Par définition de classe* permet de préciser les objets manipulés par le système ou les processus.

La stratégie *Par affinement* permet d'affiner une section (c'est-à-dire de la décrire par une carte complète) en identifiant le code et la désignation de la carte affinant.

Le processus de construction de carte As-Is par abstraction s'achève en *vérifiant les invariants* et *en appliquant les règles de validité*.

Le Tableau 27 liste les sections de la carte $C.C_{ab1}.C_{ab2}$ et précise le paragraphe où la directive de réalisation d'intention correspondante est décrite.

Sections de la carte $C.C_{ab1}.C_{ab2}$	Description
<Démarrer, Identifier carte, A partir du processus>	5.2.3.3
<Démarrer, Identifier carte, A partir du produit>	5.2.3.2
<Identifier carte, Définir une section, Par une approche Top-down>	5.2.3.4
<Définir une section, Identifier carte, Par affinement>	5.2.3.6
<Définir une section, Définir une section, Par description>	5.2.3.8
<Définir une section, Définir une section, stratégie d'alternative>	5.2.3.9
<Définir une section, Définir une section, Par abstraction>	5.2.3.10
<Définir une section, Définir une section, Par précédence/succession>	5.2.3.11
<Définir une section, Définir une section, Par définition de classe>	5.2.3.12
<Définir une section, Arrêter, Par vérification des invariants>	5.2.3.15
<Définir une section, Arrêter, Par application des règles de validité>	5.2.3.14

Tableau 27 : Liste des sections de la carte $C.C_{ab1}.C_{ab2}$

Les différentes directives permettant de naviguer dans cette carte et de réaliser les différentes intentions sont décrites dans le reste de la section 5.2.3.

5.2.3.1 Progresser vers Identifier carte

La directive <(*modèle As-Is='à jour'*), *Progresser vers Identifier carte*> propose deux alternatives pour identifier une carte. Elles s'appuient sur le contenu des documentations

existantes : orientée produit ou orientée processus. La Figure 85 présente la directive de sélection de stratégie permettant de progresser vers *Identifier carte*.

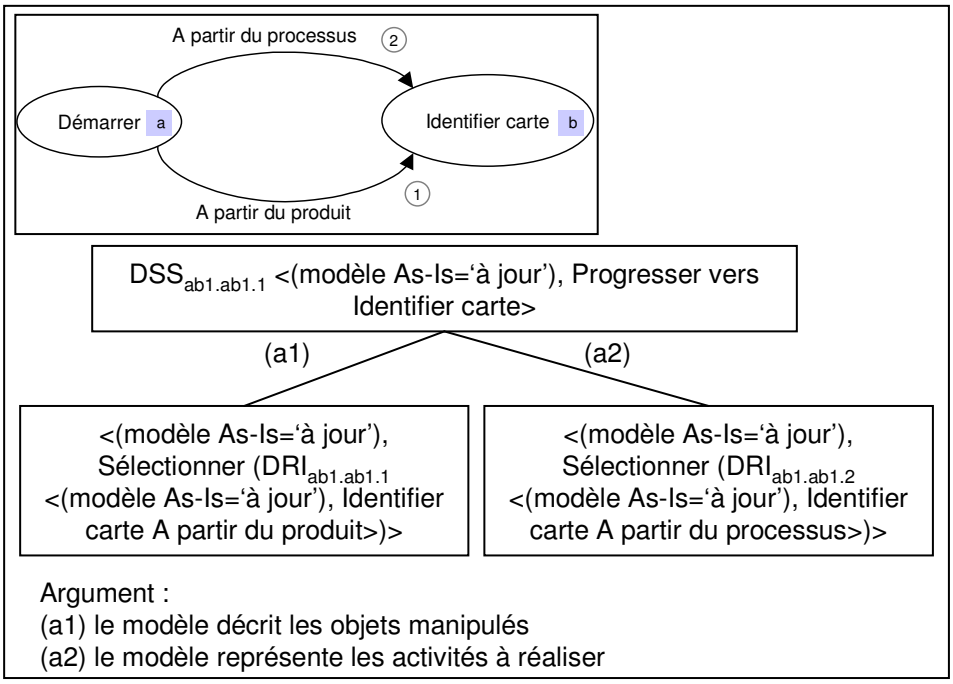


Figure 85 : Directive de sélection de stratégie permettant de Progresser vers Identifier carte

Pour construire la carte du système, l'intention *Identifier carte* est plus vraisemblablement atteinte en suivant la stratégie *A partir du produit*. En effet, un modèle de système représente le plus souvent les objets manipulés, c'est-à-dire le produit. A l'inverse, l'intention *Identifier carte* est réalisée en suivant la stratégie *A partir du processus* quand il s'agit de construire la carte des processus d'entreprise.

5.2.3.2 Identifier carte, A partir du produit

La directive <(*modèle As-Is='à jour'*), *Identifier carte A partir du produit*> est informelle. Elle s'appuie sur la structure et le contenu du modèle du système ou des processus pour identifier, à partir du produit, un objectif que le système (respectivement les processus d'entreprise) permet de satisfaire. L'identification de la carte consiste à attribuer une désignation à la carte. Cette désignation précise l'objectif que la carte doit satisfaire dans son ensemble.

5.2.3.3 Identifier carte, A partir du processus

La directive <(*modèle As-Is='à jour'*), *Identifier carte A partir du processus*> est analogue à la précédente. Cependant, elle utilise le processus pour identifier un objectif.

5.2.3.4 Définir une section Par une approche Top-Down

La directive <(*carte='identifiée'*, *Définir une section Par une approche Top down*> propose de définir une section en s'appuyant sur la structure d'une section telle qu'elle est définie dans le méta-modèle pivot. Une section est un triplet constitué d'une intention source, d'une intention cible et d'une stratégie permettant d'atteindre l'intention cible à partir de l'intention source. La directive $DRI_{ab1.ab1.3}$ est de type plan et est composée de trois sous-directives permettant chacune de construire l'un des trois éléments du triplet.

La Figure 86 présente la structure de la directive de réalisation d'intention aidant à Définir une section de façon Top-down.

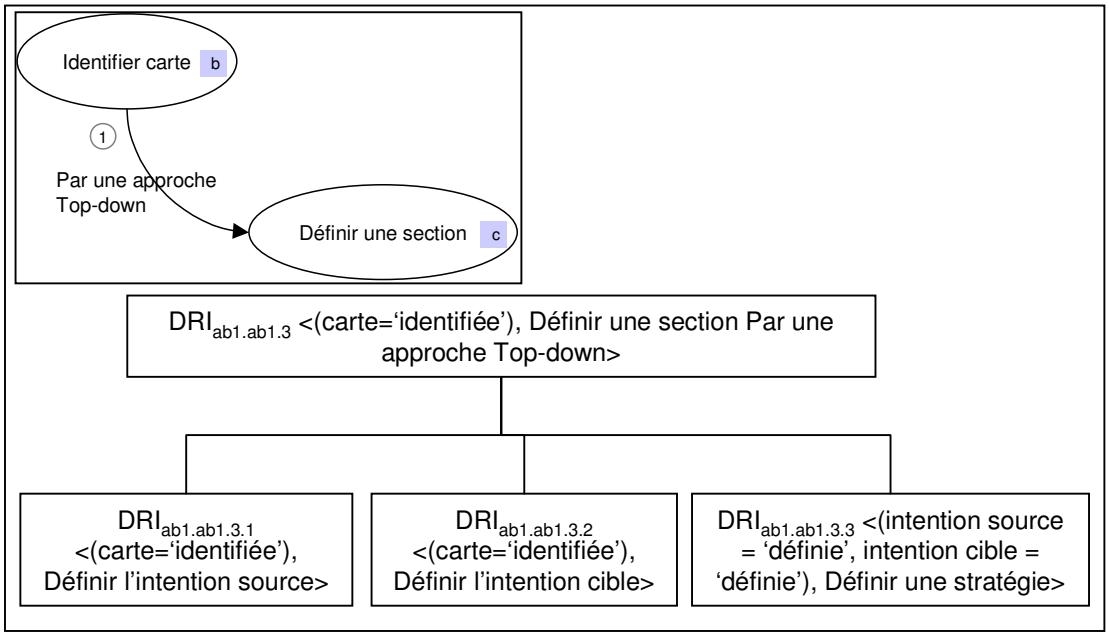


Figure 86 : Directive de réalisation d'intention aidant à Définir une section Par une approche Top-down

$DRI_{ab1.ab1.3.1}$ <(carte = 'identifiée'), Définir l'intention source>

Cette directive se décompose en deux directives correspondant à l'identification et à la description de l'intention. La directive $DRI_{ab1.ab1.3.1}$ est de type plan.

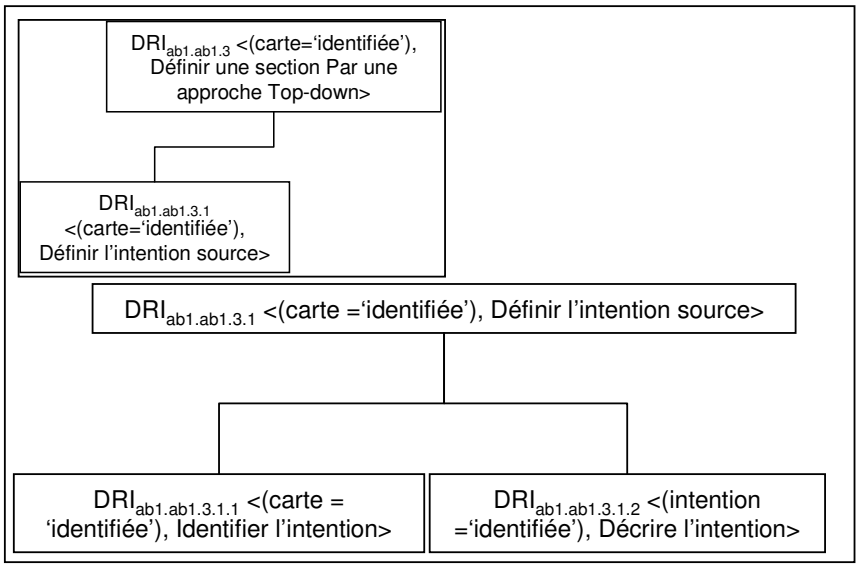


Figure 87 : Directive plan permettant de Définir l'intention source

Chacune des deux sous-directives de la Figure 87 est une directive informelle.

$DRI_{ab1.ab1.3.1.1}$ <(carte = 'identifiée'), Identifier l'intention>

Cette directive vise à identifier une intention en lui attribuant un nom. La réalisation de cette intention permet de satisfaire la désignation de la carte identifiée en cours de construction.

$DRI_{ab1.ab1.3.1.2}$ <(intention = 'identifiée'), Décrire l'intention>

Cette directive s'appuie sur la définition du concept d'intention telle qu'elle est précisée dans le méta-modèle pivot au chapitre 6. Ainsi, la description d'une intention revient à définir les états qui la constituent. En effet, le méta-modèle pivot définit une intention I comme un ensemble d'états désirables G_I.

$DRI_{ab1.ab1.3.2}$ <(carte = 'identifiée'), Définir l'intention cible>

Cette directive est similaire à la directive $DRI_{ab1.ab1.3.1}$.

$DRI_{ab1.ab1.3.3}$ <(intention source = 'définie', intention cible= 'définie'), Définir une stratégie>

Cette directive a pour but de définir une stratégie à partir d'un couple d'intentions. Comme le montre la Figure 88 , cette directive est de type plan et est composée de deux directives :

- l'identification d'une manière de réaliser l'intention cible à partir de l'intention source. L'intention source (respectivement l'intention cible) permet de préciser la situation initiale (respectivement à atteindre).
- l'identification d'une stratégie à partir du couple d'intentions et de la manière identifiée.

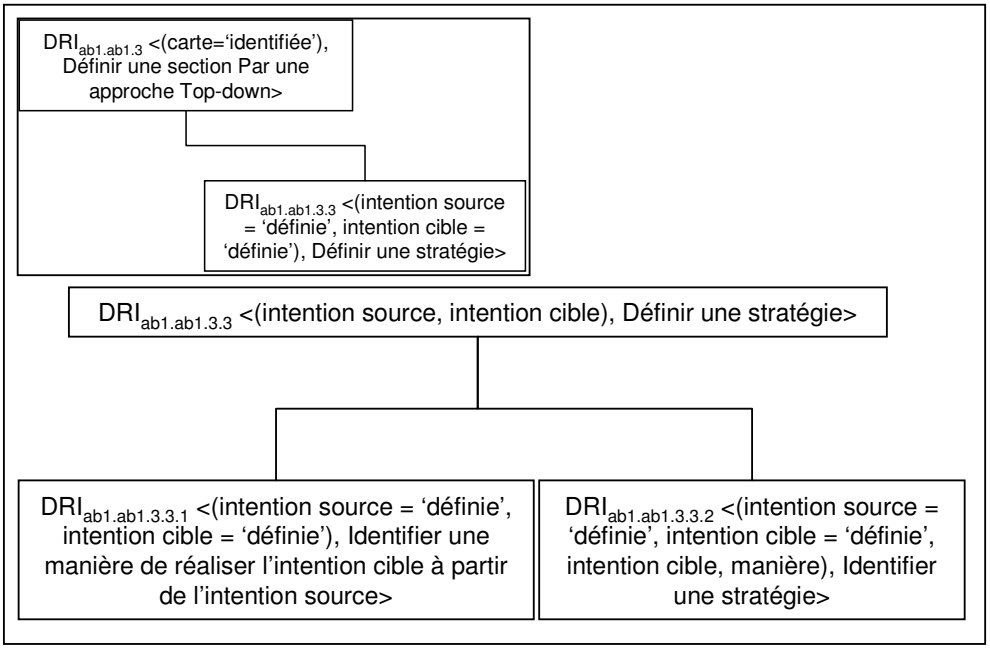


Figure 88 : Directive plan permettant de Définir une stratégie

5.2.3.5 Progresser depuis définir une section

Après avoir construit une section, l'ingénieur d'alignement peut choisir parmi plusieurs alternatives pour continuer le processus. Il peut :

1. identifier une carte par affinement. Cette directive permet d'affiner une section.
2. définir une autre section à partir de section(s) déjà construite(s).
3. arrêter le processus de construction de carte.

La directive <(*section = 'définie'*), *Progresser depuis Définir une section*> aide l'ingénieur d'alignement dans son choix. La Figure 89 présente la structure de la directive de sélection permettant de progresser depuis *Définir une section*.

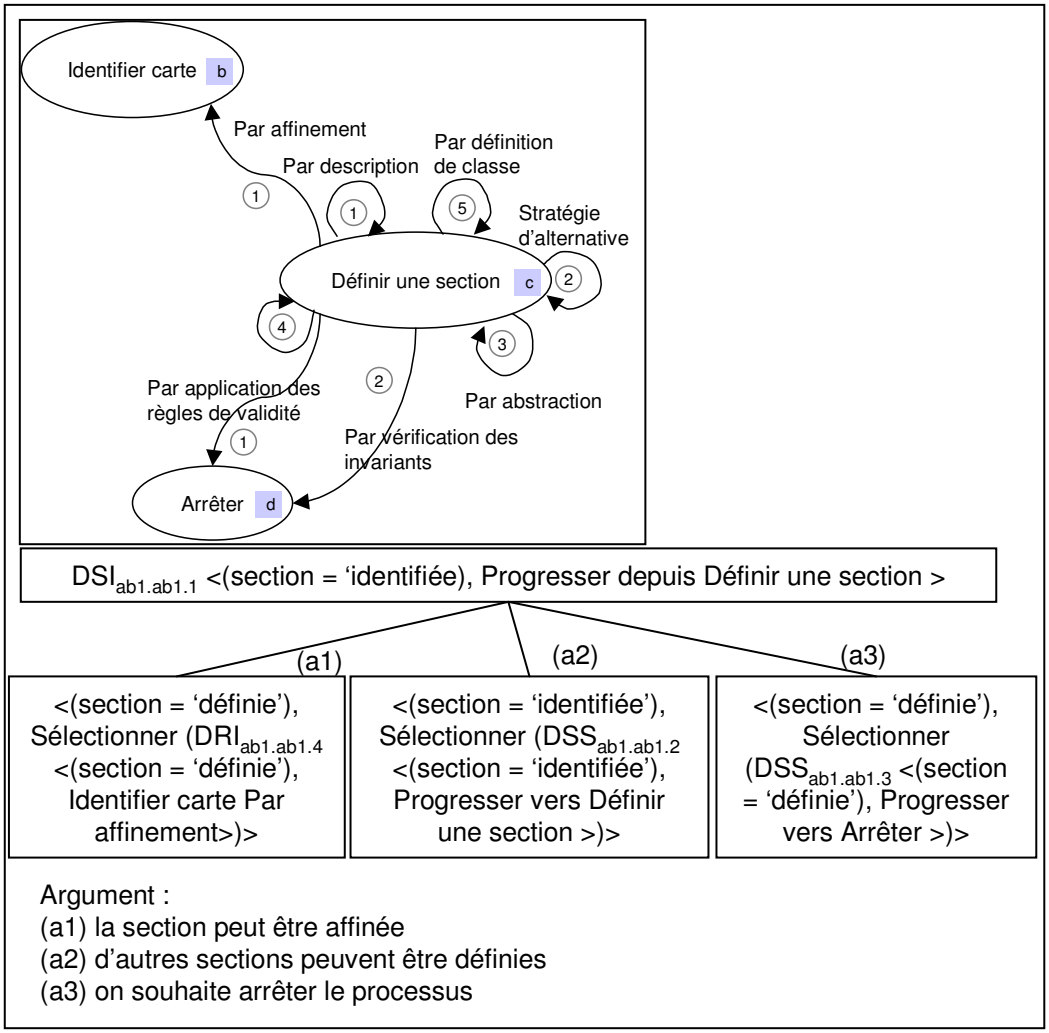


Figure 89 : Directive de sélection d'intention aidant à progresser depuis Définir une section

5.2.3.6 Identifier carte Par affinement

La directive <(*section='définie'*) *Identifier carte Par affinement*> est de type plan. Sa structure est présentée à la Figure 90. Cette DRI a pour but d'identifier dans la carte une section complexe qui nécessite d'être affinée par une carte complète ($DRI_{ab1.ab1.4.1}$). Une fois la section identifiée, une carte ayant pour désignation le nom de cette section est ajoutée dans le modèle de cartes ($DRI_{ab1.ab1.4.2}$). Elle est la cible d'un lien d'affinement qui a pour source la section affinée.

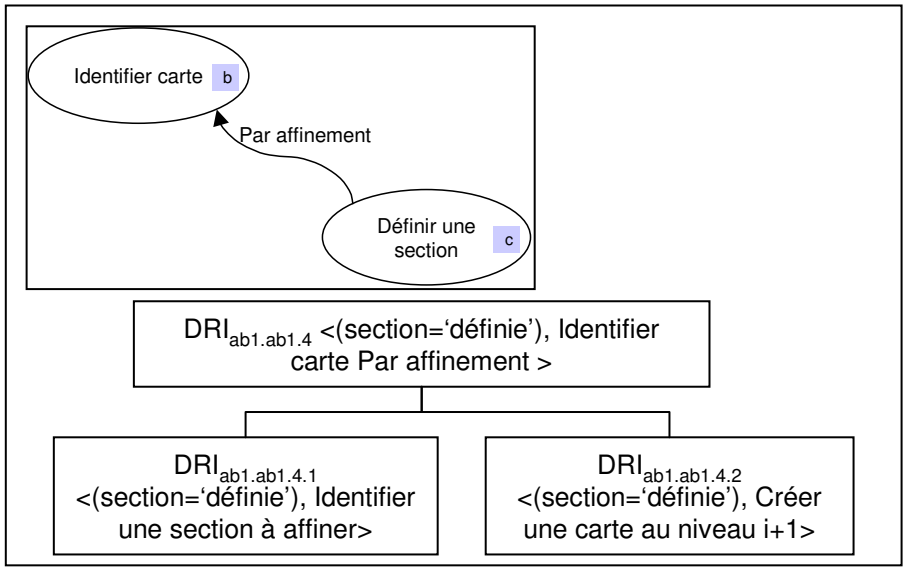


Figure 90 : Directive de réalisation d'intention aidant à Identifier carte par affinement

5.2.3.7 Progresser vers Définir une section

La directive <(*section = définie*), *Progresser vers Définir une section*> permet de boucler sur l'intention *Définir une section*, c'est-à-dire de définir une section à partir d'une section existante.

Cinq alternatives sont proposées : *Par description*, par *Stratégie d'alternative*, *Par abstraction*, *Par précédence/succession* et *Par définition de classes*. Chacune de ces cinq alternatives correspond à une feuille de l'arbre de choix de la directive de sélection de stratégie permettant de progresser vers Définir une section (Figure 91).

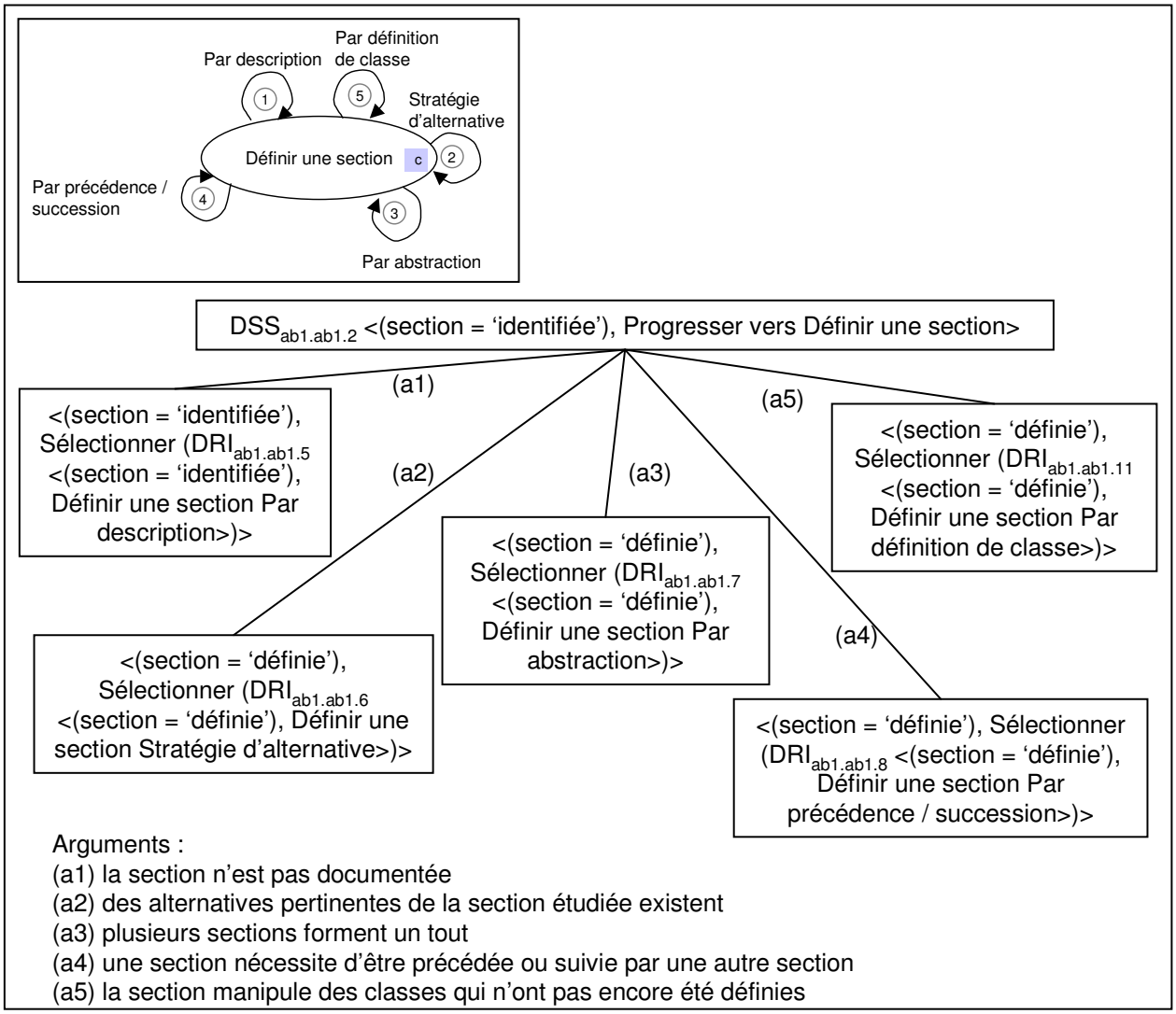


Figure 91 : Directive de sélection de stratégie permettant de progresser vers Définir une section

Les cinq directives feuilles sont des directives tactiques. Elles sont décrites dans les sections suivantes.

5.2.3.8 Définir une section Par description

La description d'une section permet de spécifier : les pré-conditions, les post conditions, les règle de gestions, l'acteur déclencheur et les ressources. Il est également possible d'identifier les états de l'intention source et/ou de l'intention cible.

La directive aidant à construire une section par description se présente sous la forme d'une directive plan où chacune des sous-directives correspond à l'identification d'une propriété particulière de la section et peut être décrite par une directive simple informelle. La Figure 92 présente cette directive.

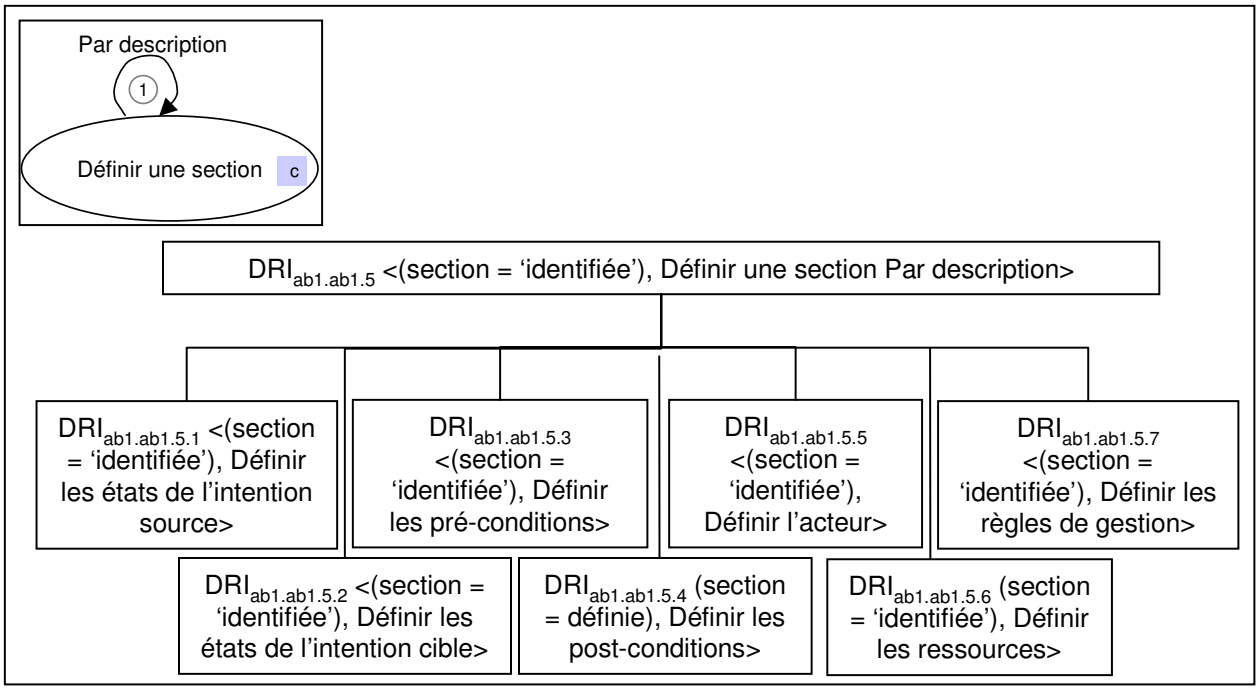


Figure 92 : Directive de réalisation d'intention correspondant à la Construction d'une section par documentation

Les deux premières sous-directives correspondent à la définition des états des intentions source et cible.

La définition des pré et post-conditions consiste à déterminer deux sous-ensembles d'états parmi les états désirables associés respectivement à l'intention source et à l'intention cible de la section.

La définition de l'acteur ainsi que celle des ressources revient à spécifier l'acteur déclencheur de l'activité modélisée par la section et les ressources utilisées lors de l'exécution.

La dernière sous-directive propose de définir les règles de gestion en utilisant les états composant les intentions source et cible de la section.

5.2.3.9 Définir une section Stratégie d'alternative

La directive de réalisation d'intention <(*section = 'définie'*), *Définir une section Stratégie d'alternative*> correspond à la définition d'une section en utilisant la stratégie d'alternative. Elle se présente sous la forme d'une hiérarchie de directives. L'ingénieur doit d'abord faire un choix entre travailler sur l'intention cible ou sur la stratégie. Chacune des directives associées à ce choix se présente sous la forme d'une directive tactique.

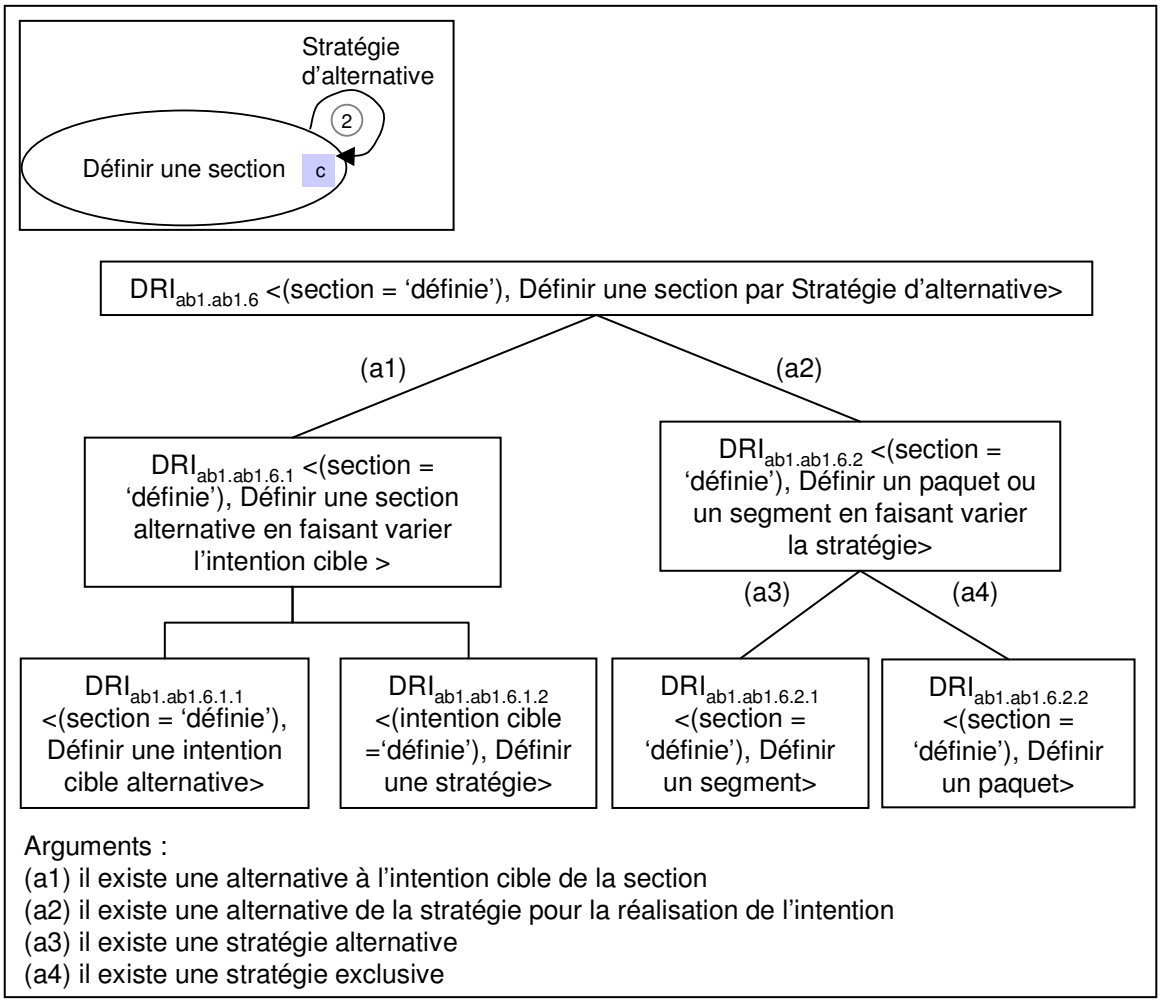


Figure 93 : Directive de réalisation d'intention correspondant à la Définition d'une section Stratégie d'alternative

$DRI_{ab1.ab1.6.1}$ <(section = 'définie'), Définir une section alternative en faisant varier l'intention cible>

Cette directive propose de définir une intention cible alternative en s'appuyant sur l'ensemble des états associés à cette intention ($DRI_{ab1.ab1.6.1.1}$), puis de spécifier une stratégie permettant d'atteindre cette intention à partir de l'intention source de la section de référence ($DRI_{ab1.ab1.6.1.2}$).

$DRI_{ab1.ab1.6.2}$ <(section = 'définie'), Définir un paquet ou un segment en faisant varier la stratégie>

Cette directive correspond à la définition d'une section alternative en faisant varier la stratégie. Il s'agit alors d'identifier une manière alternative d'atteindre l'intention cible à partir de l'intention source. Cette alternative peut être reliée à la précédente pour former un segment ou un paquet. Dans ce dernier cas, les stratégies sont mutuellement exclusives.

5.2.3.10 Définir une section Par abstraction

Il est parfois nécessaire de transformer la carte afin de l'améliorer. Lorsque deux intentions aboutissent à la même partie de produit ou qu'une intention peut être perçue comme un

moyen d'en atteindre une autre, il est nécessaire d'abstraire les sections auxquelles appartiennent ces intentions afin de n'en former qu'une seule.

La Figure 94 présente la structure de la directive de réalisation d'intention $DRI_{ab1.ab1.7}$ permettant de Définir une section Par abstraction.

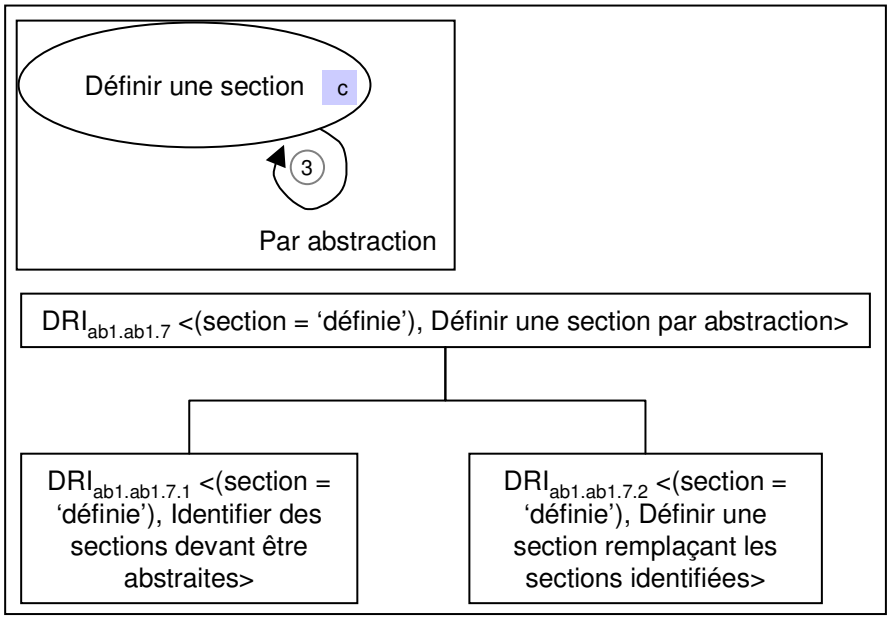


Figure 94 : Directive de réalisation d'intention permettant de Définir une section Par abstraction

Chacune des deux sous-directives de cette directive plan est une directive informelle.

$DRI_{ab1.ab1.7.1}$ <(section = 'définie'), Identifier des sections devant être abstraites>

Cette directive s'appuie sur l'étude (i) des états associés aux différentes intentions afin d'identifier des intentions qui correspondent aux mêmes ensembles d'états et (ii) des différentes intentions afin d'identifier une intention qui est un moyen d'atteindre une autre intention. Dans chacun des deux cas, les sections doivent être abstraites et remplacées par une section unique.

$DRI_{ab1.ab1.7.2}$ <(section = 'définie'), Définir une section remplaçant les sections identifiées>

La directive précédente a permis de mettre en évidence des sections à abstraire. La directive $DRI_{ab1.ab1.7.2}$, permet de remplacer ces sections par une autre section plus abstraite.

5.2.3.11 Définir une section Par précédence/succession

La directive <(*section='définie'*), *Définir une section par précédence/succession*> propose de définir une section par précédence ou par succession de deux façons différentes. L'une s'appuie sur les pré-requis de la section étudiée, l'autre sur les actions postérieures qu'elle implique. La Figure 95 montre la structure de la directive choix permettant de Définir une section par précédence/succession.

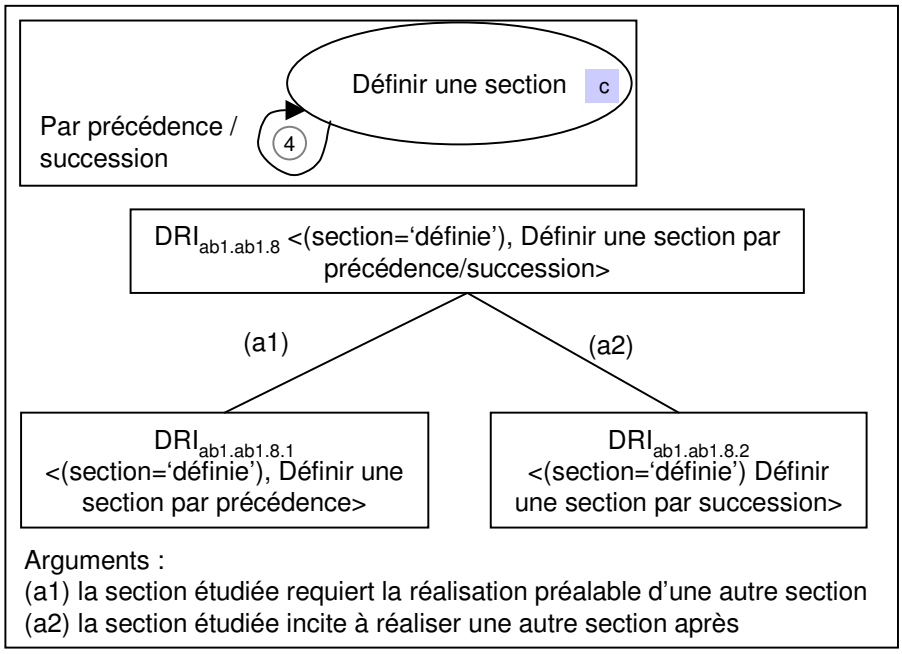


Figure 95 : Directive de réalisation d'intention permettant de Définir une section par précédence/succession

Chacune des deux sous-directives est décrite par une directive tactique.

$DRI_{ab1.ab1.8.1}$ <(section = 'définie'), Définir une section par précédence>

La définition d'une carte par précédence nécessite : (i) d'identifier les pré-requis d'une section donnée, (ii) de construire une section satisfaisant ces pré-requis. Chacune de ces deux sous-directives est de type choix. La directive $DRI_{ab1.ab1.8.1}$ se présente sous la forme d'une hiérarchie de directives (Figure 96).

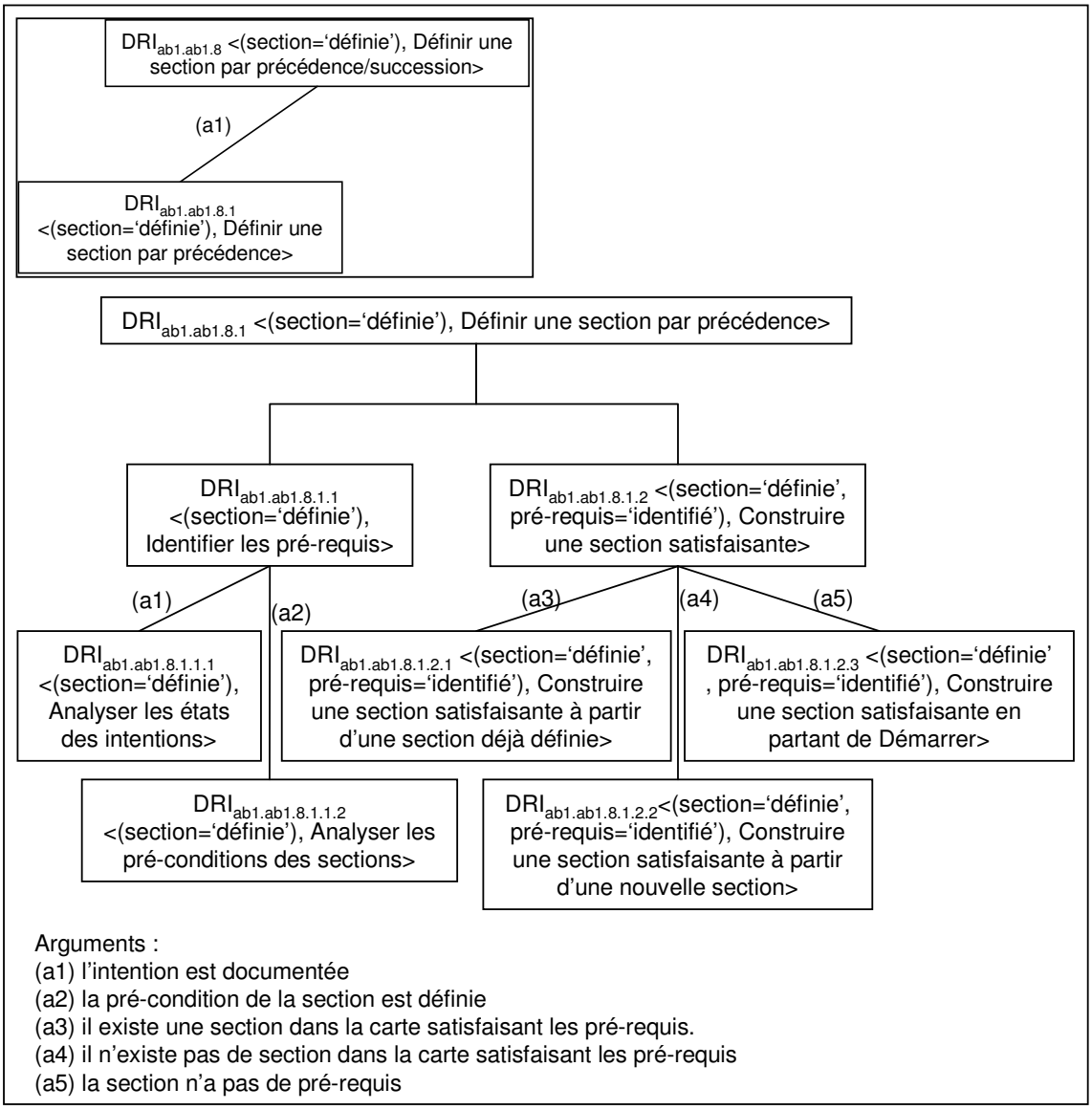


Figure 96 : Hiérarchie de directives permettant de Définir une section par précédence

$DRI_{ab1.ab1.8.1.1}$ <(section = 'définie'), Identifier les pré-requis>

L'identification des pré-requis peut se faire de deux façons différentes : en analysant les états de l'intention source ou en analysant les pré-conditions de la section étudiée. Il s'agit dans chacun des deux cas d'identifier si la réalisation de la section étudiée nécessite la réalisation d'une autre section.

Pour une section donnée, une fois les pré-requis identifiés, il est nécessaire de construire une section permettant de réaliser ces pré-requis.

$DRI_{ab1.ab1.8.1.2}$ <(section = 'définie', pré-requis = 'identifié'), construire une section satisfaisante>

Les pré-requis identifiés peuvent être satisfaits de deux façons différentes :

- par une autre section déjà définie ; dans ce cas, la section que l'on construit permet de lier les deux sections.

- par une nouvelle section qu'on construit ; cette section a pour cible l'intention source de la section étudiée.

Une autre alternative est possible correspondant au cas où la section ne nécessite aucun pré-requis. Dans ce cas, elle doit être reliée à l'intention *Démarrer* ce qui revient à définir une section entre *Démarrer* et l'intention source de la section étudiée.

$DRI_{ab1.ab1.8.2}$ <(section = 'définie'), Définir une section par succession>

Cette directive est analogue à la directive $DRI_{ab1.ab1.8.1}$. Elle s'appuie sur l'analyse des post-conditions de la section ou des états de l'intention cible. Elle permet de définir une carte à partir d'une section existante, d'une nouvelle section ou en reliant la section étudiée à *Arrêter*.

5.2.3.12 Définir une section Par définition de classes

La directive $DRI_{ab1.ab1.11}$ <(*section = 'définie'*), *Définir une section Par définition de classe*> a pour but de définir les classes correspondant aux objets métier ou aux objets système manipulés respectivement par les processus et le système. La directive $DRI_{ab1.ab1.11}$ se présente sous la forme d'une hiérarchie de directives et se décompose en deux sous-directives correspondant respectivement à l'identification et à la définition de classes. Ces deux sous-directives sont des directives tactiques. La Figure 97 présente la structure de la directive $DRI_{ab1.ab1.11}$.

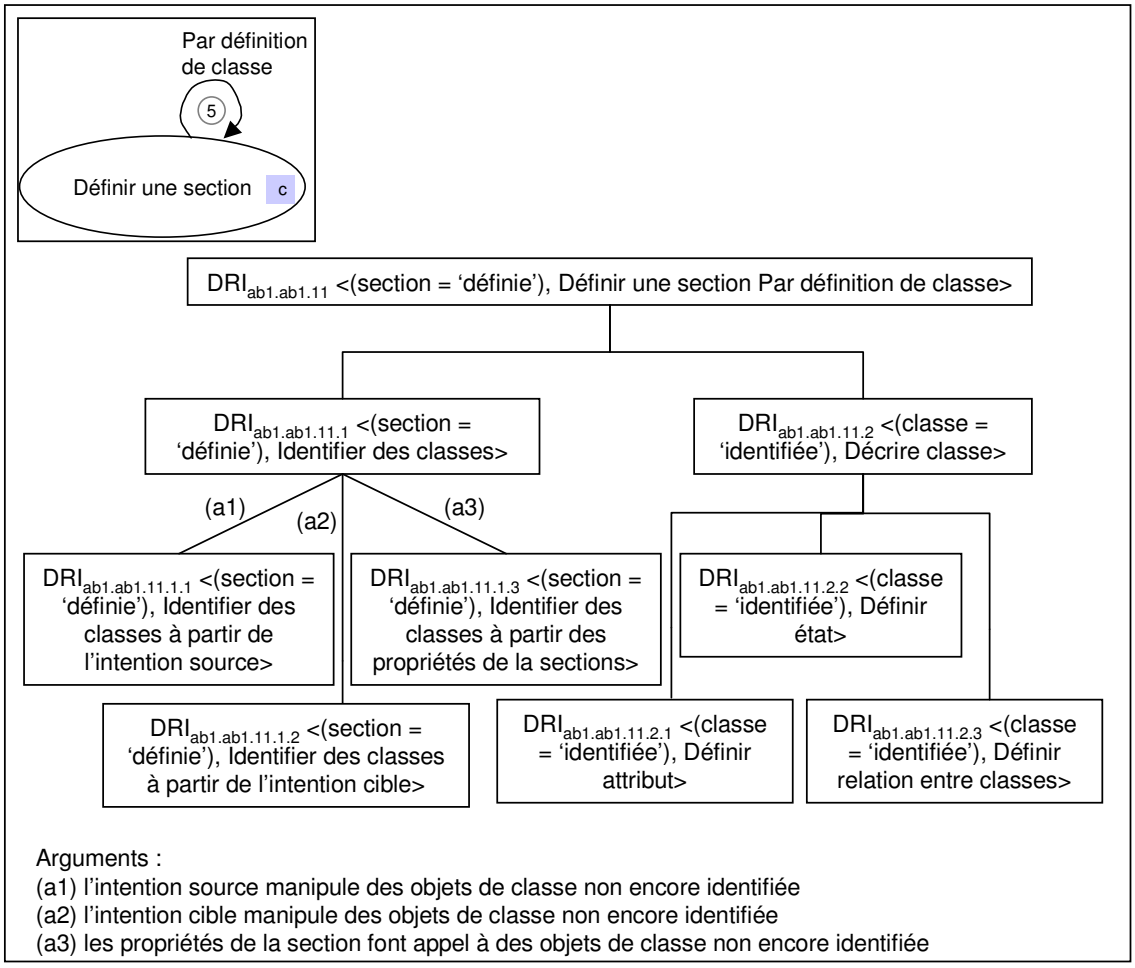


Figure 97 : Hiérarchie de directives permettant de Définir une section Par définition de classe

$DRI_{ab1.ab1.11.1}$ <(section = 'définie'), Identifier des classes>

Cette directive se présente sous la forme d'une directive choix. Elle propose différentes alternatives correspondant respectivement à l'identification de classes à partir de l'étude de l'intention source, de l'intention cible ou des propriétés de la section. Pour chacun des cas, il s'agit d'identifier les objets métier ou les objets système manipulés par les processus ou le système.

$DRI_{ab1.ab1.11.2}$ <(classe = 'identifiée'), Décrire classe>

Cette directive se présente sous la forme d'une directive plan et se décompose en trois sous-directives. Celles-ci correspondent respectivement à la définition d'attributs ou d'états relatifs à la classe identifiée ou bien à la définition de relations entre classes.

5.2.3.13 Progresser vers Arrêter

La directive <(*section='définie', carte='construite'*), *Progresser vers Arrêter*> propose l'alternative suivante pour arrêter le processus de construction d'une carte : (i) en appliquant les règles de validité ou (ii) en vérifiant les invariants.

La Figure 98 présente la directive de sélection de stratégie permettant de progresser vers *Arrêter*.

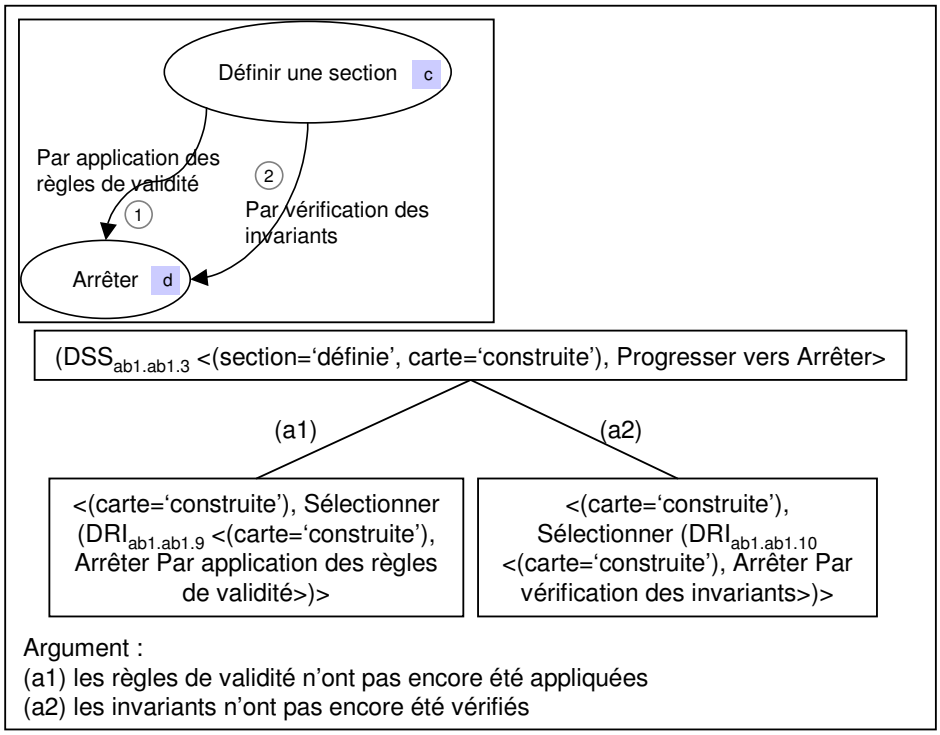


Figure 98 : Directive de sélection d'intention permettant de progresser depuis Elaborer une carte

5.2.3.14 Arrêter Par application des règles de validité

Toute carte doit vérifier les règles de validité présentées au chapitre 6. La directive $DRI_{ab1.ab1.9}$ <(*carte='construite'*), *Arrêter Par application des règles de validité*> propose d'appliquer les règles de validité d'une carte. Cette directive se décompose en deux sous-directives :

- la première permet de vérifier une règle de validité ;

- la deuxième consiste à modifier la carte afin que la règle soit vérifiée. Cette directive n'est réalisée que si, dans l'état, la carte ne vérifie pas la règle de validité.

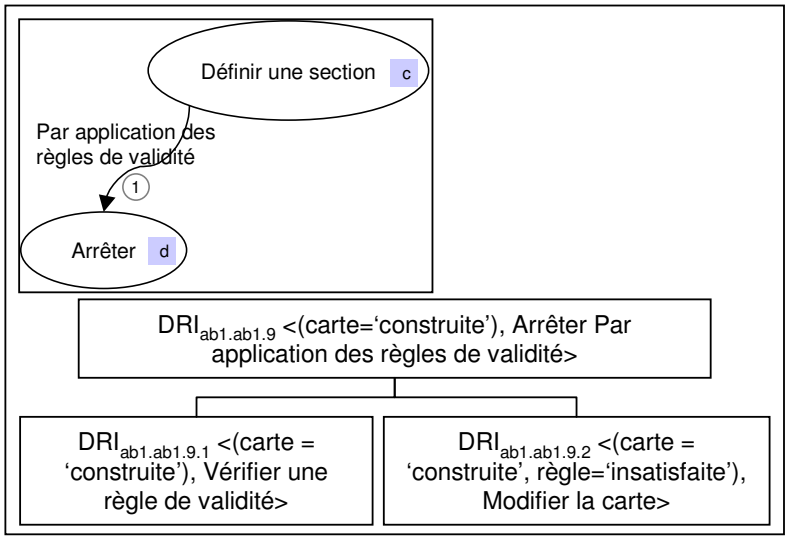


Figure 99 : Directive de réalisation d'intention permettant d'Arrêter Par application des règles de validité.

5.2.3.15 Arrêter Par vérification des invariants

La directive <(*carte='construite'*), *Arrêter Par vérification des invariants*> est construite de la même façon que la directive précédente. Elle se présente sous la forme d'une directive plan qui permet d'une part de vérifier les invariants présentés au chapitre 6 et ensuite de modifier la carte pour que ceux-ci soient vérifiés (Figure 100).

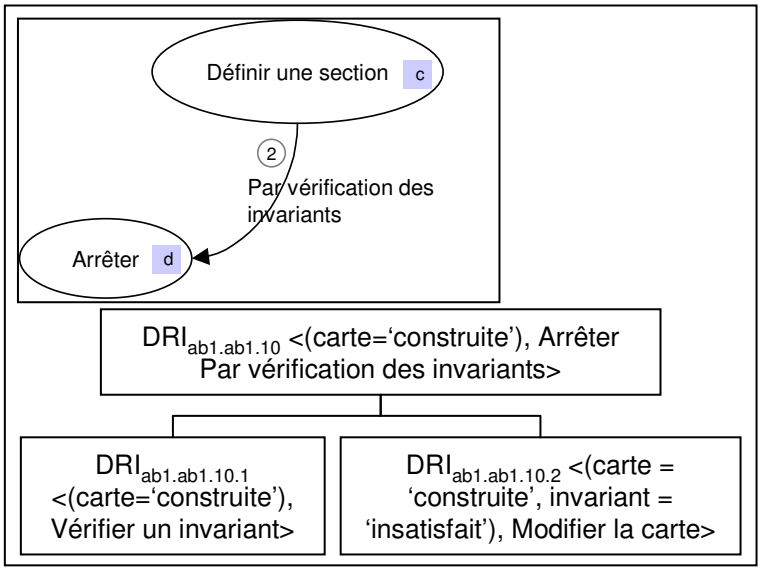


Figure 100 : Directive de réalisation d'intention permettant d'Arrêter Par vérification des invariants

5.2.4 Construire une carte du système Par actualisation

La directive <(*carte système = 'construite'*), *Construire carte du système Par actualisation*> permet de construire une carte système ou processus par mise à jour ce qui suppose qu'il

existe déjà une telle carte. La Figure 101 présente la structure de cette directive de réalisation d'intention.

La réalisation de la directive $DRI_{ab1.1}$ se décompose en plusieurs directives permettant de :

- identifier les changements sur le système ($DRI_{ab1.1.1}$). En effet, si la carte du système doit être mise à jour c'est que la version existante du modèle ne correspond pas exactement au système dans sa version actuelle. Certains changements ont eu lieu sur ce dernier sans avoir été répercutés sur le modèle.
- répercuter le changement sur la carte ($DRI_{ab1.1.2}$) ;
- appliquer les règles de validité ($DRI_{ab1.1.3}$) ;
- vérifier que la nouvelle carte est correcte, c'est-à-dire vérifier les invariants ($DRI_{ab1.1.4}$).

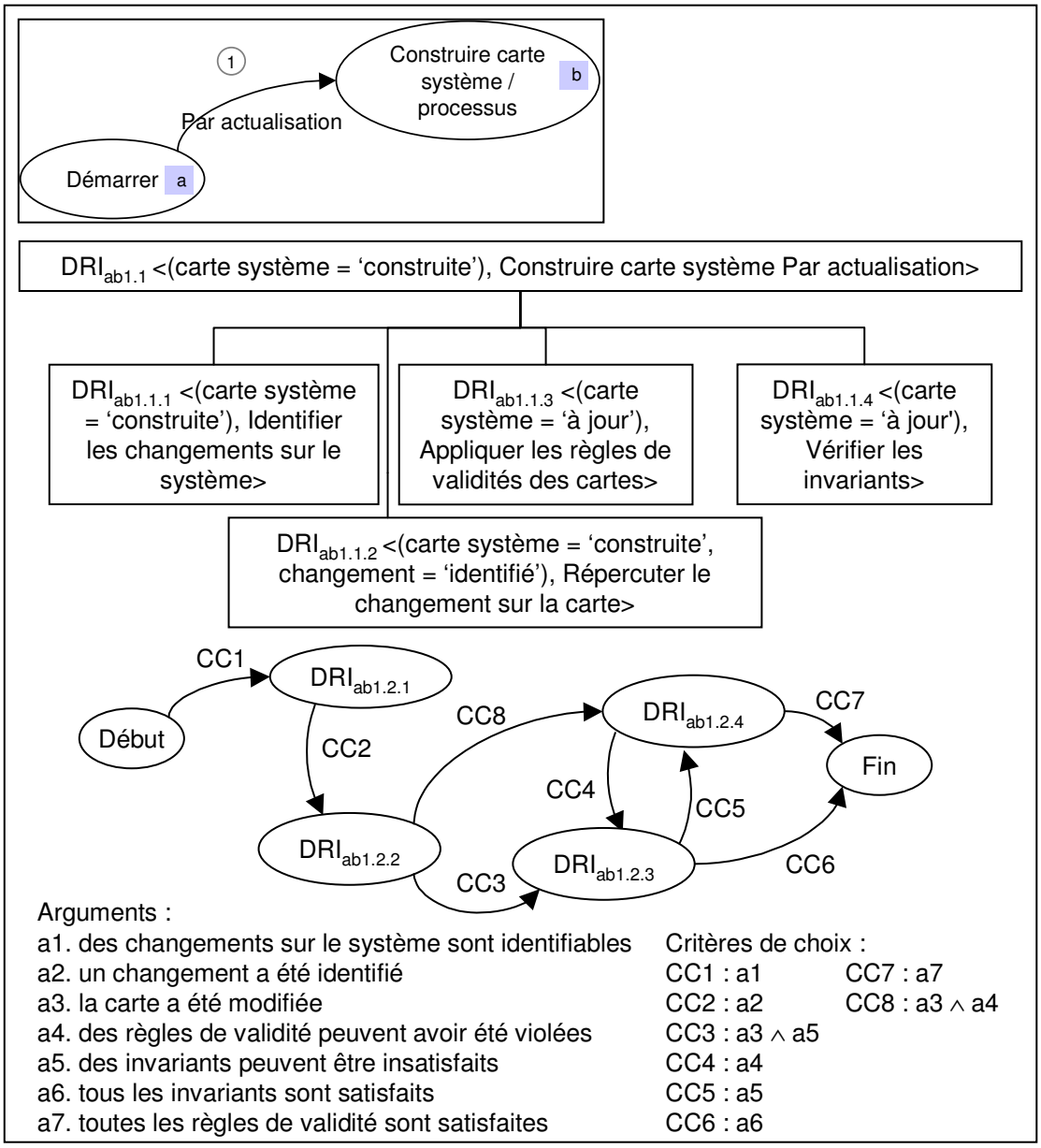


Figure 101 : Directive de réalisation d'intention aidant à Construire une carte système Par mise à jour

Ces sous-directives sont des directives tactiques et sont décrites dans les paragraphes suivants.

$DRI_{ab1.1.1}$ <(carte système = 'construite'), Identifier les changements sur le système>

Cette directive a pour but de mettre en évidence les changements qui ont eu lieu sur le système sans être retranscrits sur le modèle existant. Elle se présente sous la forme d'un choix comportant trois alternatives : (i) identifier les changements par rétro-ingénierie, (ii) identifier les changements par interview et (iii) identifier les changements par analyse des évolutions.

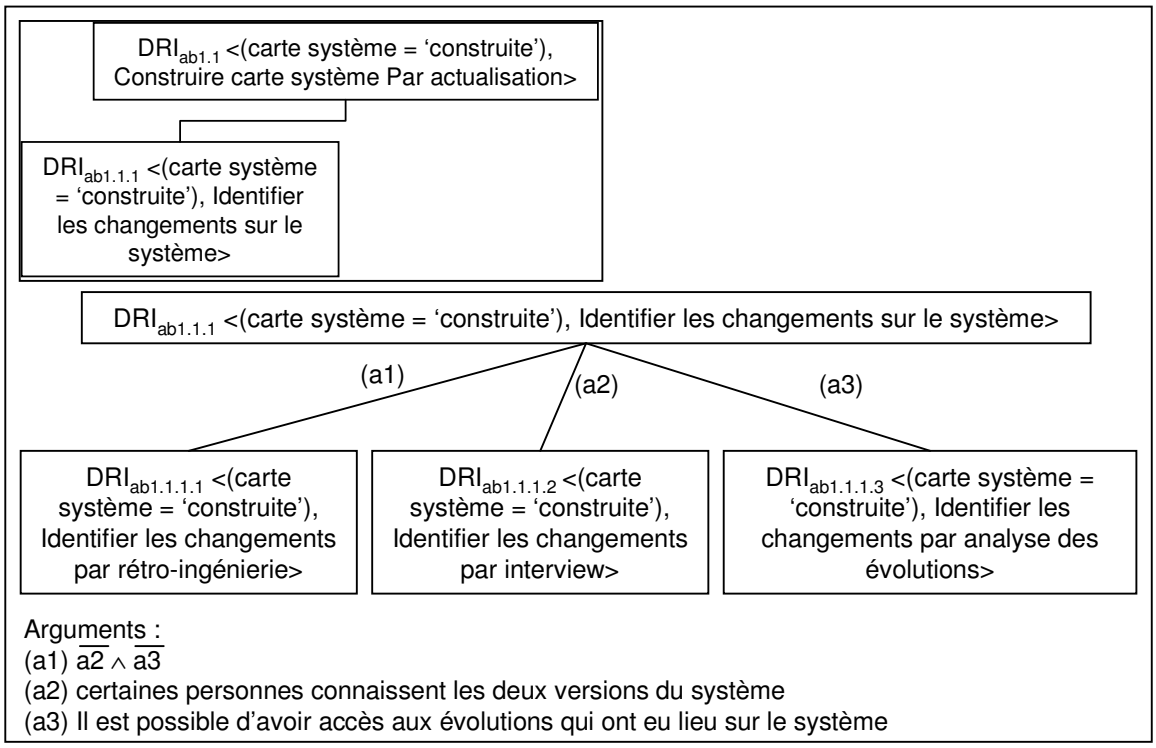


Figure 102 : Directive choix correspondant à l'Identification des changements sur le système

Une fois le changement identifié, il faut le répercuter sur la carte.

$DRI_{ab1.1.2}$ <(carte système = 'construite', changement = 'identifié'), Répercuter le changement sur la carte>

La Figure 103 présente la structure de la directive permettant de répercuter le changement identifié sur la carte système.

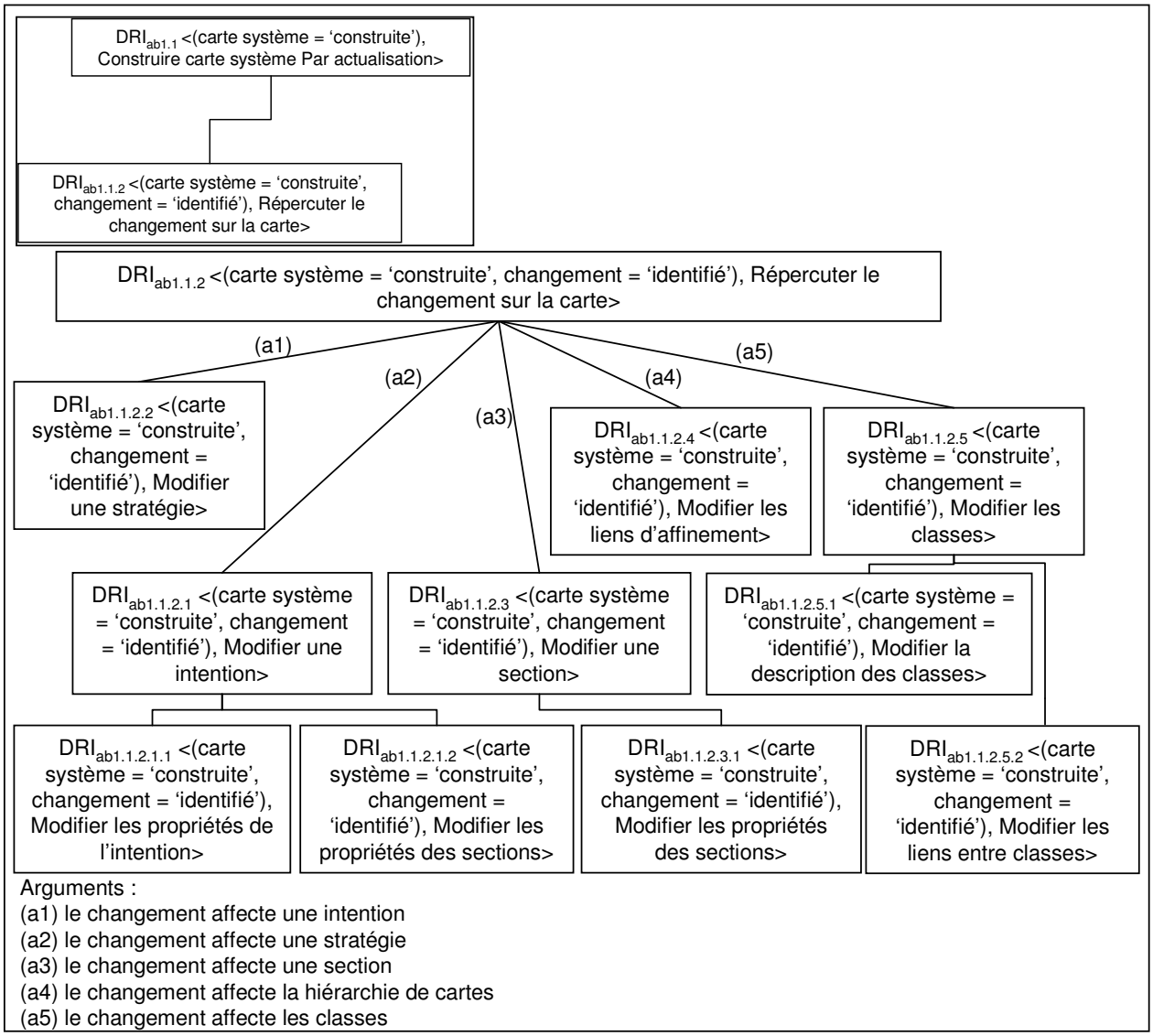


Figure 103 : Hiérarchie de directives permettant de Répercuter le changement sur la carte

Afin de répercuter le changement identifié sur une carte, plusieurs alternatives sont possibles suivant l'élément de la carte affectée par le changement :

1. modifier une intention ; dans ce cas, il est nécessaire de modifier les états composant l'intention ($DRI_{ab1.1.2.1.1.1}$) et, éventuellement, de modifier les propriétés des sections ($DRI_{ab1.1.2.1.1.2}$).

2. modifier une stratégie ; il s'agit, ici, de changer le nom de la stratégie ou de changer sa source ou sa cible.

3. modifier une section ; dans ce cas, les différentes propriétés de la section doivent être adaptées afin de prendre en compte les changements.

4. modifier les liens d'affinement ; dans ce cas, les liens entre les différentes cartes sont modifiés. En particulier, certaines cartes peuvent être supprimées et d'autres ajoutées.

5. modifier les classes ; dans ce cas, il peut être nécessaire de modifier la description d'une classe (en changeant des attributs ou des états) ou de modifier les liens entre classes.

Une fois la carte modifiée, il est nécessaire de vérifier les invariants et d'appliquer les règles de validité.

$DRI_{ab1.1.3}$ <(carte système = 'à jour'), Appliquer les règles de validité des cartes>

Cette directive est identique à la directive $DRI_{ab1.ab1.9}$ décrite à la section 5.2.3.14 (page 234).

$DRI_{ab1.1.4}$ <(carte système = 'à jour'), Vérifier les invariants>

Cette directive est semblable à la directive décrite à la section 5.2.3.15 (page 235).

5.2.5 Construire le modèle pivot A la volée

La directive <(*rien*), *Construire la carte pivot A la volée*> offre la possibilité de construire le modèle pivot directement, c'est-à-dire sans s'appuyer sur les modèles As-Is ni utiliser une version antérieure du modèle pivot. C'est le cas lorsque les modèles As-Is sont inexploitables. Plusieurs raisons peuvent expliquer une telle situation :

- les modèles n'ont pas été mis à jour depuis longtemps ;
- les modèles sont écrits dans une langue étrangère. C'est souvent le cas lorsque le système d'information est modifié après une fusion ou une acquisition d'une organisation étrangère ;
- aucun modèle n'a jamais existé.

Pour construire le modèle pivot à la volée, deux alternatives sont proposées : (i) par analyse des différences système/processus ($DRI_{ab1.2.1}$) ou (ii) par mise en évidence des objectifs ($DRI_{ab1.2.2}$). La directive $DRI_{ab1.2.2}$ se concentre sur les parties communes au système et au processus. La directive $DRI_{ab1.2.1}$, au contraire, s'intéresse aux différences qui existent entre le système et les processus. Chacune de ces deux directives est décrite par une directive stratégique analogue à la directive $DRI_{ab1.2}$ <(*carte système / processus='construite'*), *Construire carte Par abstraction*> détaillée à la section 5.2.3. Les différences entre les deux directives ont deux origines : (i) on se sert des résultats d'interviews ou de réunions de travail menées pour récolter l'information nécessaire à la construction du modèle pivot (ii) les couleurs des éléments de la carte construite sont différentes : blanc pour la directive $DRI_{ab1.2.2}$ (ce qui signifie que les sections sont communes au système et aux processus) et gris clair ou gris foncé pour la directive $DRI_{ab1.2.1}$ pour préciser l'origine de la section.

La Figure 104 présente la structure de la directive $DRI_{ab1.2}$.

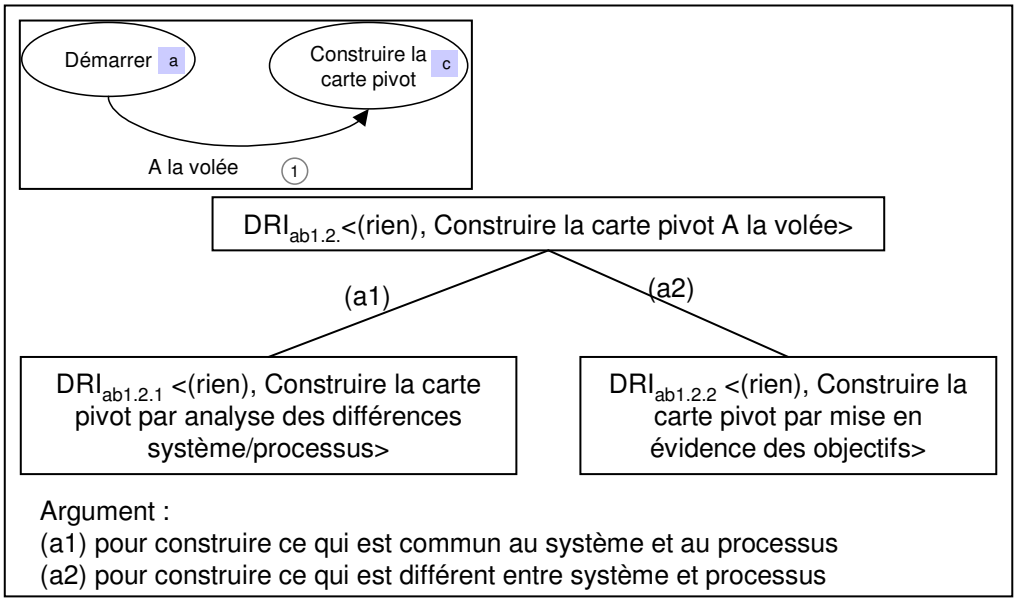


Figure 104 : Directive de réalisation d'intention permettant de Construire la carte pivot A la volée

5.2.6 Progresser vers Construire carte pivot

La directive <(*carte système/processus = 'construite' ou carte système/processus = 'à jour'*), *Progresser vers Construire carte pivot*> propose une alternative pour poursuivre le processus de construction du modèle pivot une fois que les cartes système et les cartes processus sont construites. La Figure 105 montre la structure de cette directive de sélection de stratégie ainsi que les arguments en faveur de chacune des propositions.

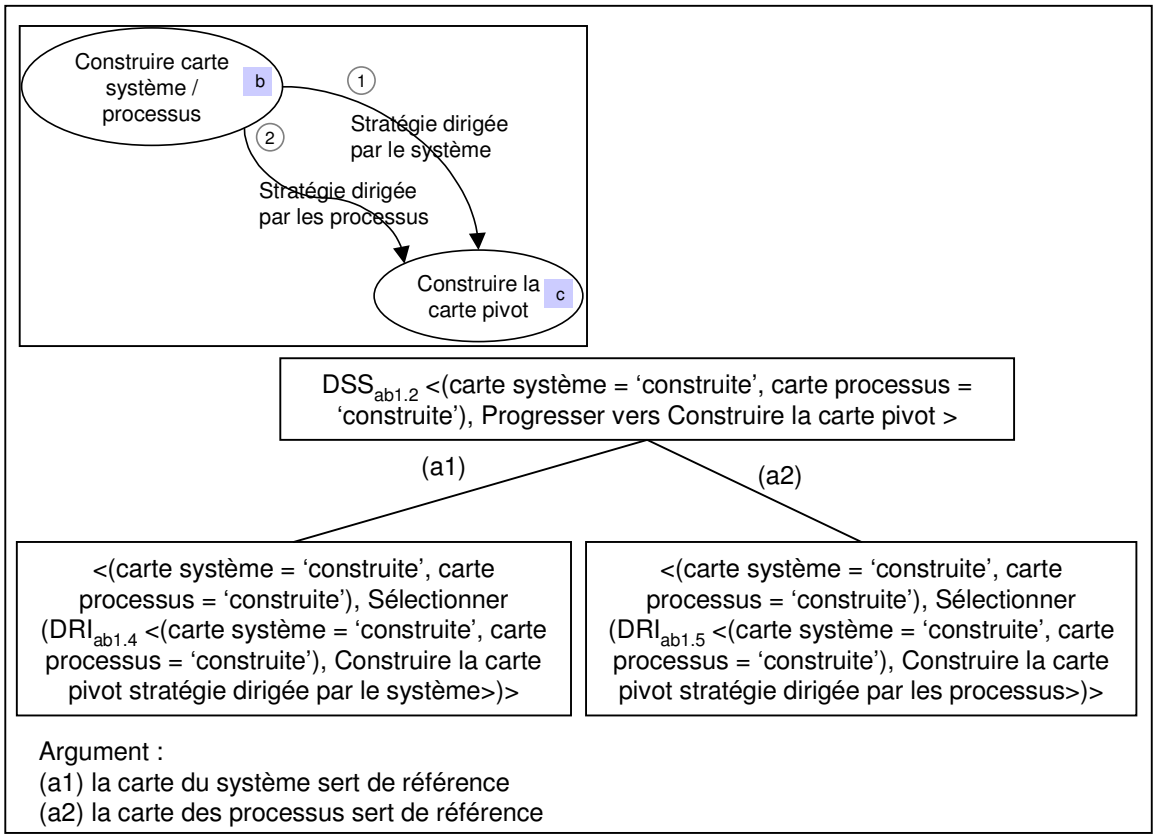


Figure 105 : Directive de sélection de stratégie permettant de progresser vers Construire la carte pivot

Ces deux directives de réalisation d'intention sont des directives tactiques. Les directives $DRI_{ab1.4}$ et $DRI_{ab1.5}$ s'exécutent de façon comparable. La seule différence réside dans le fait que dans le premier cas, la carte système sert de référence à la fusion des cartes système et processus alors que pour la $DRI_{ab1.5}$, la carte processus est considérée comme la référence.

5.2.7 Construire la carte pivot en suivant la Stratégie dirigée par le système

La directive <(*carte système = 'construite', carte processus = 'construite'*), *Construire la carte pivot Stratégie dirigée par le système*> aide à construire les cartes pivot en fusionnant les cartes système et les cartes processus. Plus précisément, les éléments des cartes système sont fusionnés avec ceux des cartes processus s'il existe un fort lien de similarité entre eux. Dans le cas contraire, les éléments des cartes système sont ajoutés. S'il existe des éléments des cartes processus qui ne sont identiques à aucun élément du système, ils sont également ajoutés. De cette façon, la carte pivot résultante représente conjointement le système et les processus. La Figure 106 présente la hiérarchie de directives permettant de construire la carte pivot lorsque les cartes système servent de référence.

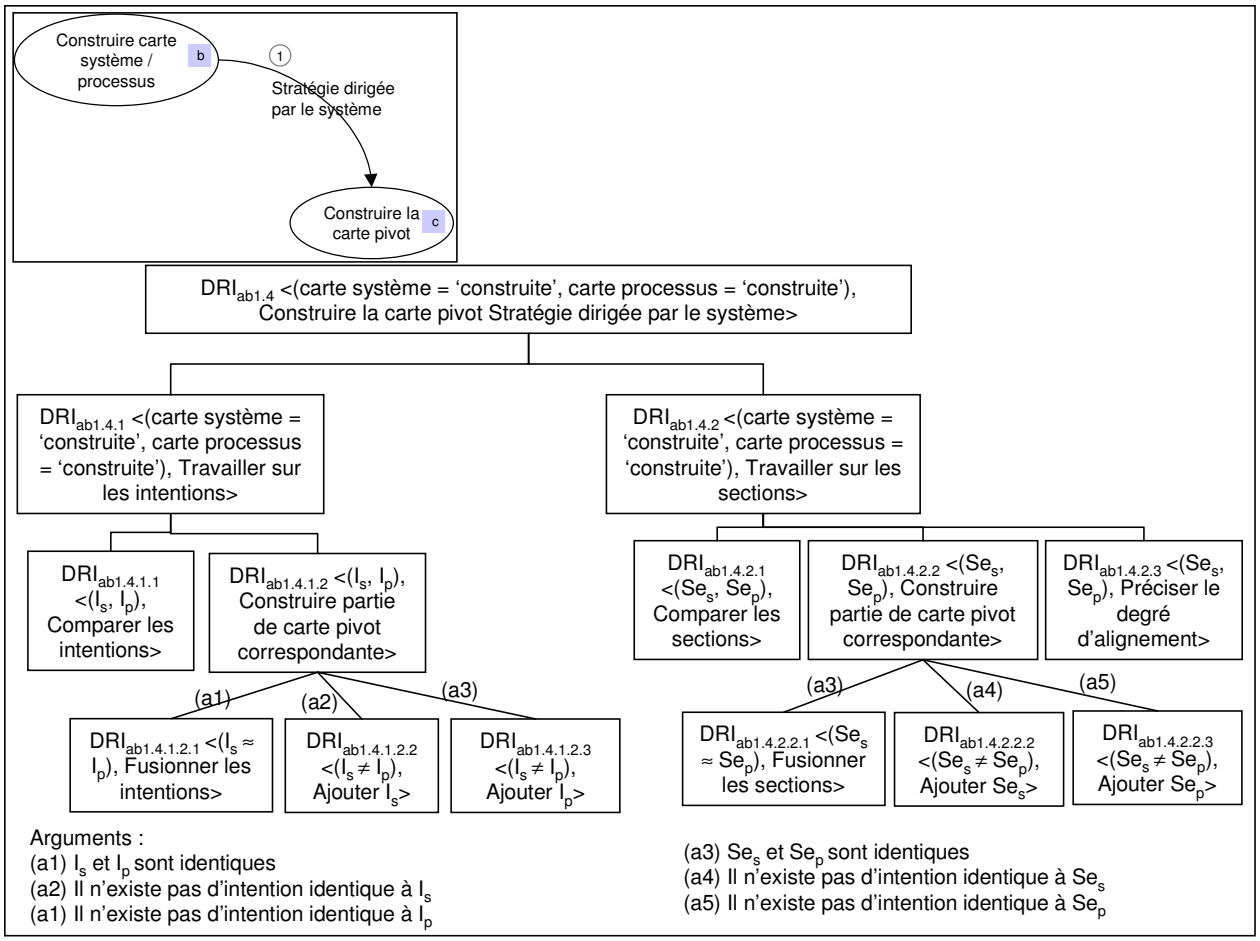


Figure 106 : Hiérarchie de directives permettant de Construire la carte pivot stratégie dirigée par le système

La directive DRI$_{ab1.4}$ se décompose en deux sous-directives formant un plan : DRI$_{ab1.4.1}$ propose de travailler sur les intentions alors que la directive DRI$_{ab1.4.2}$ travaille sur les sections.

5.2.7.1 Construire la carte pivot en travaillant sur les intentions

Cette directive propose de comparer les intentions des cartes système avec celles des cartes processus puis de construire la partie de carte pivot correspondante. Cette directive se présente sous la forme de hiérarchie de directives.

DRI$_{ab1.4.1.1}$ <(I_s, I_p), Comparer les intentions>

Cette directive consiste à chercher si une intention I_s du système est identique à une intention I_p de la carte processus. Cette comparaison pourrait se faire de façon automatisée en utilisant le calcul de similarité de la méthode MIBE [Zoukar05] ou celui proposé par [Ralyté01]. La présentation de ces deux approches est en dehors du propos de cette thèse.

On peut néanmoins préciser qu'une intention du système et une intention des processus sont identiques si elles ont le même nom et des sémantiques similaires ;

$DRI_{ab1.4.1.2}$ <(I_s, I_p), Construire partie de carte pivot correspondante>

Cette directive tactique se présente sous la forme d'un choix (Figure 106). Il s'agit de construire la partie de carte correspondante. En effet, la réalisation de cette directive se fait de façons différentes suivant qu'il existe une intention I_p identique.

$DRI_{ab1.4.1.2.1}$ <(I_s=I_p), Fusionner les intentions>

Cette directive n'est appliquée que s'il existe dans la carte processus une intention I_p similaire à I_s. La réalisation de cette directive correspond à la fusion des intentions I_s et I_p à l'aide de l'opérateur *FusionnerIntention*. L'intention résultante est coloriée en blanc pour préciser que les intentions sont identiques dans les cartes système et processus.

$DRI_{ab1.4.1.2.2}$ <($I_s \neq I_p$), Ajouter I_s>

Cette directive s'applique lorsque aucune intention d'une carte des processus n'est identique à l'intention I_s des cartes système. Dans ce cas, l'intention I_s est ajoutée dans la carte pivot en utilisant l'opérateur *AjouterIntention*. L'intention résultante est coloriée en gris clair afin de préciser que cette intention n'existe que dans le système.

$DRI_{ab1.4.1.2.3}$ <($I_s \neq I_p$), Ajouter I_p>

Cette directive s'applique lorsque toutes les intentions des cartes système ont été étudiées. S'il existe des intentions I_p des cartes processus qui n'ont pas été fusionnées avec des intentions du système, elles sont ajoutées dans la carte pivot en utilisant l'opérateur *AjouterIntention* et sont coloriées en gris foncé pour préciser qu'elles n'existent que dans les cartes processus.

5.2.7.2 Construire la carte pivot en Travaillant sur les sections

Cette directive est semblable à la directive $DRI_{ab1.4.1}$ mais travaille sur les sections. Elle propose de comparer les sections des cartes système avec celles des cartes processus puis de construire la partie de carte pivot correspondante. Cette directive se présente sous la forme d'une hiérarchie de directives.

$DRI_{ab1.4.1.2}$ <(Se_s, Se_p), Comparer les sections>

Cette directive est similaire à la directive $DRI_{ab1.4.1.1}$. On note cependant :

- qu'une stratégie du système et une stratégie des processus sont identiques si elles ont le même nom et si elles ont pour source (respectivement pour cible) deux intentions identiques.
- qu'une section de processus est identique à une section du système si chacun des trois éléments les composant (intention source, intention cible et stratégie) sont identiques.

Les directives $DRI_{ab1.4.2.2.1}$, $DRI_{ab1.4.2.2.2}$ et $DRI_{ab1.4.2.2.3}$ sont respectivement similaires aux directives $DRI_{ab1.4.1.2.1}$, $DRI_{ab1.4.1.2.2}$ et $DRI_{ab1.4.1.2.3}$. Ces deux groupes de directives ne travaillant pas sur les mêmes éléments de carte, ce ne sont pas les mêmes opérateurs qui sont utilisés. La directive $DRI_{ab1.4.2.2.1}$ utilise l'opérateur *FusionnerSection* et les directives $DRI_{ab1.4.2.2.2}$ et $DRI_{ab1.4.2.2.3}$ propose d'appliquer l'opérateur *AjouterSection*.

$DRI_{ab1.4.2.3}$ <(Se_s, Se_p), Préciser le degré d'alignement>

Cette directive est informelle et a pour but de préciser le degré d'alignement.

- pour chaque section entièrement coloriée en blanc attribuer le degré d'alignement *totalement aligné*.
- pour chaque section coloriée en gris clair, si son intention source et/ou son intention cible est coloriée en blanc, identifier une ou plusieurs sections issues des processus qui lui est proche. Attribuer à cet ensemble de sections le degré d'alignement *partiellement aligné*.
- pour chaque section coloriée en gris clair dont l'intention source et l'intention cible sont coloriées en gris clair, attribuer le degré d'alignement *pas du tout aligné*.
- pour chaque section coloriée en gris foncé dont l'intention source et l'intention cible sont coloriées en gris foncé, attribuer le degré d'alignement *pas du tout aligné*.

5.2.8 Arrêter Par validation

La directive <(*carte pivot = 'construite'*), *Arrêter Par validation*> aide à terminer le processus de construction du modèle pivot en validant le modèle pivot obtenu. Ainsi, il est important que (i) tout élément des cartes système et processus soient dans le modèle pivot et (ii) le modèle pivot vérifie les invariants et les règles de validité définis au chapitre 6. La Figure 107 présente la structure de la directive plan permettant *d'Arrêter* par validation le processus de construction du modèle pivot.

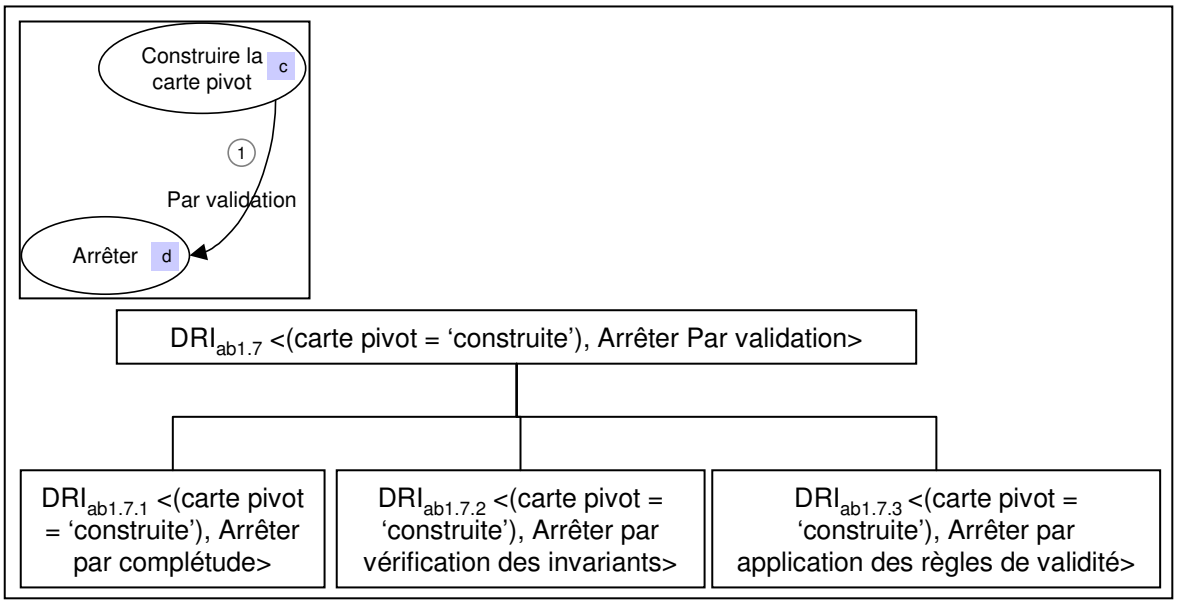


Figure 107 : Directive plan permettant d'Arrêter Par validation le processus de construction du modèle pivot

5.2.8.1 Arrêter par complétude

La directive <(*carte pivot = 'construite'*), *Arrêter par complétude*> est informelle. Il s'agit de vérifier que chaque élément de la carte système ainsi que chaque élément de la carte processus sont présents dans le modèle pivot.

5.2.8.2 Arrêter par vérification des invariants

La directive <(*carte pivot = 'construite'*), *Arrêter par vérification des invariants*> est identique à la directive DRI$_{ab1.ab3.11}$ décrite à la section 5.2.3.15.

5.2.8.3 Arrêter par vérification des règles de validité

La directive <(*carte pivot = 'construite'*), *Arrêter par application des règles de validité*> est identique à la directive $DRI_{ab1.ab3.11}$ décrite à la section 5.2.3.14.

5.3. Construire le modèle pivot Par mise à jour

La directive <(*modèle pivot = 'construit'*), *Construire le modèle pivot Par mise à jour*> est de type plan. Elle se décompose en cinq directives (Figure 108) :

1. identifier les changements à partir du système. Cette directive a pour but de mettre en évidence des changements qui sont survenus sur le système et n'ont pas été répercutés sur le modèle pivot.
2. identifier les changements à partir des processus. Cette directive est semblable à la directive précédente mais s'applique aux processus.
3. répercuter le changement sur le modèle pivot. Cette directive permet de modifier le modèle pivot pour tenir compte d'un changement identifié à partir du système ou à partir des processus.
4. appliquer les règles de validité.
5. vérifier les invariants.

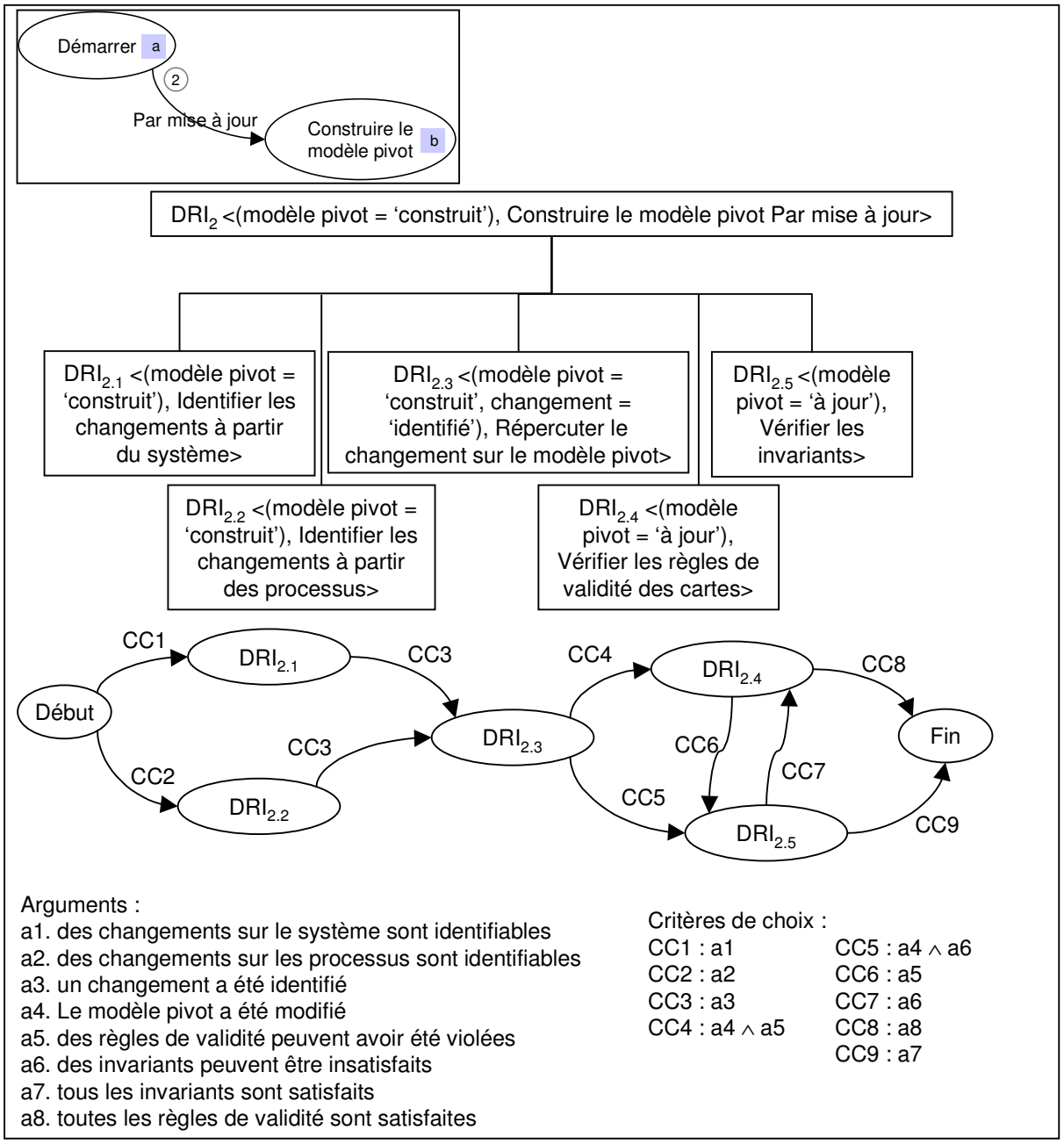


Figure 108 : Directive de réalisation d'intention correspondant à la Construction du modèle pivot par mise à jour

Chacune de ces six directives est semblable à l'une des directives définie précédemment. De façon plus précise les directives $DRI_{2.1}$ et $DRI_{2.2}$ sont toutes deux semblables à la directive $DRI_{ab1.1.1}$. La directive $DRI_{2.4}$ est identique à la directive $DRI_{ab1.1.2}$. Enfin, les directives $DRI_{2.5}$ et $DRI_{2.6}$ ont les mêmes définitions que les directives $DRI_{ab1.1.3}$ et $DRI_{ab1.1.4}$.

La description de cette directive termine la formalisation de la construction du modèle pivot. Les directives suivantes aident à découvrir des exigences d'évolution.

6. Découverte des exigences d'évolution

La carte ACEM de la Figure 78 (page 213) montre qu'il existe plusieurs alternatives pour découvrir des exigences d'évolution. La suite de cette section présente ces différentes alternatives.

6.1. Progresser vers Découvrir des exigences d'évolution à partir de Construire le modèle pivot

Une fois le modèle pivot construit et à jour, la directive <(*modèle pivot = 'à jour'*), *Progresser vers Découvrir les exigences d'évolution*> propose de découvrir des exigences d'évolution. La démarche ACEM offre trois stratégies dont le choix est guidé par la directive de sélection de stratégie présentée à la Figure 109.

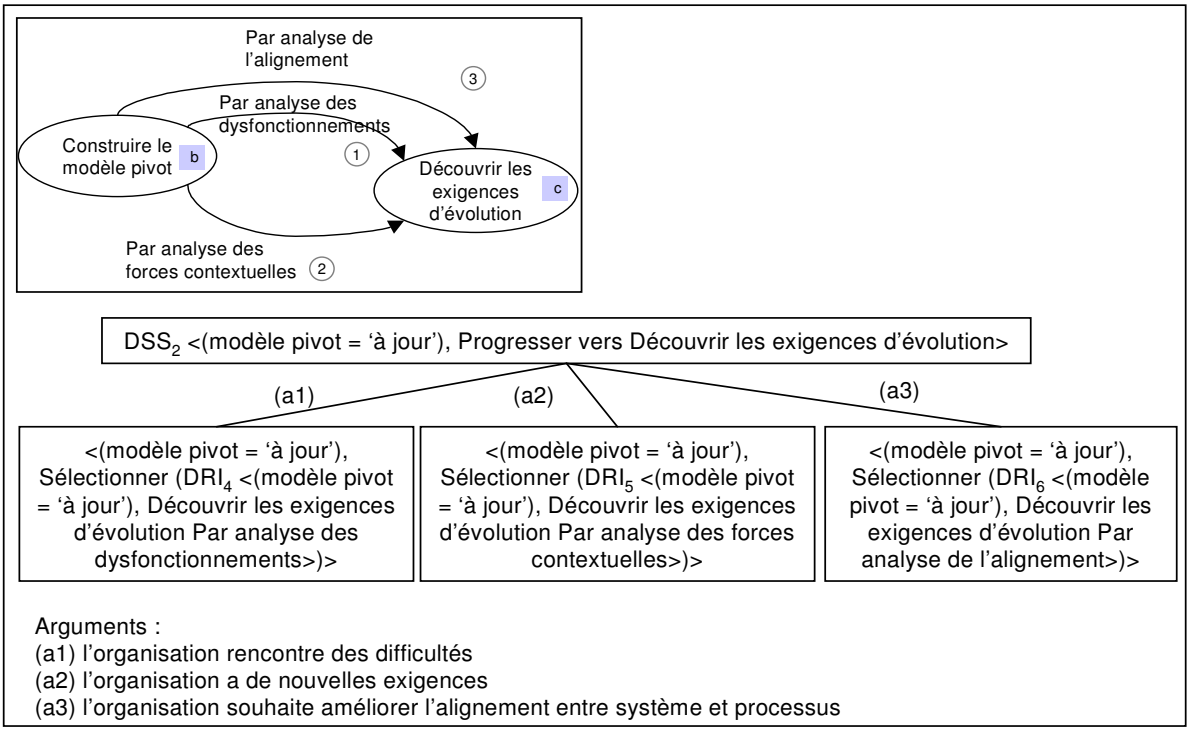


Figure 109 : Directive de sélection pour Progresser vers Découvrir des exigences d'évolution

Les trois sous-directives formant la directive choix permettent respectivement de choisir :

- la DRI_4 qui guide la découverte des exigences d'évolution par analyse des dysfonctionnements.
- la DRI_5 qui aide à découvrir des exigences d'évolution à partir de l'étude des forces contextuelles.
- la DRI_6 qui aide à découvrir des exigences d'évolution en étudiant l'alignement.

La DRI_4 s'intéresse à la découverte d'exigences d'évolution à partir de l'analyse des dysfonctionnements. Cette directive consiste à étudier les processus et le système et à essayer d'identifier les dysfonctionnements ou les améliorations à apporter à l'un ou à l'autre. Il ne

s'agit pas de prendre en compte les problèmes dus à une éventuelle rupture de l'alignement entre les processus et le système.

La directive DRI_5 aide à prendre en compte les forces contextuelles pour découvrir des exigences d'évolution. Ces forces expriment des contraintes. Elles peuvent être de deux types : internes ou externes. Les forces externes sont imposées par l'environnement externe, il s'agit des menaces et des opportunités de l'organisation. Les forces internes résultent d'un besoin de nouveauté. L'identification de ces forces peut se faire par exemple par une analyse SWOT (Strength, Weakness, Opportunity et Thread) [Robson97]. Cette analyse se fait à l'aide d'une matrice permettant d'étudier les forces, les faiblesses, les opportunités et les menaces d'une organisation en évaluant les forces et les faiblesses suivant deux critères, la performance et l'importance.

La DRI_6 aide à améliorer l'alignement entre système et processus sans prendre en compte de nouvelles exigences. La réalisation de cette directive s'appuie sur l'étude de la relation d'alignement afin de découvrir des exigences correspondant à la mise en œuvre d'actions correctives.

6.2. Découvrir les exigences d'évolution par analyse des dysfonctionnements

La directive <(*modèle pivot = 'à jour'*), *Découvrir des exigences d'évolution Par analyse des dysfonctionnements* > aide à découvrir les exigences d'évolution par analyse des dysfonctionnements. Elle se présente sous la forme d'une hiérarchie de directives (Figure 110). Plusieurs alternatives sont proposées. Elles correspondent respectivement à l'analyse des dysfonctionnements du système et à l'analyse des dysfonctionnements des processus. Chacune de ces propositions se décompose en trois directives correspondant à :

- l'identification des dysfonctionnements ;
- l'analyse de l'impact de ces dysfonctionnements sur le reste du modèle ;
- la découverte d'exigences d'évolution à partir de ces dysfonctionnements.

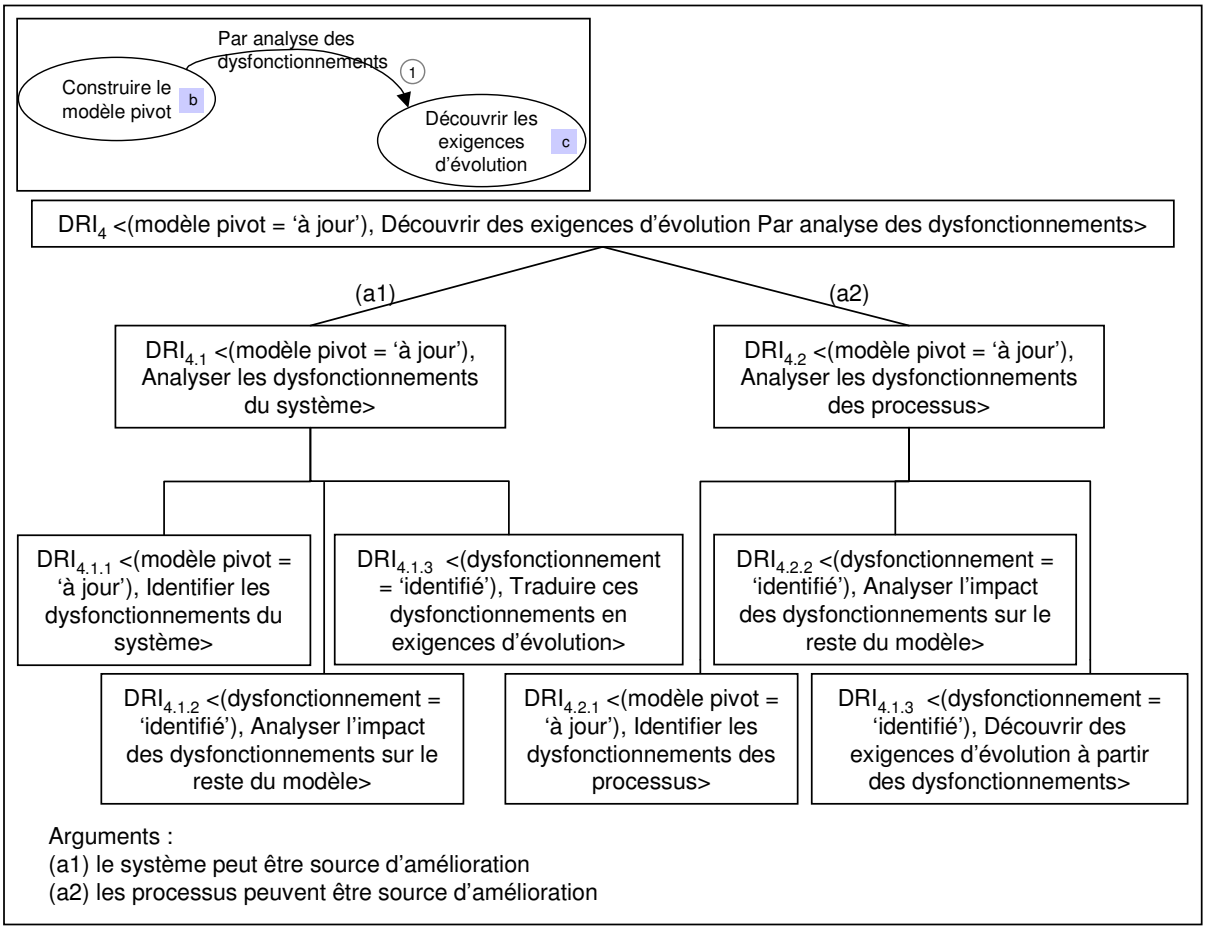


Figure 110 : Directive plan aidant à Découvrir des exigences d'évolution Par analyse des dysfonctionnements de l'existant

Cette section présente les trois directives permettant d'analyser les dysfonctionnements du système.

6.2.1 Identifier les dysfonctionnements du système

La directive $DRI_{4.1.1}$ aide à identifier les dysfonctionnements du système. Elle se présente sous la forme d'une hiérarchie de directives (Figure 111). Cette directive propose différentes alternatives :

1. interviewer les parties prenantes, par exemple les utilisateurs, les concepteurs ou les développeurs chargés de la maintenance ;
2. étudier l'utilisation du système. Cette directive se décompose en deux sous-directives correspondant à l'analyse de la gestion des exceptions et à l'analyse de l'utilisation normale du système ;
3. analyser le modèle pivot. Il s'agit ici de se concentrer uniquement sur la partie du modèle pivot colorée en blanc ou en gris clair. Une autre directive (DRI_6) s'intéresse aux ruptures de l'alignement ;
4. trouver des dysfonctionnements par étude du modèle de système.

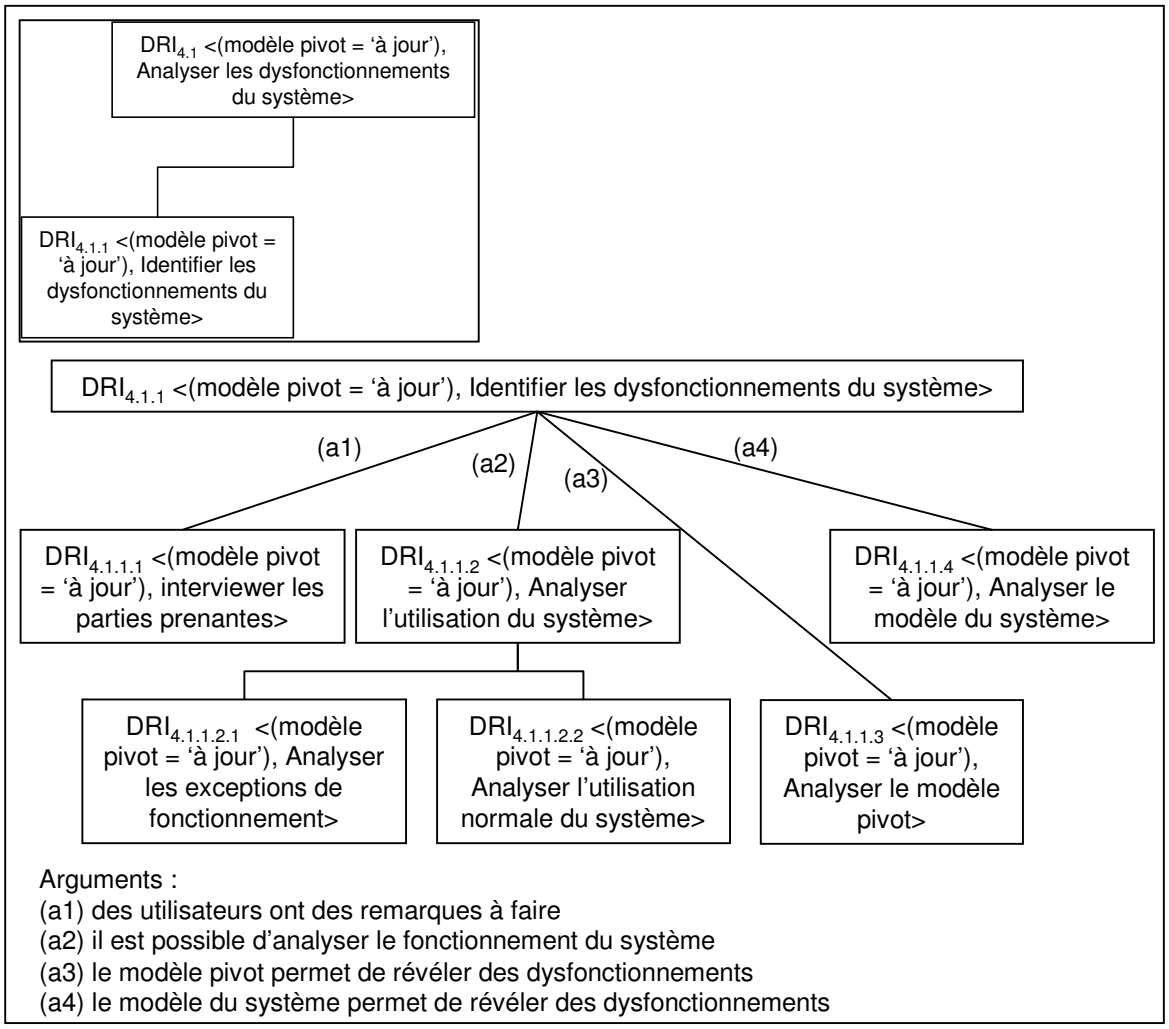


Figure 111 : Hiérarchie de directives aidant à Identifier les dysfonctionnements du système

6.2.2 Analyser l'impact des dysfonctionnements sur le reste du modèle

La directive $DRI_{4.1.2}$ aide à analyser l'impact des dysfonctionnements sur le reste du modèle. Elle se décompose en deux directives :

- analyser l'impact des dysfonctionnements découverts sur le reste du système. En effet, un dysfonctionnement identifié sur une partie du système peut avoir des incidences sur d'autres parties du système.
- analyser l'impact des dysfonctionnements sur les processus. Un dysfonctionnement du système peut se répercuter sur le processus.

La Figure 112 présente la structure de la directive $DRI_{4.1.2}$.

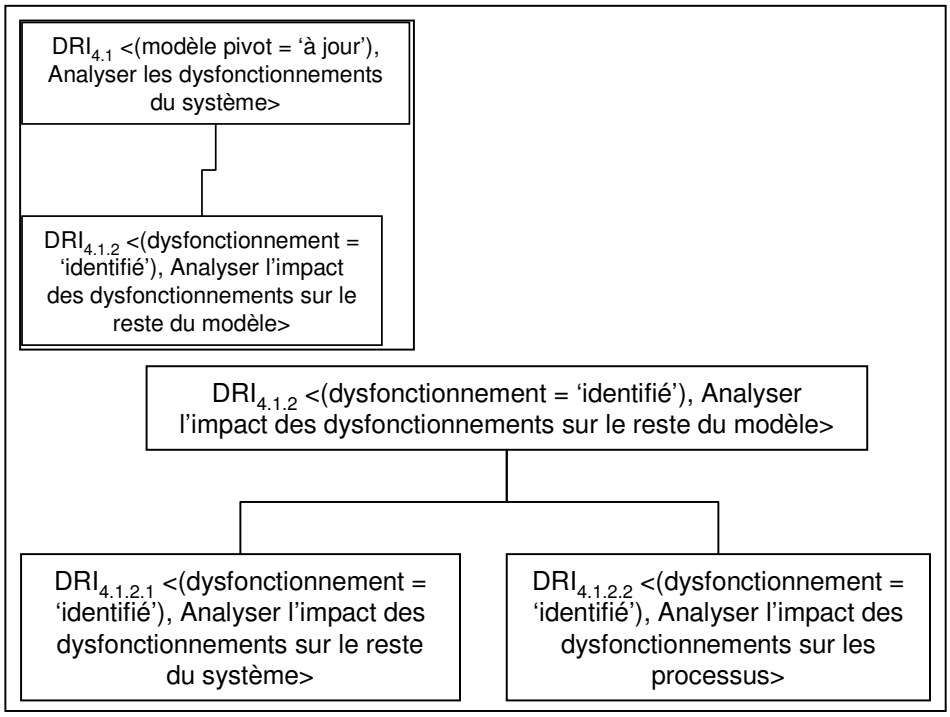


Figure 112 : Directive plan correspondant à l'Analyse de l'impact des dysfonctionnements sur le reste du modèle

6.2.3 Découvrir des exigences d'évolution à partir des dysfonctionnements

La directive <(*dysfonctionnement = 'identifié'*), *Découvrir des exigences d'évolution à partir des dysfonctionnements*> se présente sous la forme d'un plan et se décompose en cinq sous-buts pour découvrir des exigences d'évolution [Etien04].

La Figure 113 présente (i) la directive $DRI_{4.1.3}$ comme un plan et (ii) les critères de choix aidant à choisir une sous-directive plutôt qu'une autre.

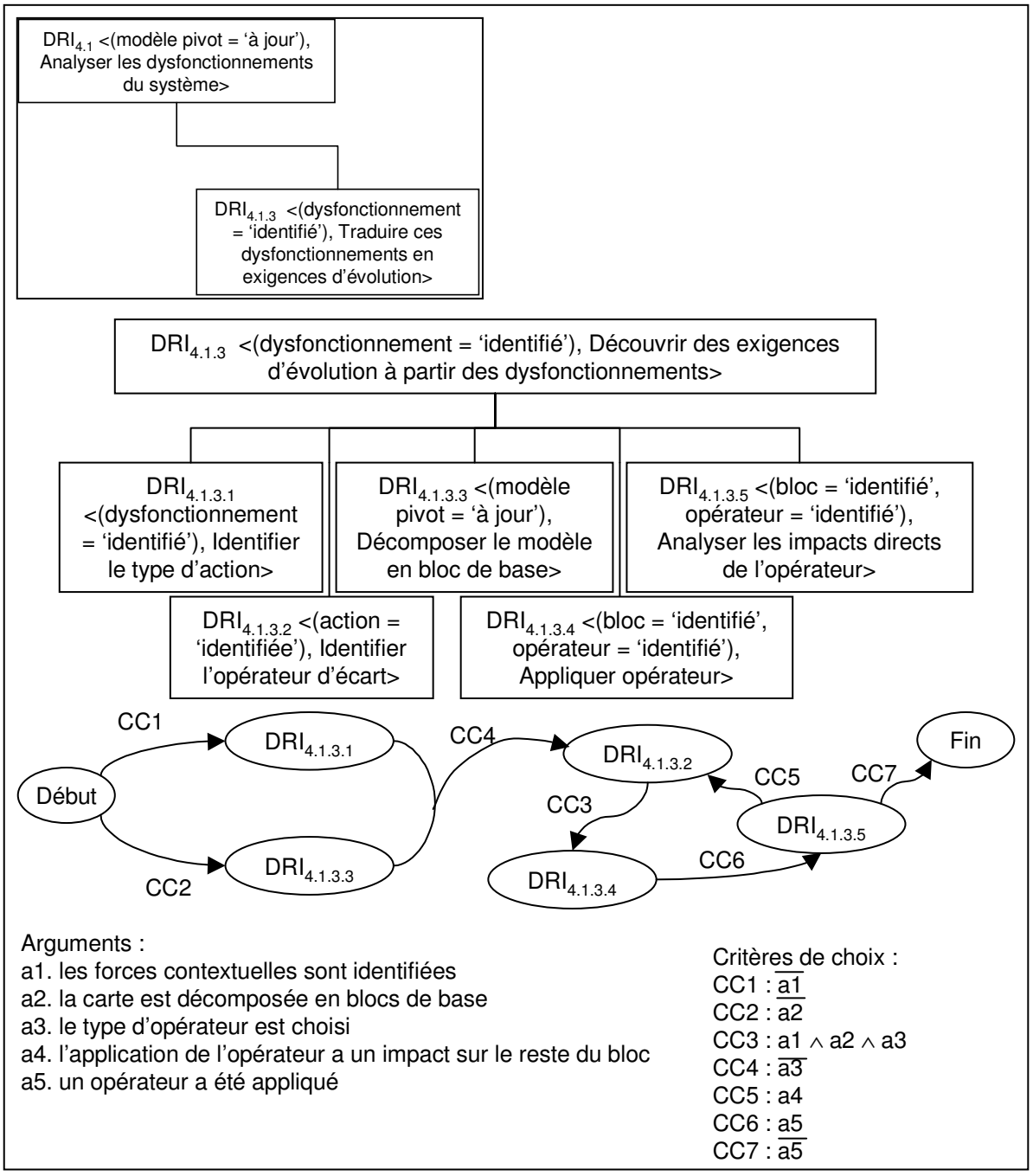


Figure 113 : Directive plan permettant de Découvrir des exigences d'évolution à partir des dysfonctionnements

$DRI_{4.1.3.1}$ <(dysfonctionnement = 'identifié'), Identifier le type d'action>

Cette directive propose d'identifier le type d'action que l'on souhaite entreprendre pour gérer le dysfonctionnement. [Nurcan99] identifie cinq types d'action :

- "maintenir" est utilisé quand il n'est pas nécessaire de modifier le composant existant ;
- "améliorer" permet de procéder à quelques changements sur le composant existant sans le modifier drastiquement ;
- "cesser" implique de supprimer du système un composant qui n'est plus considéré comme bénéfique ;

- “étendre” doit être utilisé si le composant existant est considéré comme toujours pertinent mais que sa portée doit être élargie ;
- “introduire” permet de prendre en compte un besoin complètement nouveau pour l’organisation compte tenu de sa situation actuelle.

Le type d’action est identifié en fonction de la situation courante, mais aussi de la façon dont on souhaite résoudre le dysfonctionnement.

$DRI_{4.1.3.2}$ <(action = ‘identifiée’), Identifier l’opérateur d’écart>

Cette directive associe un opérateur d’écart à l’action identifiée. Afin d’aider l’ingénieur dans l’identification de l’opérateur, la directive propose les différentes correspondances action/opérateur mentionnées dans le Tableau 28.

Type d’action	Opérateurs
Maintenir	N/A
Améliorer	*Renommer, Remplacer, Fusionner, Diviser, Joindre, Enlever, Modifier, ChangerOrigine, Retyper, AjouterComposant, SupprimerComposant, DéplacerComposant*
Cesser	*Supprimer, Fusionner, Enlever, SupprimerComposant*
Etendre	*Diviser, Remplacer, Ajouter, Joindre, AjouterComposant*
Introduire	*Ajouter, Joindre, AjouterComposant*

Tableau 28 : Lien entre les types d’action et les opérateurs

Il s’agit, grâce à cette directive, non seulement d’identifier l’opérateur générique que l’on souhaite utiliser pour remédier au dysfonctionnement, mais aussi le composant du modèle pivot sur lequel on souhaite appliquer cet opérateur. On peut choisir de n’appliquer les opérateurs que sur les parties blanche et gris clair correspondant au système ou au contraire, profiter de l’évolution pour réduire l’alignement donc de ne les appliquer qu’aux éléments provenant du processus et/ou du système.

$DRI_{4.1.3.3}$ <(modèle pivot = ‘à jour’), Décomposer le modèle en blocs de base>

Cette directive correspond à la décomposition du modèle pivot en blocs de base. Un bloc de base est constitué (i) d’une intention et de toutes les stratégies entrant et sortant de cette intention et (ii) ainsi que du réseau de classes correspondant aux différentes sections constituant le bloc. Un bloc de base peut être coloré de trois couleurs de la même façon que le modèle pivot. La Figure 114 présente la directive plan permettant de décomposer le modèle pivot en blocs de base.

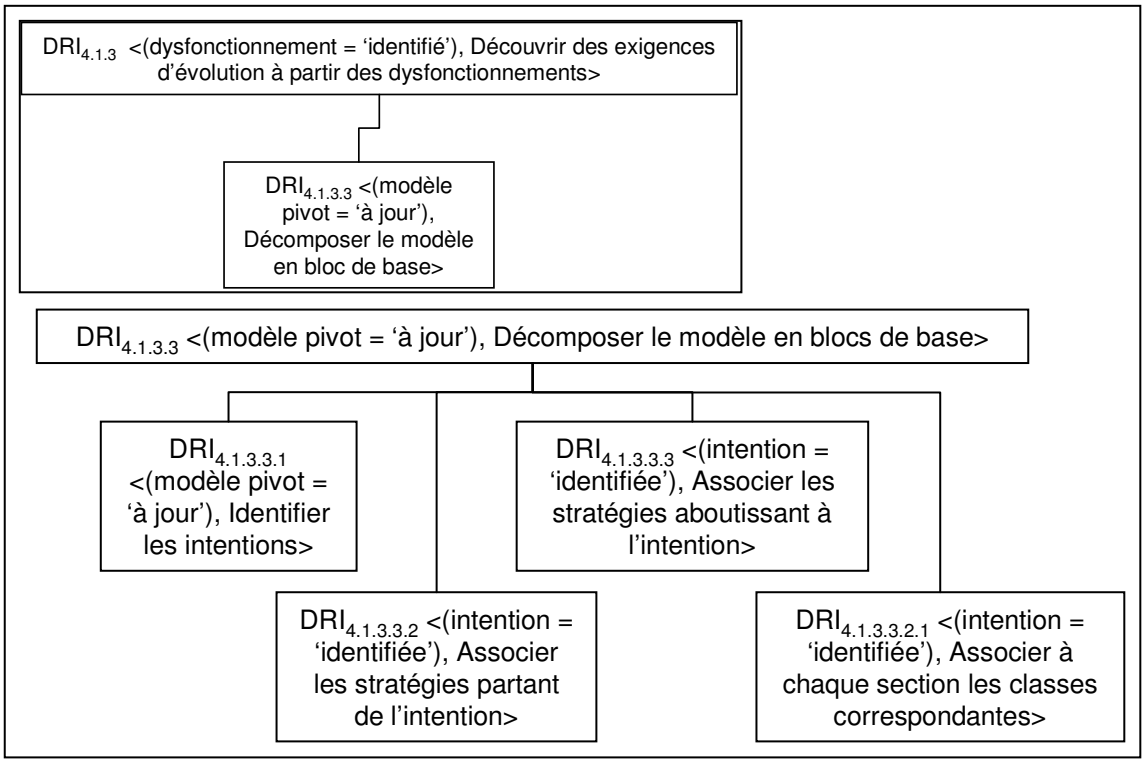


Figure 114 : Directive plan permettant de Décomposer le modèle pivot en blocs de base

La décomposition en blocs de base permet :

1. d'analyser les impacts que l'application d'un opérateur peut avoir sur d'autres parties du bloc.
2. de spécifier l'impact qu'un changement sur une section ou une intention peut avoir sur une classe et vice-versa.

$DRI_{4.1.3.4}$ <(bloc = 'identifié', opérateur = 'identifié'), Appliquer opérateur>

Cette directive correspond à l'application de l'opérateur sur le modèle pivot. Le chapitre 7 présente pour chaque opérateur spécifique associé au méta-modèle pivot, la situation dans laquelle il doit laisser le modèle après avoir été appliqué.

$DRI_{4.1.3.5}$ <(bloc = 'identifié', opérateur = 'identifié'), Analyser les impacts directs de l'opérateur>

Cette directive a pour but d'analyser les impacts directs de l'application d'un opérateur. Ainsi, la modification d'une intention peut avoir des incidences sur les pré et post conditions des sections auxquelles elle appartient, mais aussi sur l'ensemble des classes.

Il est important de souligner qu'un même dysfonctionnement peut nécessiter l'application de plusieurs opérateurs sur différentes parties du modèle. Chacun de ces opérateurs peut correspondre à des types d'action différents.

6.3. Découvrir des exigences d'évolution Par analyse des forces contextuelles

La directive <(*modèle pivot = 'à jour'*), *Découvrir des exigences d'évolution Par analyse des forces contextuelles*> présente une manière alternative de découvrir des exigences d'évolution. Celle-ci repose sur l'analyse des forces contextuelles. La Figure 115 montre la structure de cette directive plan.

Elle se décompose en deux sous-directives permettant de (i) définir des forces contextuelles et (ii) spécifier les exigences d'évolution.

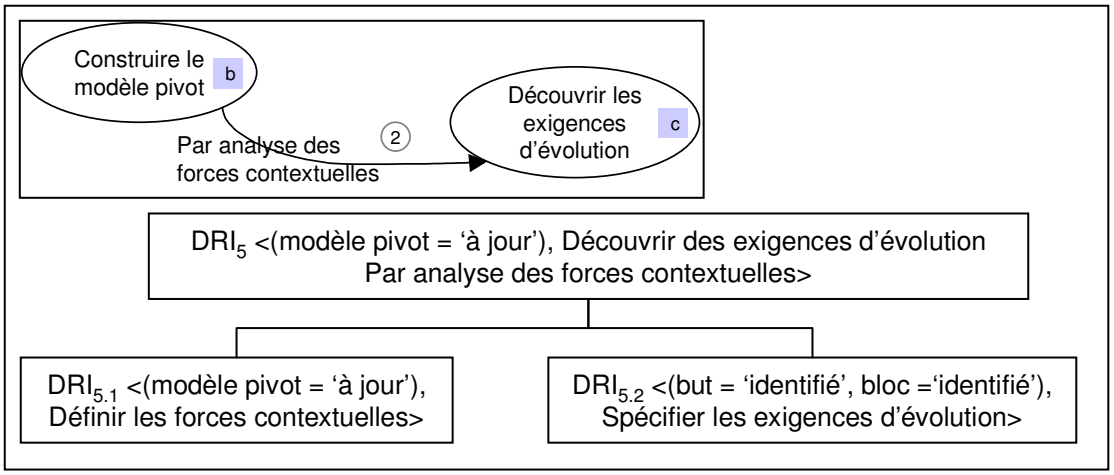


Figure 115 : Directive de réalisation d'intention aidant à Découvrir des exigences d'évolution Par analyse des forces contextuelles

6.3.1 Définir les forces contextuelles

La directive <(*modèle pivot = 'à jour'*), *Définir les forces contextuelles*> aide à définir des forces contextuelles. Pour cela, elle propose :

1. d'identifier des forces contextuelles ;
2. de les exprimer sous forme de buts.

La Figure 116 montre la structure de la directive plan permettant de définir les forces contextuelles.

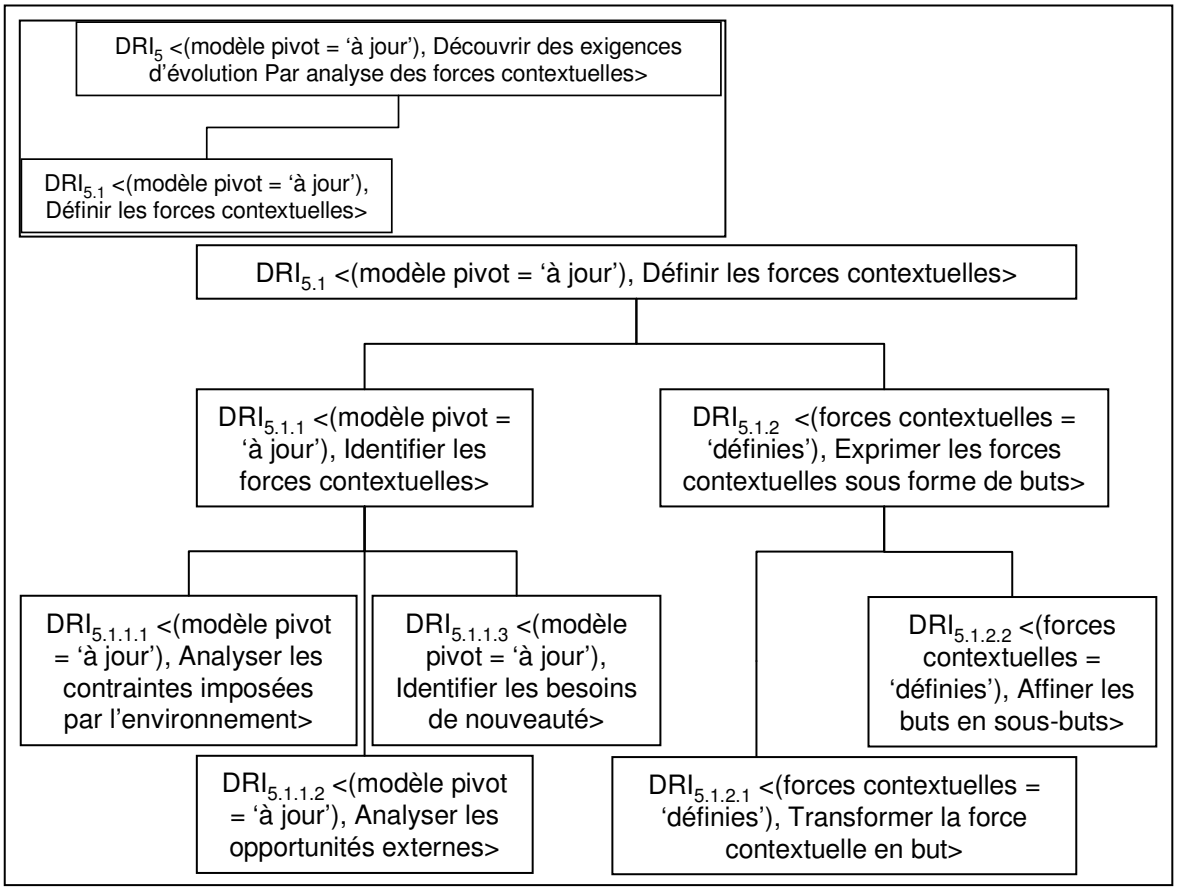


Figure 116 : Directive plan correspondant à la Définition des forces contextuelles

La directive $DRI_{5.1.1}$ a pour but de définir les grandes orientations de l'organisation. C'est par exemple au cours de la réalisation de cette directive qu'il sera décidé de changer la politique marketing pour une politique "orientée-client".

La réalisation de cette directive relève du management stratégique. On peut néanmoins préciser qu'elle se décompose en trois sous-directives correspondant à l'analyse SWOT. En d'autres termes, il s'agit : (i) d'analyser les contraintes imposées par l'environnement ; (ii) d'analyser les opportunités externes et (iii) d'identifier les besoins de nouveauté.

La directive $DRI_{5.1.2}$ permet de traduire les forces contextuelles sous forme de buts éventuellement décomposés en sous-buts.

6.3.2 Spécifier les exigences d'évolution

La directive $DRI_{5.2}$ <(*but = 'identifié', bloc ='identifié'*), *Spécifier les exigences d'évolution*> est analogue à la directive $DRI_{4.1.3.3}$. Elle prend en compte non pas les dysfonctionnements, mais les sous-buts identifiés précédemment.

6.4. Découvrir des exigences d'évolution Par analyse de l'alignement

Nous avons vu au chapitre 5 que l'approche ACEM propose à la fois de faire co-évoluer le système et les processus afin de prendre en compte de nouveaux besoins mais aussi de corriger un mauvais alignement. Les deux sections précédentes ont présenté deux façons différentes de prendre en compte de nouveaux besoins. La directive <(*modèle pivot = 'à jour'*), *Découvrir les exigences d'évolution Par analyse de l'alignement*> aide à découvrir des exigences d'évolution pour améliorer l'alignement.

Cette directive se présente sous la forme d'une hiérarchie de directives (Figure 117). Elle se décompose en deux sous-directives correspondant : (i) à l'identification d'une rupture d'alignement et (ii) au choix d'une procédure de rétablissement de l'alignement.

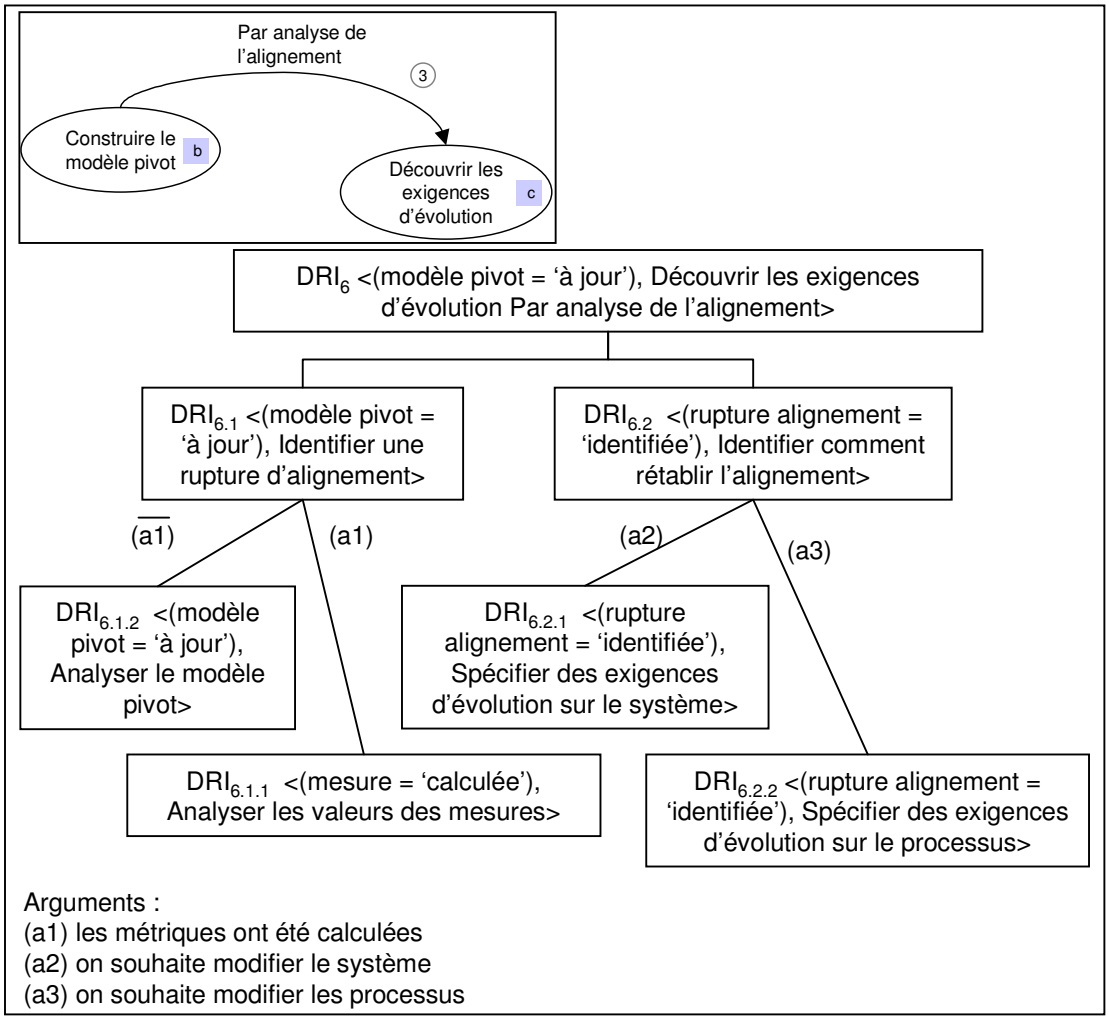


Figure 117 : Hiérarchie de directives proposant de Découvrir des exigences d'évolution par analyse de l'alignement

6.4.1 Identifier une rupture de l'alignement

La directive $DRI_{6.1}$ propose deux approches différentes permettant d'identifier une rupture d'alignement.

6.4.1.1 Analyser les valeurs des mesures

La directive <(*mesure = 'calculée'*), *Analyser les valeurs des mesures*> propose d'utiliser les métriques d'alignement définies au chapitre 3 pour identifier des ruptures de l'alignement. Les mesures d'alignement peuvent être utilisées de deux façons :

1. elles peuvent servir à identifier une rupture d'alignement. Dans ce cas, les ruptures d'alignement décelées doivent être traduites dans les termes du modèle pivot.

2. associées à une analyse du modèle pivot, elles peuvent aider à choisir les corrections à entreprendre. En effet, comme nous l'avons vu au chapitre 3, l'introduction des notions de seuil et de poids dans le calcul des mesures peut donner des éclairages différents de l'alignement et aider à établir des priorités.

L'utilisation des métriques pour identifier des ruptures de l'alignement est optionnelle mais recommandée.

6.4.1.2 Analyser le modèle pivot

La directive <(*modèle pivot = 'à jour'*), *analyser le modèle pivot*> constitue une alternative à la directive précédente. Lors de sa construction, le modèle pivot a été coloré mettant ainsi en évidence trois ensembles :

- les parties propres au système ;
- les parties propres aux processus ;
- les parties communes au système et aux processus.

L'identification d'une rupture d'alignement par analyse du modèle pivot s'appuie sur l'étude et la comparaison des deux premiers ensembles. En effet, si le système et les processus étaient parfaitement alignés, le modèle pivot serait totalement blanc. L'existence de parties grises signifie que certains aspects des processus (respectivement du système) ne sont pas pris en compte ou sont mal gérés par le système (respectivement les processus).

6.4.2 Identifier comment rétablir l'alignement

La directive <(*rupture alignement = 'identifiée'*), *identifier comment rétablir l'alignement*> propose deux approches pour identifier comment rétablir l'alignement : (i) mettre en œuvre des actions correctives sur le système ou (ii) mettre en œuvre des actions correctives sur les processus. La première proposition modifie les parties du modèle pivot colorées en gris clair. La deuxième proposition porte sur les parties gris foncé.

Dans chacun des cas, il s'agit de spécifier des exigences d'évolution. Les directives $DRI_{6.2.1}$ et $DRI_{6.2.2}$ ont une définition comparable à la directive $DRI_{4.1.3}$ décrite à la section 6.2.3. Au lieu de prendre en considération des dysfonctionnements, ces directives spécifient des exigences d'évolution à partir des observations de rupture d'alignement.

6.5. Découvrir les exigences d'évolution Directement

La directive <(*modèle pivot = 'à jour'*), *Découvrir les exigences d'évolution Directement*> propose de découvrir les exigences d'évolution sans construire au préalable le modèle pivot. Cette directive est appliquée quand le modèle pivot existe et est à jour. Elle propose de

découvrir les exigences d'évolution de trois façons différentes : par analyse des dysfonctionnements ($DRI_{9.1}$), par analyse des forces contextuelles ($DRI_{9.2}$) ou par analyse de l'alignement ($DRI_{9.3}$). La Figure 118 présente cette directive choix.

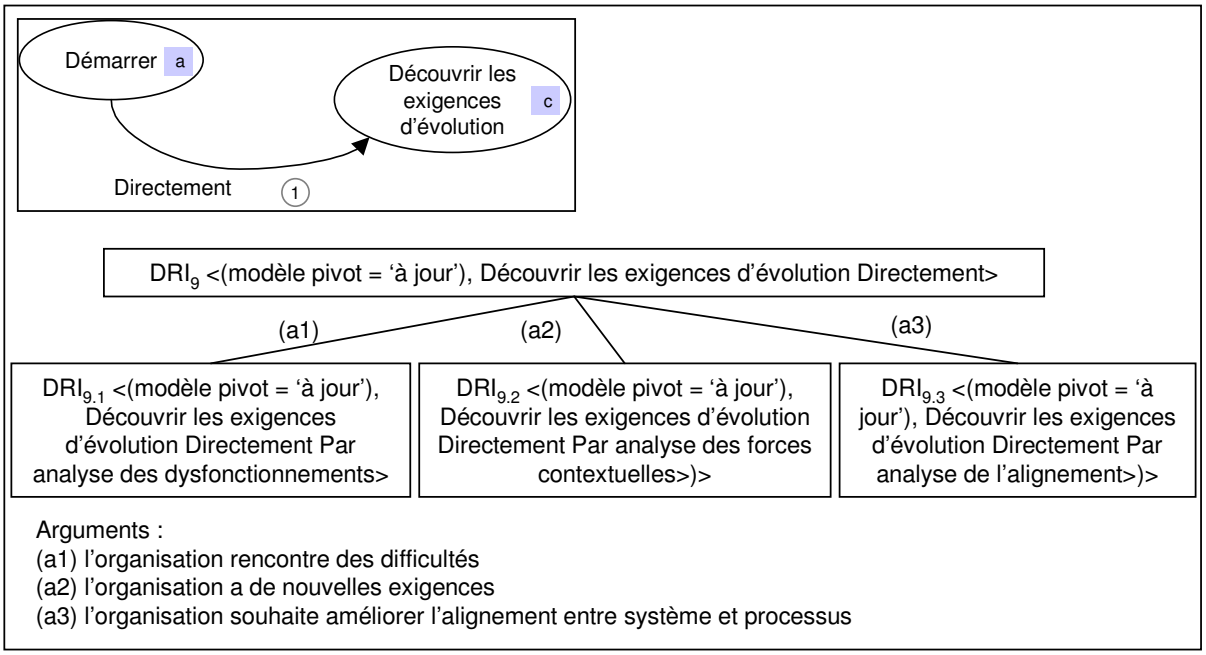


Figure 118 : Directive de réalisation d'intention permettant de Découvrir les exigences d'évolution Directement

Chacune des trois sous-directives est semblable respectivement aux directives DRI_4, DRI_5 et DRI_6.

6.6. Progresser depuis Découvrir des exigences d'évolution

La directive <(*exigences = 'exécutées'*), *Progresser depuis Découvrir les exigences d'évolution*> présente différentes alternatives pour progresser dans la démarche ACEM lorsque des exigences d'évolution ont été découvertes. Il est ainsi possible d'arrêter le processus en propageant les écarts identifiés sur les modèles de système et de processus. Une autre alternative réside dans la découverte d'autres exigences d'évolution à partir des premières. Chacune de ces deux approches correspond à une intention différente. La directive de sélection d'intention suivante aide à progresser depuis *Découvrir des exigences d'évolution* (Figure 119).

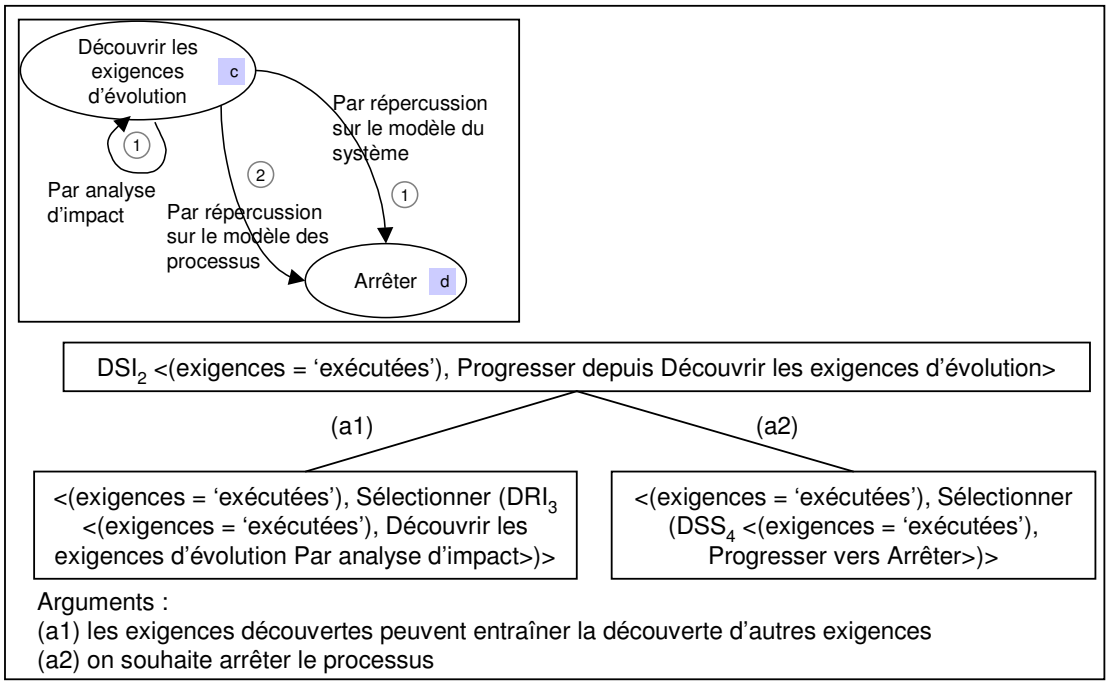


Figure 119 : Directive de sélection d'intention permettant de progresser depuis Découvrir des exigences d'évolution

6.7. Découvrir des exigences d'évolution Par analyse d'impact

La directive <(*exigences = 'exécutées'*), *Découvrir les exigences d'évolution Par analyse d'impact*> repose sur le principe que les exigences d'évolution appliquées sur certaines parties du modèle pivot peuvent avoir un impact sur d'autres parties du modèle. La directive $DRI_{4.1.3}$ décrite à la section 6.2.3 propose de gérer l'impact direct des exigences d'évolution dès leur mise en œuvre. Un impact direct est un impact qui a lieu au sein d'un même bloc de base. Or, certaines exigences peuvent avoir un impact indirect, c'est-à-dire sur des parties du modèle se trouvant en dehors du bloc de base.

La directive DRI_8 aide à découvrir des exigences d'évolution par analyse d'impact. La Figure 120 présente la structure de cette directive.

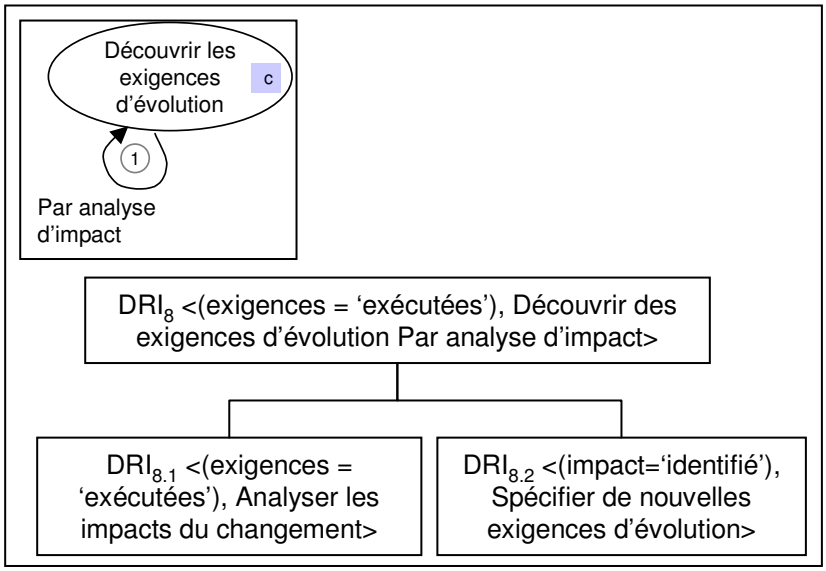


Figure 120 : Directive de réalisation d'intention permettant de Découvrir des exigences d'évolution par analyse d'impact

La directive DRI_8 se décompose en deux directives correspondant à :

- l'analyse des impacts d'une exigence d'évolution. Les modifications ayant lieu sur une section affinée ont des répercutions sur la carte de niveau i+1 qui l'affine, mais peut-être aussi sur d'autres cartes du niveau i+1. En effet, plus le niveau de la carte est faible, plus son niveau d'abstraction est élevé et plus elle représente une large partie du système ou des processus. Les exigences d'évolution sur ces cartes là ont des incidences multiples. Des exigences d'évolution peuvent également avoir des impacts au sein de la même carte en particulier quand les intentions ou les sections partagent les mêmes parties de produit.
- la spécification de nouvelles exigences d'évolution. Cette directive est similaire à la directive $DRI_{4.1.3}$, mais prend en compte les impacts et non les dysfonctionnements.

7. Propager les changements sur les modèles du système et des processus

L'organisation utilise quotidiennement ses propres modèles et non le modèle pivot. Il est donc important de propager les changements sur les modèles du système et des processus afin que les modèles utilisés par l'organisation représentent la situation To-Be.

7.1. Progresser vers Arrêter

La directive <(*exigences = 'exécutées'*), *Progresser vers Arrêter*> propose deux alternatives pour arrêter le processus : répercuter les exigences d'évolution (i) sur le modèle de système et (ii) sur le modèle de processus. La Figure 121 présente la structure de la directive de sélection de stratégie permettant de progresser vers *Arrêter*.

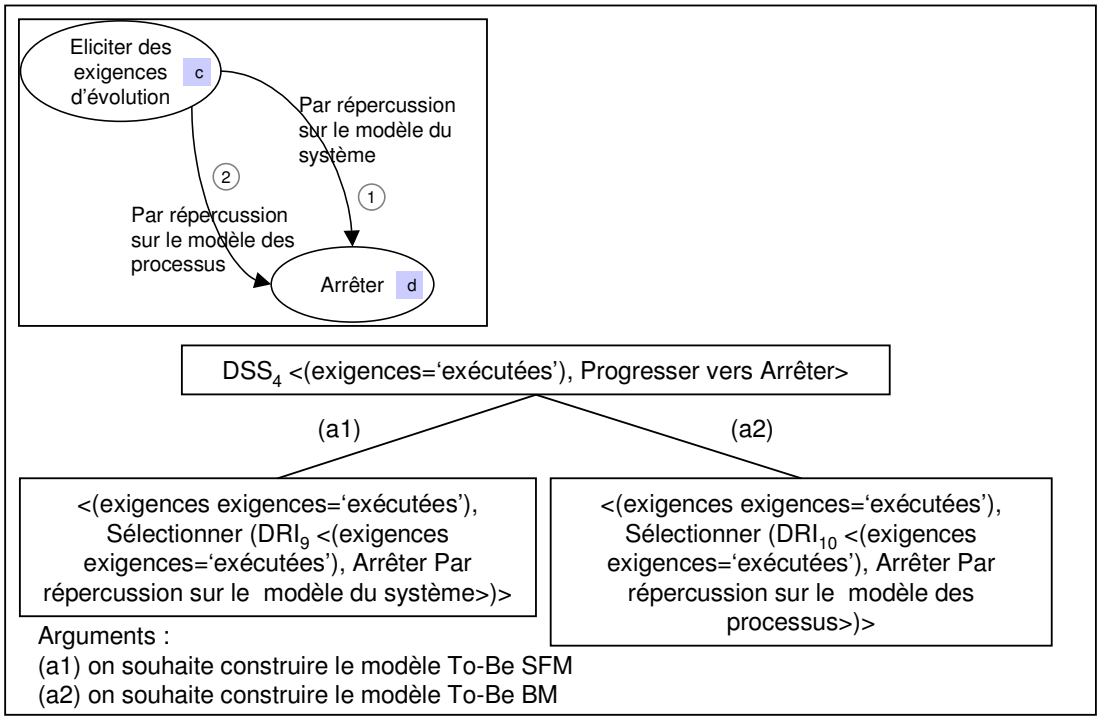


Figure 121 : Directive de sélection de stratégie permettant de progresser vers Arrêter

7.2. Répercuter les exigences d'évolution sur le modèle de système et le modèle de processus

La répercussion des exigences d'évolution sur l'un ou l'autre des modèles se fait différemment suivant le méta-modèle utilisé pour représenter le système et les processus. Il est donc difficile, dans l'état actuel de nos recherches, de décrire une directive formelle pour aider l'ingénieur.

Il est néanmoins important de noter que :

- la répercussion de chaque exigence d'évolution est indispensable pour obtenir des modèles To-Be du système et des processus qui prennent en compte tous les changements identifiés ;
- la répercussion sur le modèle de système s'appuie sur les exigences d'évolution mises en œuvre sur les parties en gris clair et en blanc du modèle pivot ;
- la répercussion sur le modèle de processus s'appuie sur les exigences d'évolution mises en œuvre sur les parties en gris foncé et en blanc du modèle pivot ;
- la répercussion doit commencer par les cartes de plus bas niveau d'abstraction. Les cartes des plus hauts niveaux permettent d'assurer la cohésion entre les différentes parties du modèle de système ou du modèle de processus.

Procéder par répercussion permet ainsi de tenir compte de tout le bénéfice de l'utilisation des exigences d'évolution sur le modèle pivot. En effet, répercuter les changements évite une longue analyse de rétro-ingénierie puisque seules les parties du modèle pivot affectées par les changements nécessitent d'être prises en compte.

A la fin du processus de la méthode ACEM, les modèles To-Be de système et de processus ne sont pas forcément alignés. Tout dépend du but que l'on cherchait à atteindre en exécutant ce processus (co-évolution ou rétablissement, partiel ou total, de l'alignement). En effet, nous avons vu que la prise en compte de nouveaux besoins pouvait être l'occasion de rétablir partiellement l'alignement. Il est également possible que les dirigeants ne souhaitent pas forcément rétablir l'alignement à 100%, par exemple en cas de non retour sur investissement prévisible.

8. Conclusion

Ce chapitre a présenté la méthode ACEM permettant de satisfaire des besoins d'évolution et d'améliorer l'alignement entre le système et les processus. Le modèle de processus de notre approche, présenté sous forme de carte, permet de guider l'évolution du modèle pivot, de proposer plusieurs alternatives et d'en choisir une. Cette évolution repose sur la spécification d'exigences d'évolution matérialisées par des opérateurs d'écarts.

L'utilisation du méta-modèle de Carte comme méta-modèle de processus pour la modélisation de la démarche permet une approche structurée et guidée. Le concept de stratégie permet d'expliciter des alternatives différentes et montre l'aspect multi-démarche du méta-modèle de Carte. Ceci permet d'offrir une certaine liberté de choix aux ingénieurs qui exécutent le processus. Les trois types de directives (DRI, DSS, DSI) associées à la carte facilitent l'exécution de tâches, d'une part et la progression au sein du processus, d'autre part.

L'utilisation du méta-modèle pivot permet de développer une approche intentionnelle et d'abstraire les détails techniques pour se concentrer sur l'essentiel. Elle permet également de modéliser dans les mêmes termes le système et les processus.

Le principe de coloration permet de représenter au sein d'un même modèle, un système et des processus qui ne sont pas parfaitement alignés. Il permet également d'identifier visuellement les ruptures d'alignement entre système et processus.

Les opérateurs présentés au chapitre 7 ont été utilisés pour construire le modèle pivot à partir des modèles As-Is, mais aussi pour le faire évoluer.

Le chapitre 9 illustre la démarche en présentant une étude de cas pour la gestion de réservations de chambres d'hôtels où la méthode ACEM est appliquée.

CHAPITRE 9 : APPLICATION DE LA METHODE ACEM

1. Introduction

Ce chapitre poursuit de manière détaillée l'étude de cas introduite au chapitre 4. Nous proposons d'appliquer la méthode ACEM pour faire évoluer et corriger l'alignement entre le système d'information et les processus d'entreprise. En effet, les dirigeants du groupement d'hôtels ont pris conscience des difficultés rencontrées au quotidien par les utilisateurs du système. Ils souhaitent corriger l'alignement entre le système d'information et les processus d'entreprise et prendre en compte de nouvelles exigences pour rester compétitifs face aux autres stations européennes. Le budget étant limité, les dirigeants sont bien conscients qu'il ne sera pas possible de prendre en compte toutes leurs exigences et d'aboutir à un alignement parfait.

Ce chapitre est organisé comme suit : la section 2 rappelle brièvement la situation courante et présente les différentes exigences des dirigeants ; la section 3 présente une vue globale de la mise en œuvre de la méthode ACEM. Les sections 4 à 6 décrivent en détail les différentes étapes de cette mise en œuvre. Enfin, les apports et les conclusions sont présentés en section 7.

2. Introduction du cas et description des axes d'évolution

Cette section présente dans les grandes lignes la situation de départ de l'étude de cas et introduit les axes d'évolution que les dirigeants souhaitent entreprendre.

2.1. Situation de départ

L'étude de cas est celle d'un groupement d'hôtels. Il s'agit de l'exemple présenté au chapitre 4. Cette section résume ce qui a été présenté au chapitre 4 et à l'Annexe 1.

2.1.1 Principe fondateur : le partenariat hôtelier

Il y a plusieurs années, des hôteliers de la région alpine ont décidé de devenir partenaires et de centraliser la gestion de leurs réservations afin de lutter contre la concurrence des grandes chaînes hôtelières multi-nationales. Ce projet novateur a abouti à la mise en place de processus et d'un système d'information implémentant ce partenariat.

Dans la situation actuelle, le système permet de gérer des réservations de chambres dans différents hôtels des stations de ski françaises. Il est par ailleurs possible de modifier les grilles de tarifs ainsi que la liste des ressources hôtelières mises à la disposition des clients.

Les réservations peuvent être faites par téléphone, sur place ou par Internet. Dans chaque cas, divers renseignements sont demandés aux clients afin de constituer leurs demandes. S'il n'existe pas de disponibilités satisfaisant une demande, celle-ci peut être mise en attente ou annulée. Dans le cas où la demande est satisfaite, une réservation est créée.

Les annulations de réservation sont acceptées sans contrepartie financière si elles parviennent au système de réservation au moins une semaine avant le début de la réservation. Dans le cas contraire, le client doit payer 75% du prix.

2.1.2 Dysfonctionnements

L'évaluation du degré d'alignement entre les processus d'entreprise et le système supportant le partenariat a permis de mettre en évidence un certain nombre de dysfonctionnements :

- Les activités concernant la gestion des ressources (stations, hôtels, chambres) ne sont pas prises en charge par le système ;
- D'autres activités sont certes gérées par le système, mais pas de façon satisfaisante dans la mesure où l'information utilisée par les processus et par le système est différente ;
- L'information est globalement gérée de façons différentes dans le système et dans les processus. Les définitions des données de part et d'autre ne se correspondent pas ;
- Une faible proportion de buts et chemins des processus organisationnels est mise en œuvre par le système.

L'ensemble de ces problèmes a des conséquences sur la performance de l'organisation mise en place au travers du partenariat. De nombreuses activités doivent être réalisées manuellement alors que le système est sensé les faciliter. Les saisies redondantes sont fréquentes et certaines données sont incohérentes. Le diagnostique de ces dysfonctionnements montre que de nombreuses évolutions sont requises pour amener le système à un état de fonctionnement satisfaisant pour les utilisateurs, pour améliorer la performance de l'organisation et, finalement, pour améliorer la qualité des services fournis aux clients des hôtels.

2.2. Axes d'évolution

Afin d'être davantage compétitifs, les dirigeants des hôtels souhaitent :

- mettre en place une politique orientée-client et non plus orientée-produit afin de mieux satisfaire les clients et ainsi les fidéliser ;
- diversifier leur offre en proposant des produits plus évolués intégrant des activités telles que théâtre, cinéma, thalassothérapie, clubs enfants, aquagym, massage, cours de ski à différents niveaux, sport extrême, clubbing, soirées spéciales…

Il est par ailleurs envisagé dans le cas où cette évolution serait un succès d'étendre l'utilisation du système à d'autres régions de France ou d'Europe.

3. Vue d'ensemble de la mise en œuvre de la méthode ACEM

Les hôteliers ont décidé de mettre en œuvre la méthode ACEM pour monitorer l'évolution du système de réservation hôtelière. La Figure 122 présente en gras les parties de la carte décrivant la méthode ACEM qui ont été appliquées dans l'exemple. La figure montre que les deux principaux buts de la méthode ont été mis en œuvre : construire un modèle pivot, puis découvrir des exigences d'évolution.

Pour construire le modèle pivot, la stratégie employée a été *En s'appuyant sur les modèles As-Is*. Aucun modèle pivot n'étant disponible, une stratégie de mise à jour n'était pas envisageable. Comme le montre la figure, toutes les autres sections de la carte ont été utilisées au moins une fois.

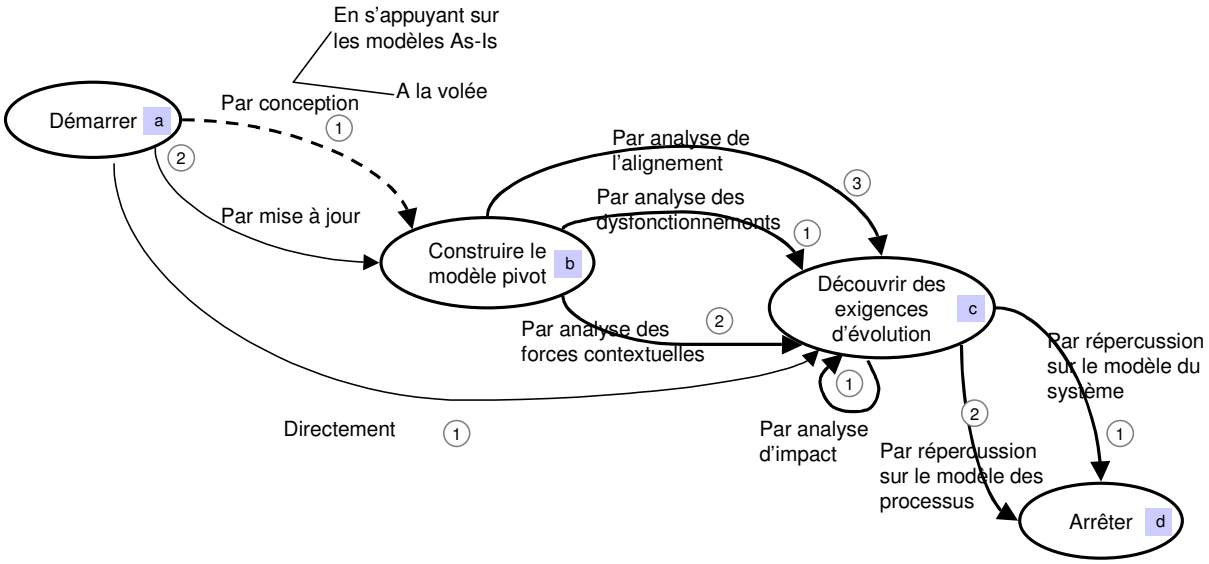


Figure 122 : Carte de haut niveau utilisée dans le projet d'évolution

La carte a permis d'identifier plusieurs directives permettant de découvrir les exigences d'évolution à partir du modèle pivot. Le choix et l'exploitation de ces directives ont permis de découvrir des exigences d'évolution par analyse des dysfonctionnements de l'existant, par analyse des forces contextuelles et par analyse de la relation d'alignement. Comme le propose la méthode, les divers impacts provoqués par certaines de ces exigences d'évolution ont été analysés, permettant ainsi de définir de nouvelles exigences d'évolution.

La démarche se termine en répercutant sur le modèle pivot les évolutions requises. Comme le montre la Figure 122, la répercussion des évolutions s'est faite en exécutant les stratégies *Par répercussion sur le modèle de processus* et *Par répercussion sur le modèle de système*.

La section suivante montre de manière détaillée l'application de la méthode ACEM pour construire les cartes du système et des processus d'entreprise afin d'obtenir le modèle pivot.

4. Construire le modèle pivot

Plusieurs directives sont proposées par la méthode ACEM pour construire le modèle pivot. Toutes n'ont pas été choisies dans le projet, en particulier celles reposant sur des stratégies de

mise à jour car aucune carte de l'exemple n'était disponible au départ. La Figure 123 ci-dessous présente en gras les directives choisies pour construire le modèle pivot *En s'appuyant sur les modèles As-Is*.

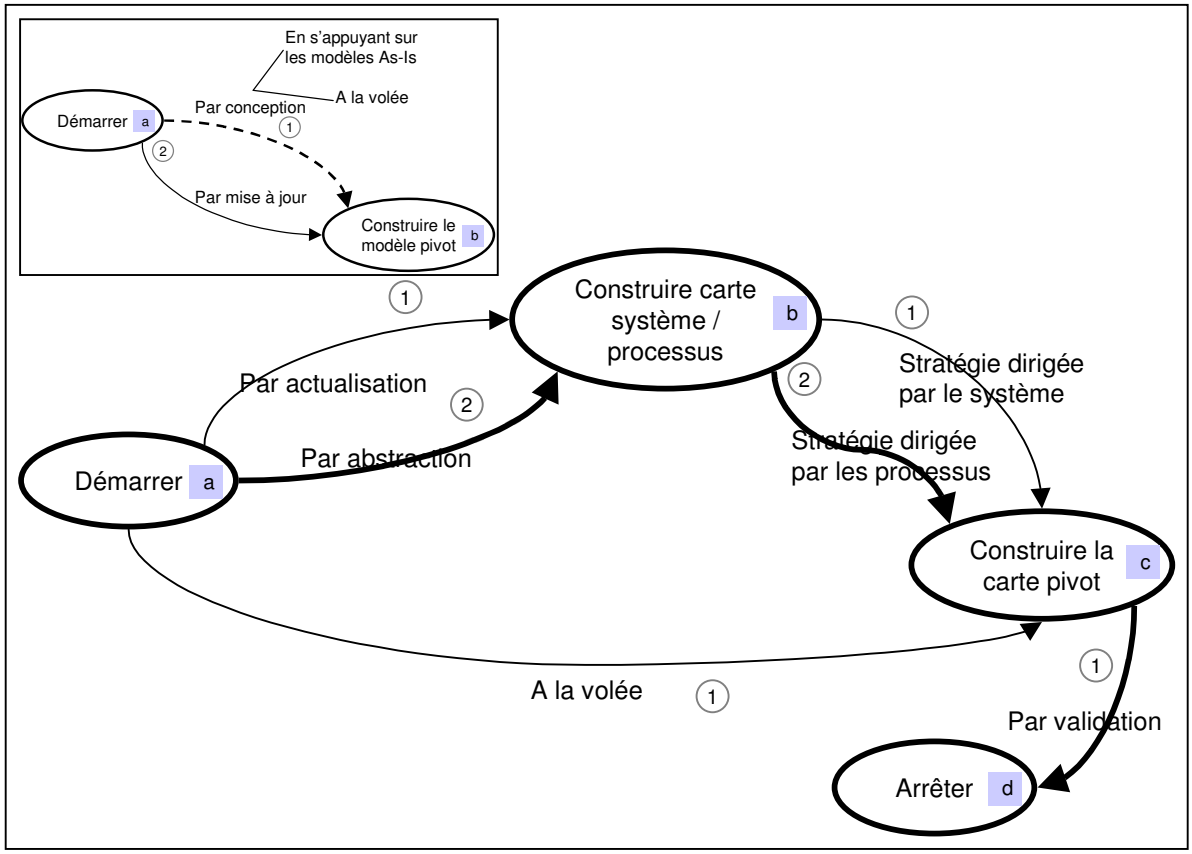


Figure 123 : Partie de la carte $C.C_{ab1}$ exécutée pour construire le modèle pivot

La carte système et la carte processus ont été construites de façon analogue en commençant par mettre en évidence les objectifs du système et des processus d'entreprise, comme le propose la section $C.C_{ab1}$.ab2 (*Par abstraction*).

Selon la méthode ACEM, la carte pivot est obtenue par fusion des cartes système et des cartes processus. Cette fusion relève de la réalisation de la section $C.C_{ab1}$.bc2 (*Stratégie dirigée par les processus*). Le processus s'est terminé en validant la carte pivot.

Le déroulement de chacune de ces étapes est détaillé dans les sous-sections ci-après de la manière suivante : la sous-section 4.1 illustre la construction des cartes système, la sous-section 4.2 illustre la construction des cartes processus et la sous-section 4.3 illustre la construction des cartes pivot.

4.1. Construire les cartes système

Cette section a pour but de décrire le système en termes intentionnels sous forme de carte et de préciser les objets système utilisés. La description intentionnelle du système ainsi produite est représentée par une carte de haut niveau, nommée Cs. La carte Cs ainsi que deux cartes plus détaillées Cs_{ab2} et Cs_{bc1} sont présentées dans les sections 4.1.2 et 4.1.3.

4.1.1 Carte Cs de plus haut niveau

Comme indiqué précédemment, la responsabilité du système s'étend de la conception d'un catalogue à la consommation du produit par le client. Les deux objectifs principaux assignés au système sont donc : « offrir un produit » et « gérer les réservations hôtelières ». L'application de la directive d'abstraction des objectifs pour construire la carte système amène à décrire la carte Cs présentée en Figure 124. Comme le montre la figure, plusieurs stratégies ont été identifiées pour la réalisation de chacun de ces objectifs, ainsi que pour terminer la gestion de la réservation de chambres d'hôtel.

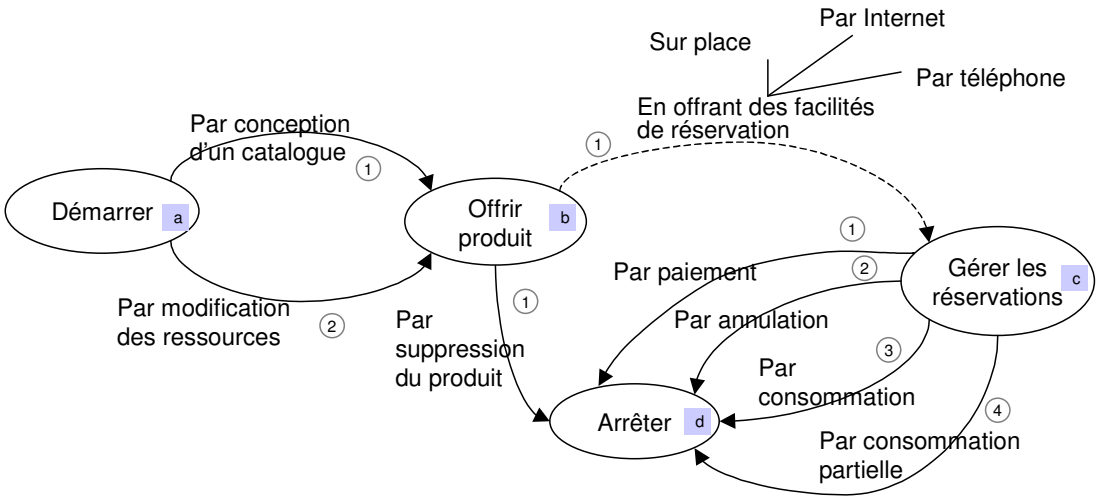


Figure 124 : la carte système Cs de plus haut niveau

Les 4 intentions et 10 stratégies de la carte Cs sont détaillées ci-dessous. Les sections ab1, ab2, bd1, cd2 et bc1 sont affinées, les autres sections sont opérationnalisées directement par des règles de gestion.

Intentions de la carte Cs

Les intentions de la carte Cs sont *Démarrer*, *Offrir produit*, *Gérer les réservations* et *Arrêter* :

- *Démarrer* : indique le point de départ de l'utilisation du système.
- *Offrir produit* : correspond à la mise à disposition des clients d'un ensemble de produits. Le catalogue de produits ainsi constitué est régulièrement mis à jour pour tenir compte des éventuelles fermetures ou créations d'hôtels et de chambres.
- *Gérer les réservations* : fait référence à la réception des demandes clients et à la création des réservations.
- *Arrêter* : le processus de gestion de réservations hôtelières peut se terminer par la suppression d'un produit du catalogue (lorsque la chambre concernée n'est plus offerte), une fois la réservation payée et consommée, ou par annulation de la réservation.

Stratégies de la carte Cs

Deux stratégies sont mises en œuvre dans le système pour *Offrir produit* :

- *Par conception d'un catalogue* : cette stratégie est mise en œuvre dans le cadre de la l'ouverture de nouveaux hôtels.
- *Par modification des ressources* : cette stratégie permet de modifier les caractéristiques d'un hôtel en lui associant de nouvelles chambres ou en en fermant d'autres.

Trois stratégies permettent de réaliser, avec le système, l'intention *Gérer les réservations*. Ces trois stratégies sont regroupées dans le paquet *En offrant des facilités de réservation*. Le regroupement des trois stratégies dans un paquet permet d'indiquer qu'elles sont mutuellement exclusives.

- *Sur place* : cette stratégie correspond à la prise en charge de la création d'une réservation lorsque celle-ci est effectuée par le client sur place, c'est-à-dire directement dans l'hôtel souhaité.
- *Par téléphone* : cette stratégie est mise en œuvre lorsque le client appelle la réservation de l'hôtel auprès duquel il souhaite faire une réservation.
- *Par Internet* : contrairement aux deux stratégies précédentes, le choix de l'hôtel n'est pas établi préalablement, et le client est susceptible de réserver une chambre dans n'importe quel hôtel participant au partenariat.

Dans ces trois stratégies, le système permet de saisir la demande du client et de la mettre éventuellement en attente si aucune disponibilité ne correspond à sa demande. Une réservation peut également être créée si les disponibilités le permettent.

Cinq stratégies sont proposées par le système pour *Arrêter* :

- *Par suppression du produit* : cette stratégie correspond à la suppression d'un hôtel du catalogue ainsi qu'aux différents traitements associés. En effet, si un produit est supprimé alors que des réservations à venir lui sont associées, il peut être nécessaire d'annuler ces réservations et d'avertir les clients concernés.
- *Par paiement* : cette stratégie correspond au paiement par le client de sa réservation ou de ses pénalités. Lorsque le client accède au système par Internet, ceci est fait via un site sécurisé.
- *Par annulation* : de nombreux traitements peuvent découler de l'annulation d'une réservation (en particulier la gestion des disponibilités puisqu'une chambre devient à nouveau disponible). Cette stratégie identifie la saisie de l'annulation et recouvre les traitements qui en découlent.
- *Par consommation* : cette stratégie permet de confirmer dans le système que le client s'est présenté à l'hôtel et est parti le jour prévu par la réservation.
- *Par consommation partielle* : lorsque le client quitte sa chambre avant la date prévue par sa réservation, le système calcule une pénalité qu'il lui inflige. Le système met à disposition les disponibilités laissées par la ou les chambre(s) libérée(s) prématurément, afin éventuellement de satisfaire de nouvelles demandes.

Sections de la carte Cs

Chacune des sections décrites ci-dessus peut être documentée par des pré-conditions, des post-conditions, un acteur, des ressources et des règles des gestions. Le Tableau 29 présente

cette documentation pour les huit sections de la carte Cs. A noter que les règles de gestion ne sont pas précisées pour les sections affinées.

Code	Pré-condition	Post-condition	Acteur	Ressource	Règle de gestion
ab1	rien	Hôtel, DisponibilitéHôtel	Hôtelier	-	
ab2	Hôtel	Hôtel	Hôtelier	-	
bc1	Hôtel	Réservation.état>="facturée"	Client	Hôtel	
bd1	Hôtel	Hôtel.état="fermé"	Hôtelier	-	
cd1	Réservation.état>"facturée"	Réservation.état="payée"	Client	Hôtel	<Réservation.état="consommée", Réservation.état="payée"> OR (<Réservation.état="partiel_conso", Réservation.état="payée">
cd2	Réservation.état>= "créée"	Réservation.état="annulée"	Client	Hôtel	
cd3	Réservation.état="facturée"	Réservation.état= "consommée"	Client	Hôtel	<Réservation.état="facturée", Réservation.état="annulée">
cd4	Réservation.état="facturée"	Réservation.état= "partiel_conso"	Client	Hôtel	<Réservation.état="facturée", Réservation.état="partiel_conso"> AND <old.DisponibilitéHôtel, new.DisponibilitéHôtel>

Tableau 29 : Documentation des sections de la carte Cs

La directive $DRI_{ab1.ab1.4}$ permet d'affiner les sections ab1, ab2, bd1, cd2 et bc1. Les deux paragraphes ci-dessous présentent les cartes affinant les sections ab2 et bc1. Les cartes affinant les autres sections sont présentées dans l'Annexe 2 (p. 340).

4.1.2 Carte Cs_{ab2} permettant d'offrir un produit par modification de ressources

La carte système Cs_{ab2} affine la section Cs.ab2 <*Démarrer, Offrir produit*, *Par modification de ressources*> de la carte Cs.

Cette carte, présentée en Figure 125, décrit de manière détaillée comment le système enregistre l'ajout de nouvelles chambres dans un hôtel, permet d'enregistrer leur suppression, et gère les réservations et les disponibilités en conséquence.

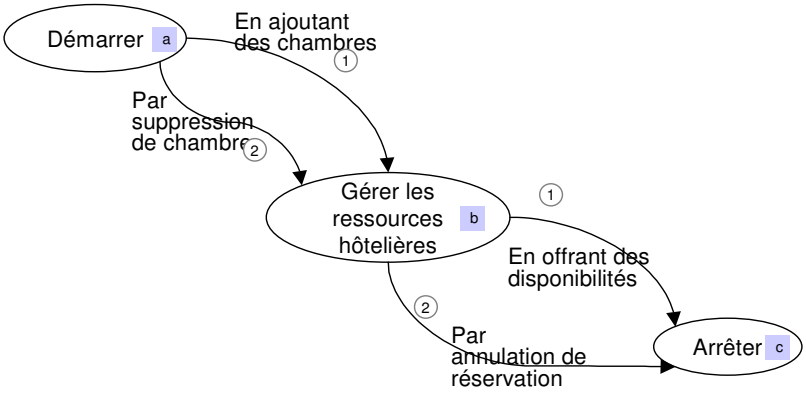


Figure 125 : Carte Cs_{ab2} décrivant l'offre d'un produit par modification de ressources

La carte Cs_{ab2} contient 3 intentions et 4 stratégies.

Intentions de la carte Cs_{ab2}

Les intentions de la carte Cs_{ab2} sont *Démarrer*, *Gérer les ressources hôtelières* et *Arrêter*. Elles sont détaillées comme suit :

- *Démarrer* : pour modifier les ressources hôtelières, il est nécessaire que celles-ci existent. La documentation de l'intention *Démarrer* précise les pré-conditions contrôlées par le système.
- *Gérer les ressources hôtelières* : le système permet de modifier le nombre de chambres associées à un hôtel en ajoutant ou supprimant une chambre.
- *Arrêter* : le système tient compte des répercussions de la suppression de chambres en annulant les réservations qui ne pourraient pas être honorées suite à ces suppressions.

Comme indiqué dans le chapitre 6, une intention peut être documentée par des états. L'analyse du système a permis de mettre en évidence les différents états associés à chacune des trois intentions présentées ci-dessous. Le Tableau 30 présente le résultat de cette analyse.

Code	Intention	états
a	Démarrer	Hôtel.état="ouvert"
b	Gérer les ressources Hôtelières	Hôtel.état="ouvert", new.Hôtel.nbchambres, Réservation.état="créée", Réservation.état="facturée"
c	Arrêter	new.DisponibilitéHôtel, Réservation.état="annulée"

Tableau 30 : Description des intentions de la carte Cs_{ab2}

Stratégies de la carte Cs_{ab2}

Deux stratégies permettent de réaliser l'intention *Gérer les ressources hôtelières* au moyen du système :

- *En ajoutant des chambres* : permet d'augmenter le nombre de chambres dans un hôtel.
- *Par suppression de chambre* : permet de diminuer le nombre de chambres dans un hôtel.

Par ailleurs, le système offre deux stratégies pour terminer le processus :

- *En offrant des disponibilités* : la suppression ou l'ajout de chambres entraîne des modifications dans les disponibilités de l'hôtel.
- *Par annulation de réservation* : le système ne peut supporter plus de réservations dans un hôtel qu'il n'y a de chambres, il peut donc s'avérer nécessaire d'annuler certaines réservations lorsque des chambres sont supprimées du système.

Sections de la carte Cs_{ab2}

Conformément à la directive $DRI_{ab1.ab1.5}$, chaque section de la carte est documentée dans le Tableau 31 ci-dessous en précisant ses pré-conditions, ses post-conditions, ses règles de gestion et éventuellement son acteur déclencheur et ses ressources.

Code	Pré-condition	Post-condition	Acteur	Ressource	Règle de gestion
ab1	Hôtel	Hôtel.nb-chambre= Hôtel.nb-chambre+1	Hôtelier	-	<old.Hôtel.nbchambres, new.Hôtel.nbchambres>
ab2	Hôtel	Hôtel.nb-chambre= Hôtel.nb-chambre- 1	Hôtelier	-	<old.Hôtel.nbchambres, new.Hôtel.nbchambres>
bc1	new.Hôtel.nb-chambre<>old Hôtel.nb-chambre	<old.DisponibilitéHôtel, new.DisponibilitéHôtel>	Hôtelier	Hôtel	<old.DisponibilitéHôtel, new.DisponibilitéHôtel>
bc3	Réservation.état <= "facturée" AND new.Hôtel.nb-chambre < old.Hôtel.nb-chambre	Réservation.état= "annulée"	Hôtelier	Hôtel	<Réservation.état="créée", Réservation.état="annulée"> OR <Réservation.état="facturée", Réservation.état="annulée">

Tableau 31 : Documentation des sections de la carte Cs_{ab2}

Modèle de classes associé à la carte Cs_{ab2}

L'analyse de la carte Cs_{ab2} met en évidence trois classes utilisées par le système : *Réservation*, *Hôtel* et *DisponibilitéHôtel*. L'étude des propriétés des sections de la carte permet d'identifier les attributs correspondants. La Figure 126 présente le modèle de classes associé à la carte Cs_{ab2}.

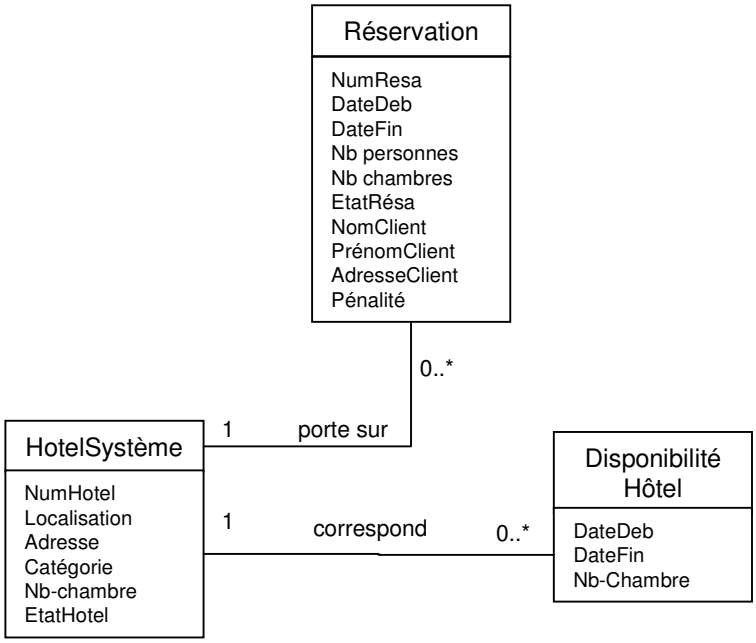


Figure 126 : Modèle de classes associé à la carte Cs_{ab2}

4.1.3 Carte Cs_{bc1} décrivant l'offre des facilités de réservation

La section Cs.bc1 est un paquet de trois stratégies mutuellement exclusives. Chacune des stratégies indique comment le système permet (i) de prendre en compte la demande du client, et (ii) de la satisfaire en gérant les disponibilités, en créant une réservation et en la facturant. L'ensemble des intentions et stratégies sous-jacentes est spécifié dans la carte présentée à la Figure 127.

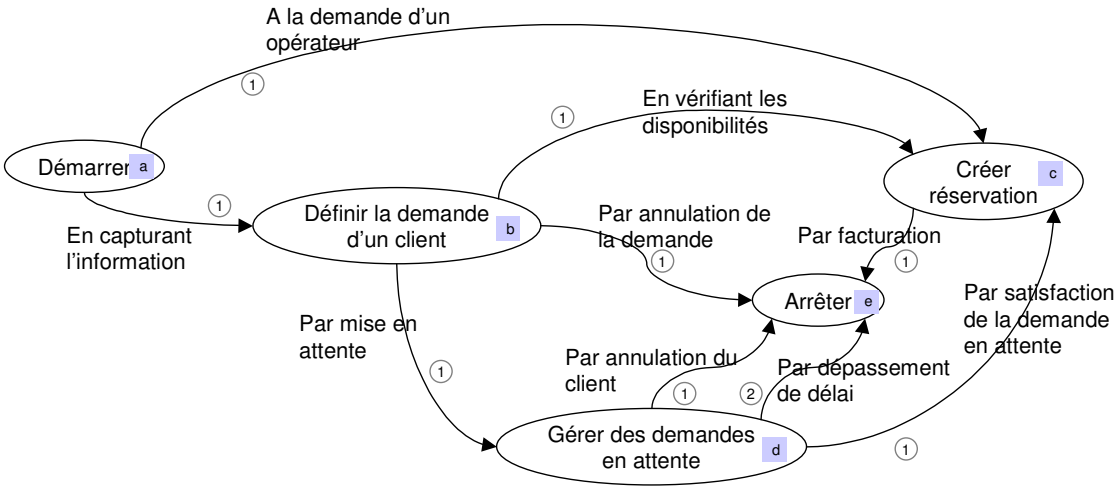


Figure 127 : Carte Cs_{bc1} décrivant comment le système offre des facilités de réservation

Cette carte compte 5 intentions et 9 stratégies.

Intentions de la carte Cs_{bc1}

Les 5 intentions de la carte Cs_{bc1} sont *Démarrer*, *Définir la demande d'un client*, *Créer réservation*, *Gérer des demandes en attente* et *Arrêter*.

- *Démarrer* : cette intention précise que des réservations ne peuvent être faites dans le système que si un catalogue de produits a été préalablement conçu.
- *Définir la demande d'un client* : cette intention correspond à la création d'une demande dans le système.
- *Créer réservation* : cette intention est réalisée quand le système trouve des disponibilités correspondant à la demande du client. Dans ce cas, le système crée la réservation correspondante.
- *Gérer des demandes en attente* : le système met en œuvre cette intention pour les demandes des clients qui ne peuvent être satisfaites directement. En effet, le système cherche à satisfaire ces demandes à chaque fois que de nouvelles disponibilités apparaissent.
- *Arrêter* : cette intention permet d'annuler des demandes de différentes façons et de facturer les réservations.

Le Tableau 32 présente les états associés aux différentes intentions de la carte.

Code	Intention	états
a	Démarrer	Hôtel.état="ouvert"
b	Définir la Demande d"un client	Demande.état = "créée"
d	Gérer des Demandes en attente	Demande.état="attente"
c	Créer Réservation	Demande.état="satisfaite", Réservation.état="créée", new.DisponibilitéHôtel
e	Arrêter	Réservation.état="facturée", Demande.état="annulée"

Tableau 32 : Description des intentions de la carte Cs_{bc1}

Stratégies de la carte Cs_{bc1}

Une stratégie indique comment atteindre l'intention *Définir la demande d'un client* avec le système.

- *En capturant l'information* : le système permet de récolter toute l'information nécessaire pour créer une demande, c'est-à-dire, la destination, la période, le nombre de chambres et le nombre de personnes. La façon de récolter cette information sera différente suivant que la demande se fait sur place, par téléphone ou par Internet.

Trois stratégies indiquent comment le système permet de *Créer une réservation*.

- *En vérifiant les disponibilités* : selon cette stratégie, le système crée une réservation à partir d'une demande pour laquelle il est capable d'identifier des disponibilités.
- *Par satisfaction de la demande en attente* : le système crée une réservation satisfaisant une demande en attente lorsqu'il détecte une disponibilité.
- *A la demande d'un opérateur* : cette stratégie permet à un opérateur de créer une réservation en cas de besoin.

Le système n'offre qu'un service dans le cadre de la gestion des demandes en attente. Ce service est identifié par la stratégie ci-dessous.

- *Par mise en attente* : cette stratégie permet, si le client a donné son accord, de mettre en attente une demande qui ne peut pas être satisfaite immédiatement.

Quatre stratégies permettent de terminer le processus.

- *Par annulation de la demande* : lorsqu'une demande ne peut être satisfaite, et que le client refuse de la mettre en attente, elle est annulée.
- *Par annulation du client* : il est possible pour un client d'annuler sa demande en attente.
- *Par dépassement de délai* : quinze jours avant le début de la période de réservation, les demandes en attente sont automatiquement supprimées.
- *Par facturation* : le système met en œuvre cette stratégie pour facturer toute réservation qui vient d'être créée.

Sections de la carte Cs_{bc1}

Le Tableau 33 documente les sections de la carte Cs_{bc1} en spécifiant ses pré-conditions, ses post-conditions, ses règles de gestion et, éventuellement, son acteur déclencheur et ses ressources conformément aux explications données ci-dessus.

Code	Pré-condition	Post-condition	Acteur	Ressource	Règle de gestion
ab1	Hôtel	Demande.état = "créée"	Client	Hôtel	<-, Demande.état="créée">
de1	Demande.état="en attente" AND Demande.DateDeb > date-du-jour + 7j	Demande.état= "annulée"	Client	Hôtel	<Demande.état="attente", Demande.état="annulée">
bd1	Demande.état="créée" AND Demande.PeutEtreEnAttente = vrai AND (DisponibilitéHôtel.DateDeb > Demande.DateDeb OR DisponibilitéHôtel.DateFin < Demande.DateFin)	Demande.état="en attente"	Opérateur	Hôtel	<Demande.état="créée", Demande.état="attente">
be1	Demande.état="créée" AND (Demande.PeutEtreEnAttente = faux AND (DisponibilitéHôtel.DateDeb > Demande.DateDeb OR DisponibilitéHôtel.DateFin < Demande.DateFin))	Demande.état="annulée"	Opérateur	Hôtel	<Demande.état="créée", Demande.état="annulée">
dc1	Demande.état="en attente" AND DisponibilitéHôtel.DateDeb <= Demande.DateDeb AND DisponibilitéHôtel.DateFin >= Demande.DateFin	Demande.état="satisfaite" AND Reservation.état="créée" AND	-	Hôtel	<Demande.état="en attente", Demande.état="satisfaite"> AND <old.DisponibilitéHôtel, new.DisponibilitéHôtel>
bc1	Demande.état="créée" AND DisponibilitéHôtel.DateDeb <= Demande.DateDeb AND DisponibilitéHôtel.DateFin >= Demande.DateFin	Demande.état="satisfaite" AND Reservation.état="créée" AND	Opérateur	Hôtel	<Demande.état="créée", Demande.état=satisfaite"> AND <-, Réservation.état="créée">
ce1	Réservation.état="créée"	Réservation.état="facturée"	-	Hôtel	<Réservation.état="créée", Réservation.état="facturée">
de2	Demande.état="en attente" AND Demande.DateDeb < date-du-jour + 7j	Demande.état="annulée"	-	Hôtel	<Demande.état="attente", Demande.état="annulée">
ac1	Réservation.état="créée"	Réservation.état="créée"	Opérateur	Hôtel	<-, Réservation.état="créée">

Tableau 33 : Documentation des sections de la carte Cs_{bc1}

Modèle de classes associé à la carte Cs_{bc1}

Une analyse des intentions et des sections de la carte Cs_{bc1} permet d'identifier la classe *Demande* en plus des classes *Réservation*, *Hôtel* et *DisponibilitéHotel*, et de spécifier les propriétés de ces quatre classes. La Figure 128 présente le modèle de classes résultant de cette analyse.

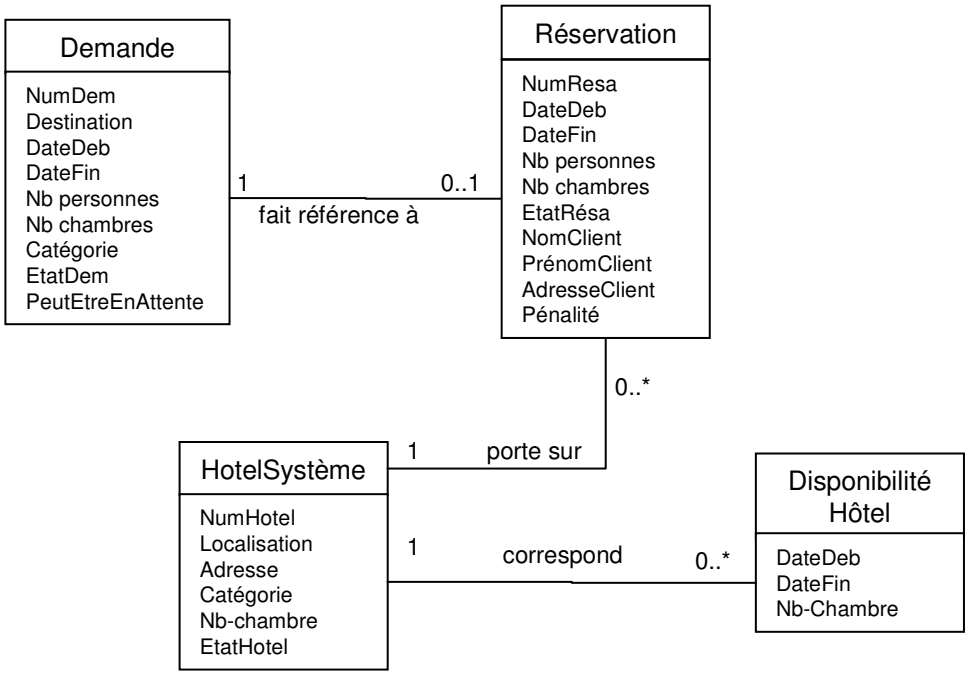


Figure 128 : Modèle de classes associé à la carte Cs_{bc1}

4.2. Construire les cartes processus

Une démarche analogue à celle présentée dans la section ci-dessus est appliquée pour construire les cartes des processus d'entreprise. La directive $DRI_{ab1.2}$ de la méthode ACEM propose d'abstraire les objectifs du métier pour construire les cartes processus.

Les trois sous-sections suivantes présentent respectivement les cartes Cp, Cp_{ab2} et $Cp_{bc1.}$ Comme pour les cartes du système et pour des raisons de place et de lisibilité, nous avons choisi de ne pas montrer les détails de l'affinement de toutes les sections de Cp.

4.2.1 Carte Cp de plus haut niveau

Comme pour le système, l'objectif des processus organisationnels est d'optimiser le remplissage des hôtels par la mise en place d'un système informatisé centralisé. La carte Cp décrivant la mise en œuvre de cet objectif de haut niveau est présentée en Figure 129. La figure montre que deux préoccupations sont au centre de ces processus : *Offrir un produit* et *Gérer les réservations*. On observe que ces intentions sont les mêmes que celles présentées en Figure 124 dans la carte Cs du système.

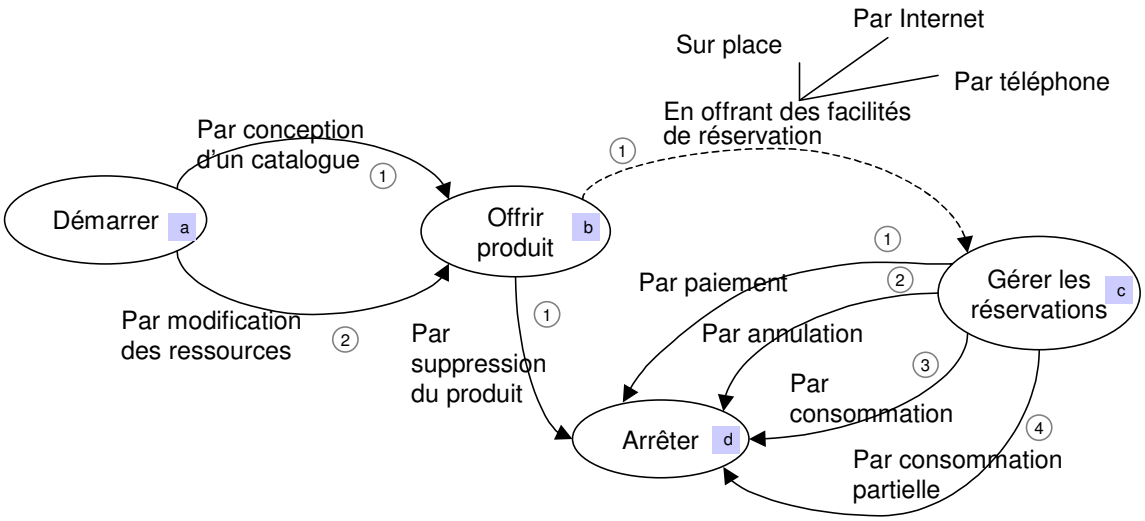


Figure 129 : la carte processus Cp de plus haut niveau

Une comparaison détaillée de la Figure 129 et de la Figure 124 montre que les cartes Cs et Cp partagent les mêmes intentions, les mêmes stratégies et les mêmes sections. Ceci confirme que le système prend effectivement en charge les objectifs assignés aux processus. En revanche, cela ne permet pas de connaître la qualité de l'alignement : les ruptures de l'alignement identifiées au chapitre 4 ne sont pas perceptibles à ce haut niveau de définition des objectifs.

Les propriétés des sections de la carte Cp sont récapitulées dans le Tableau 34. Les sections de la carte Cp pour lesquelles aucune règle de gestion n'est présentée sont affinées par des cartes.

Code	Pré-condition	Post-condition	Acteur	Ressource	Règle de gestion
ab1	rien	Hôtel, Station, Chambre, Disponibilité	Hôtelier	-	
bc1	Chambre	Réservation.état>="facturée"	Hôtelier	-	
ab2	Hôtel OR Station OR Chambre	Hôtel OR Station OR Chambre, Periode	Client	Hôtel, Station	
bd1	Hôtel OR Station	Hote.état="fermé" OR Station.état="fermée"	Hôtelier	-	
cd1	Réservation.état>="créée"	Réservation.état="annulée"	Client	Hôtel, Station	<Réservation.état="consommée", Réservation.état="payée"> OR (<Réservation.état="partiel_conso", Réservation.état="payée">
cd2	Réservation.état="facturée"	Réservation.état= "consommée"	Client	Hôtel, Station	
cd3	Réservation.état="facturée"	Réservation.état= "partiel_conso"	Client	Hôtel, Station	<Réservation.état="facturée", Réservation.état="annulée"> AND <old.Disponibilité, new.Disponibilité>
cd4	Réservation.état>"facturée"	Réservation.état="payée"	Client	Hôtel, Station	<Réservation.état="facturée", Réservation.état="partiel_conso">

Tableau 34 : Documentation des sections de la carte Cp

La suite de la section présente les cartes Cp_{ab2} et Cp_{bc1} affinant respectivement les sections ab2 et bc1 de la carte Cp. Les cartes Cp_{ab1} et Cp_{bd1} sont décrites en Annexe 2 (p. 340).

4.2.2 Carte Cp_{ab2} décrivant les processus d'offre d'un produit par modification de ressources

La carte Cp_{ab2} affine la section Cp.ab2 <*Démarrer, Offrir produit*, *Par modification de ressources*> de la carte Cp.

Cette section décrit les processus de création de nouvelles chambres dans un hôtel, de suppression de chambres, de fermeture temporaire d'un hôtel, de fermeture temporaire d'une station et de gestion des réservations et des disponibilités en conséquence. Ces processus peuvent être représentés, de façon abstraite, par les intentions et stratégies rassemblées dans la carte présentée en Figure 130.

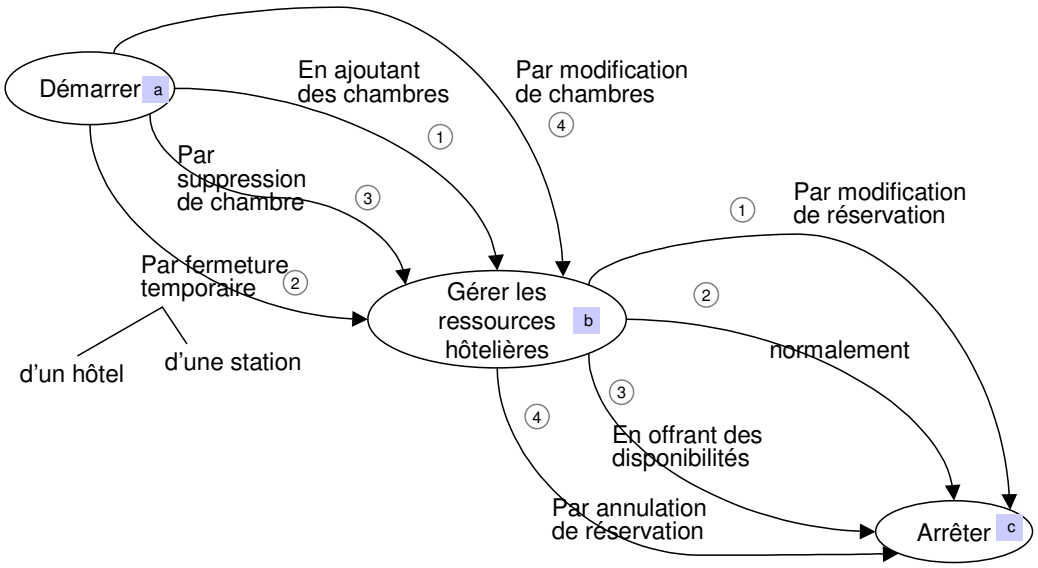


Figure 130 : Carte Cp_{ab2} décrivant les processus permettant d'offrir un produit par modification de ressources

Intentions de la carte Cp_{ab2}

La carte Cp_{ab2} contient trois intentions : *Démarrer*, *Gérer les ressources hôtelières* et *Arrêter*. Ces deux dernières sont détaillées comme suit.

- *Gérer les ressources hôtelières* : selon cette intention, la modification des ressources vise entre autres à modifier les caractéristiques d'une chambre, créer ou supprimer des chambres ainsi que fermer temporairement une station ou un hôtel.

- *Arrêter* : indique qu'il est nécessaire de gérer les disponibilités des chambres en tenant compte de la création ou de la suppression de chambres, de la fermeture temporaire d'une station ou d'un hôtel et éventuellement d'annuler des réservations qui ne pourraient pas être honorées à la suite de ces changements.

Le Tableau 35 présente les états associés aux trois intentions de la carte Cp_{ab2}.

Code	Intention	états
a	Démarrer	Hôtel.état="ouvert", Hôtel.état="fermé_temp", Station.état="ouverte", Station.état="fermée_temp"
b	Gérer les ressources Hôtelières	Station_Fermeture_temp.état="créée", Station.état="ouverte", Station.état="fermée_temp", new.Période, old.Chambre.prixch, new.Chambre.prixch
c	Arrêter	new.Disponibilité, Réservation.état="créée", Réservation.état="annulée" Réservation.état="facturée",

Tableau 35 : Description des intentions de la carte Cp_{ab2}

Stratégies de la carte Cp_{ab2}

Quatre stratégies permettent d'atteindre l'intention *Gérer les ressources hôtelières* depuis l'intention *Démarrer* :

- *En ajoutant des chambres* : qui permet de créer une nouvelle chambre associée à un hôtel.

- *Par fermeture temporaire* : cette stratégie est un paquet. Elle permet de fermer temporairement une station ou un hôtel.

- *Par suppression de chambres* : permet de supprimer définitivement une chambre d'un hôtel.

- *Par modification de chambres* : permet de modifier les caractéristiques d'une chambre, en particulier son prix.

Quatre stratégies permettent de terminer le processus :

- *Normalement* : lorsque la modification d'une chambre n'a aucun impact sur les réservations en cours, le processus se termine normalement.

- *En offrant des disponibilités* : cette stratégie est mise en œuvre lorsque la suppression, la création de chambre ou la fermeture temporaire d'une station ou d'un hôtel entraîne des modifications dans les disponibilités des chambres.

- *Par annulation de réservation* : les réservations qui portent sur les chambres supprimées et ne peuvent pas être reportées sur d'autres chambres de l'hôtel doivent être annulées. Il en est de même pour les réservations dans des hôtels ou des stations qui ferment temporairement.

- *Par modification de réservation* : les réservations portant sur les chambres supprimées peuvent éventuellement, s'il reste des chambres libres, être modifiées et affectées à ces chambres.

Sections de la carte Cp_{ab2}

Chaque section de la carte est documentée dans le Tableau 36 en précisant ses pré-conditions, ses post-conditions, ses règles de gestion, l'acteur déclencheur et les ressources.

Code	Pré-condition	Post-condition	Acteur	Ressource	Règle de gestion
ab1	Hôtel	Chambre	Hôtelier	Hôtel, Station	<-, Chambre.état="ouverte">
ab2	Hôtel OR Station	new.Période AND Hôtel.état="fermé_temp" AND Station.état="fermée_temp" AND Chambre.état="fermée"	Hôtelier	-	[<-, new.Période> AND <Station.état="ouverte", Station.état="fermée_temp">] OR [<-, new.Période> AND <Hôtel.état="ouvert", Hôtel.état="fermé_temp">]
ab3	Hôtel	Chambre.état="fermée"	Hôtelier	Hôtel, Station	<Chambre.état="ouverte", Chambre.état="fermée">
bc1	Réservation.état <= "facturée" AND Chambre.état="fermée" AND Chambre1.état="ouverte" AND Disponibilité.DateDeb <= Demande.DateDeb AND Disponibilité.DateFin >= Demande.DateFin	Réservation.état="créée"	Hôtelier	Hôtel, Station	<old.Réservation.Chambre, new.Réservation.Chambre>
bc2	Chambre.état="fermée" AND Non Réservation.état<=facturée	rien	Hôtelier	Hôtel, Station	-
bc3	chambre.état="fermée"	Disponibilité (supprimée) et ajoutée	Hôtelier	Hôtel, Station	<old.DisponibilitéHôtel, new.DisponibilitéHôtel>
bc4	Réservation.état <= "facturée" AND Chambre.état="fermée" AND NON (Chambre1.état="ouverte" AND Disponibilité.DateDeb <= Demande.DateDeb AND Disponibilité.DateFin >= Demande.DateFin)	Réservation.état="annulée"	Hôtelier	Hôtel, Station	<Réservation.état="créée", Réservation.état="annulée"> OR <Réservation.état="facturée", Réservation.état="annulée">
ab4	Chambre	Chambre	Hôtelier	Hôtel, Station	<old.Chambre.prixch, new.Chambre.prixch>

Tableau 36 : Documentation des sections de la carte Cp_{ab2}

Modèle de classes associé à la carte Cp_{ab2}

L'étude des sections de la carte Cp_{ab2} permet d'identifier un certain nombre de concepts formant le modèle de classes présenté à la Figure 131. Les attributs des classes sont identifiés à partir des propriétés des sections documentées dans le Tableau 36, le Tableau 37 et le Tableau 38.

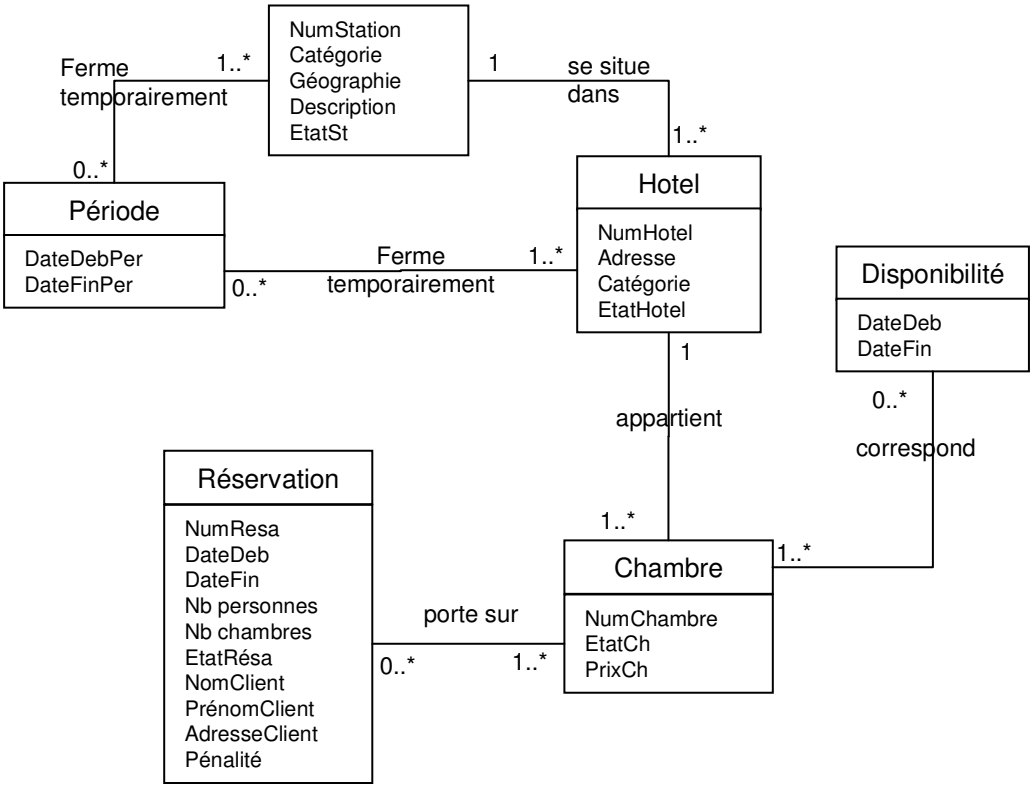


Figure 131 : Modèle de classes associé à la carte Cp_{ab2}

4.2.3 Carte Cp_{bc1} décrivant les processus d'offre des facilités de réservation

Comme la section Cs.bc1, la section Cp.bc1 est un paquet recouvrant la gestion des réservations de la création de la demande à la facturation de la réservation.

La carte Cp_{bc1} est présentée par la Figure 132 ci-dessous.

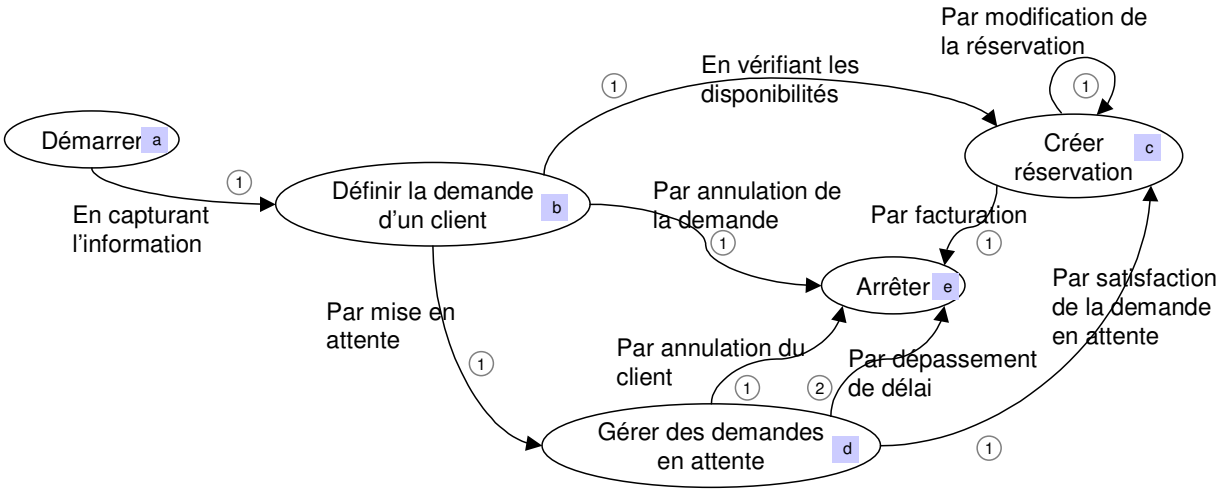


Figure 132 : Carte Cp_{bc1} permettant d'offrir des facilités de réservation

Cette carte compte 5 intentions et 9 stratégies.

Intentions de la carte Cp_{bc1}

Les intentions de la carte Cp_{bc1} *Démarrer*, *Définir la demande d'un client*, *Créer réservation*, *Gérer des demandes en attente* et *Arrêter* sont les suivantes :

- *Démarrer* : afin de permettre aux clients d'effectuer des demandes, il est nécessaire qu'un catalogue de produits ait été préalablement conçu et que des stations et des hôtels soient ouverts.
- *Définir la demande d'un client* : cette intention correspond à la création d'une demande.
- *Créer réservation* : lorsque les disponibilités de chambres le permettent, la demande du client est satisfaite et une réservation correspondante est créée.
- *Gérer des demandes en attente* : les demandes insatisfaites des clients peuvent être mises en attente. Leur satisfaction se fait quand des disponibilités se libèrent, suite à des annulations de réservation.
- *Arrêter* : permet d'annuler des demandes de différentes façons et de facturer les réservations.

Le Tableau 37 décrit les intentions de la carte Cp_{bc1} sous forme d'ensembles d'états.

Code	Intention	états
a	Démarrer	Station.état="ouverte", Hôtel.état="ouvert"
b	Définir la Demande d"un client	Demande.état = "créée"
d	Gérer des Demandes en attente	Demande.état="attente"
c	Créer Réservation	Demande.état="satisfaite", Réservation.état="créée", new.Disponibilité
e	Arrêter	Réservation.état="facturée", Demande.état="annulée", Demande.état="sans_suite", Demande.état="hors delai"

Tableau 37 : Description des intentions de la carte Cp_{bc1}

Stratégies de la carte Cp_{bc1}

Une seule stratégie permet d'atteindre l'intention *Définir la demande d'un client*.

- *En capturant l'information* : cette stratégie permet de récolter toute l'information nécessaire pour créer une demande, c'est-à-dire, la destination, la période, le nombre de chambres, et le nombre de personnes.

Trois stratégies spécifient la réalisation de l'intention *Créer réservation*.

- *En vérifiant les disponibilités* : cette stratégie correspond à la création d'une réservation à partir d'une demande, si des disponibilités permettent de la satisfaire.
- *Par satisfaction de la demande en attente* : permet de satisfaire une demande en attente pour laquelle il existe des disponibilités et de créer une réservation correspondante.
- *Par modification d'une réservation* : cette stratégie permet de modifier une réservation existante.

Une stratégie permet de *Gérer des demandes en attente*.

- *Par mise en attente* : lorsqu'une demande ne peut être satisfaite, celle-ci est mise en attente si le client a donné son accord.

Quatre stratégies permettent de terminer le processus :

- *Par annulation de la demande* : lorsqu'une demande ne peut être satisfaite et que le client refuse de la mettre en attente, elle est considérée comme 'sans_suite' ce qui met fin au processus.
- *Par annulation du client* : il est possible pour un client d'annuler sa demande en attente.
- *Par dépassement de délai* : quinze jours avant le début de la période de réservation, les demandes en attente sont considérées 'hors délai' et ne peuvent plus être satisfaites.
- *Par facturation* : cette stratégie permet de facturer une réservation qui vient d'être créée.

Sections de la carte Cp_{bc1}

Les sections de la carte sont décrites dans le Tableau 38 avec leurs pré-conditions, post-conditions, règles de gestion, acteurs et ressources associés.

Code	Pré-condition	Post-condition	Acteur	Ressource	Règle de gestion
ab1	Hôtel, Station	Demande.état = "créée"	Client	Hôtel, Station	<-, Demande.état="créée">
de1	Demande.état="en attente" AND Demande.DateDeb > date-du-jour + 7j	Demande.état= "annulée"	Client	Hôtel, Station	<Demande.état="attente", Demande.état="annulée">
bd1	Demande.état="créée" AND Demande.PeutEtreEnAttente = vrai AND (Disponibilité.DateDeb > Demande.DateDeb OR Disponibilité.DateFin < Demande.DateFin)	Demande.état="en attente"	Opérateur	Hôtel, Station	<Demande.état="créée", Demande.état="attente">
be1	Demande.état="créée" AND (Demande.PeutEtreEnAttente = faux AND (Disponibilité.DateDeb > Demande.DateDeb OR Disponibilité.DateFin < Demande.DateFin))	Demande.état="sans_suite"	Opérateur	Hôtel, Station	<Demande.état="créée", Demande.état="sans_suite">
dc1	Demande.état="en attente" AND Disponibilité.DateDeb <= Demande.DateDeb AND Disponibilité.DateFin >= Demande.DateFin	Demande.état="satisfaite" AND Reservation.état="créée"	-	Hôtel, Station	<Demande.état="en attente", Demande.état="satisfaite"> AND <old.Disponibilité, new.Disponibilité>
bc1	Demande.état="créée" AND Disponibilité.DateDeb <= Demande.DateDeb AND Disponibilité.DateFin >= Demande.DateFin	Demande.état="satisfaite" AND Reservation.état="créée"	Opérateur	Hôtel, Station	<Demande.état="créée", Demande.état=satisfaite"> AND <-, Réservation.état="créée">
ce1	Réservation.état="créée"	Réservation.état="facturée"	-	Hôtel, Station	<Réservation.état="créée", Réservation.état="facturée">
de2	Demande.état="en attente" AND Demande.DateDeb < date-du-jour + 7j	Demande.état="hors delai"	-	Hôtel, Station	<Demande.état="attente", Demande.état="hors delai">
cc1	Réservation.état="créée" OR Réservation.état="facturée"	<old.Réservation.Hôtel, new.Réservation.Hôtel> OR <old.Réservation.Chambre, new.Réservation.Chambre>	Opérateur	Hôtel, Station	<old.Réservation.Hôtel, new.Réservation.Hôtel> OR <old.Réservation.Chambre, new.Réservation.Chambre>

Tableau 38 : Description des sections de la carte Cp_{bc1}

Modèle de classes associé à la carte Cp_{bc1}

Une analyse des sections de la carte Cp_{bc1} permet de lui associer le modèle de classes présenté en Figure 133.

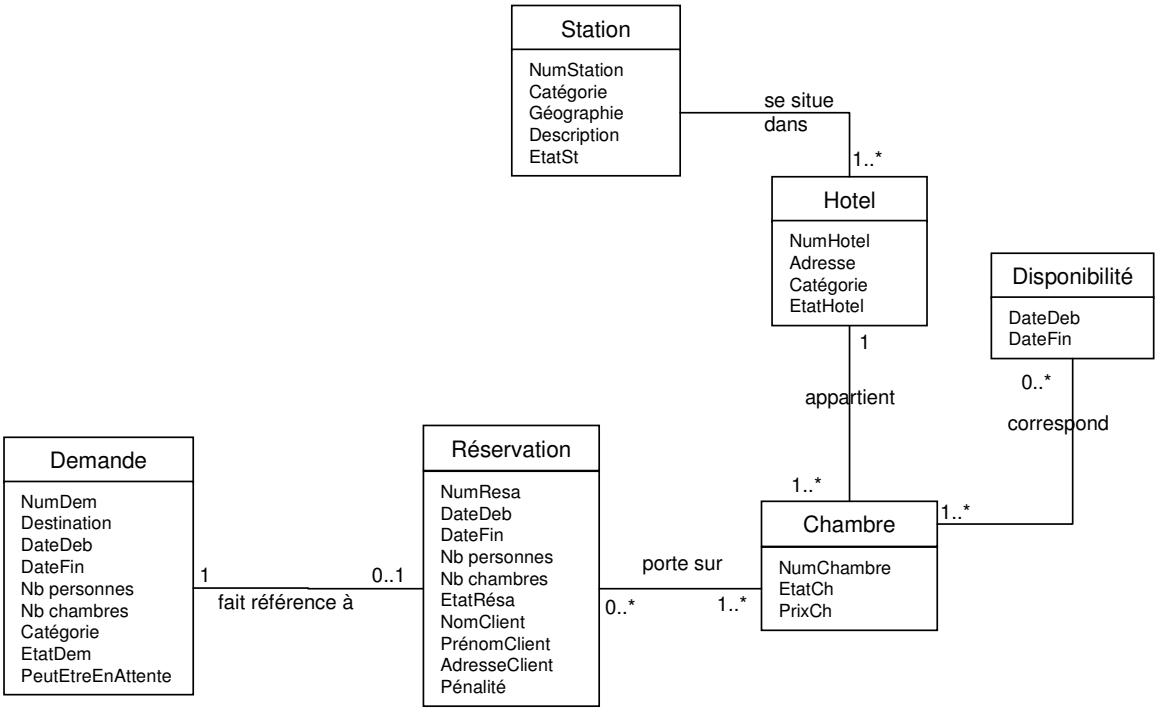


Figure 133 : Modèle de classes associé à la carte Cp_{bc1}

Les cartes du système et des processus étant construites, il est désormais possible de construire la carte unifiée du modèle pivot représentant conjointement le système et les processus.

4.3. Construction de la carte pivot

La construction de la carte pivot peut se faire de deux façons différentes en suivant les directives $DRI_{ab1.4}$ (*Construire la carte pivot stratégie dirigée par le système*) et $DRI_{ab1.5}$ (*Construire la carte pivot stratégie dirigée par le processus*) de la méthode ACEM. Nous avons choisi de suivre la $DRI_{ab1.5}$.

Nous ne présentons dans cette section que la construction de la carte de haut niveau Cu ainsi que des cartes affinant les sections ab2 et bc1 de cette carte.

4.3.1 Construction de la carte pivot de haut niveau

Une analyse des intentions de la carte Cp puis de ses sections montre qu'elles sont toutes respectivement identiques à une intention ou une section de la carte Cs. En effet, pour chaque intention de la carte Cp, par construction il existe, dans la carte Cs, une intention ayant le même nom et une définition similaire. En appliquant la directive $DRI_{ab1.4.1.1}$, on peut affirmer que les intentions de Cp sont similaires à celles de Cs. La carte Cs ne possèdant pas d'autres intentions, il est possible de construire la carte pivot correspondante en fusionnant les intentions de la carte Cp et celles de la carte Cs. Une étude des sections entre *Démarrer* et *Offrir produits* montre que les sections de Cs sont identiques à celles de Cp. L'analyse des autres sections mène à la même conclusion. La fusion des cartes Cs et Cp aboutit donc à une carte Cu identique à Cs et Cp, entièrement coloriée en blanc, indiquant que tous ses éléments

sont communs au système et au processus. Le degré d'alignement des différentes sections est donc *totalement aligné*. La Figure 134 présente la carte pivot Cu de plus haut niveau.

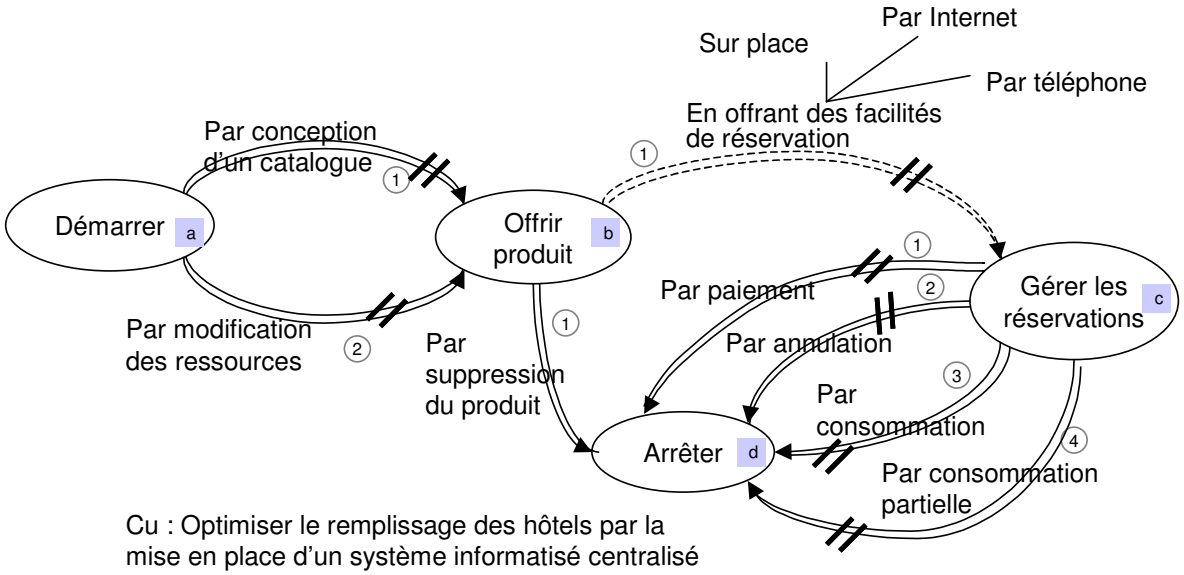


Figure 134 : Carte pivot Cu de plus haut niveau

La carte pivot Cu montre que les objectifs de haut niveau du système et des processus sont alignés. Ceci confirme l'impression des directeurs des systèmes d'information et des chefs d'entreprise qui considèrent majoritairement, qu'au niveau stratégique, système et processus sont en concordance.

Cependant, la coloration de la carte Cu n'implique pas que les cartes affinées soient blanches On verra qu'en fait les cartes de niveau d'affinement supérieur montrent des différences entre le système et les processus. Il est donc important et utile d'analyser la relation d'alignement à plusieurs niveaux de détails.

La carte pivot Cu ne contredit pas les mesures effectuées au chapitre 4. Celles-ci correspondent à un niveau de granularité plus fin que Cu. On verra dans les sections suivantes que les cartes affinées confirment les disparités apparues dans les mesures.

4.3.2 Construction de la carte pivot affinant la section ab2

La directive $DRI_{ab1.5}$ propose de commencer la construction de la carte pivot par l'étude des intentions puis de poursuivre par l'étude des sections.

Chacune des deux cartes Cs_{ab2} et Cp_{ab2} compte trois intentions. L'étude des intentions de la carte Cp_{ab2}, montre que chacune d'entre elles a le même nom et la même sémantique qu'une intention de la carte Cs_{ab2}. Toutes les intentions de la carte Cp_{ab2} sont donc fusionnées avec celles de la carte Cs_{ab2} et sont coloriées en blanc (Figure 135).

Quatre sections de la carte Cp_{ab2}, <*Démarrer*, *Gérer les ressources hôtelières*, *Par modification de chambre*>, <*Démarrer*, *Gérer les ressources hôtelières*, *Par fermeture temporaire*>, <*Gérer les ressources hôtelières*, *Arrêter*, *Par modification de réservations*> et <*Gérer les ressources hôtelières*, *Arrêter*, *Normalement*> n'ont pas d'équivalent dans la carte Cs_{ab2}. Ces quatre sections sont donc ajoutées dans la carte pivot Cu_{ab2} et coloriées en gris foncé. Une étude détaillée des cinq autres sections de la carte Cp_{ab2} montre qu'elles sont

toutes identiques à l'une des sections de la carte Cs_{sb2}. Ces sections sont donc colorées en blanc, le degré d'alignement *totalement aligné* leur est attribué.

Une étude des quatre sections coloriées en gris foncé permet de dire :

- La section <*Démarrer*, *Gérer les ressources hôtelières*, *Par modification de chambres*> est partiellement alignée avec les sections issues de la carte système <*Démarrer*, *Gérer les ressources hôtelières*, *En ajoutant des chambres*> et <*Démarrer*, *Gérer les ressources hôtelières*, *Par suppression de chambres*>. En effet, supprimer des chambres puis en ajouter d'autres permet au final de modifier les caractéristiques de ces chambres. Cependant, dans la mesure où de nouvelles chambres sont créées et où de l'information est perdue, nous considérons qu'entre ces sections le degré d'alignement est partiellement aligné.

- La section <*Démarrer*, *Gérer les ressources hôtelières*, *Par fermeture temporaire*> est partiellement alignée avec les sections issues de la carte système <*Démarrer*, *Gérer les ressources hôtelières*, *En ajoutant des chambres*> et <*Démarrer*, *Gérer les ressources hôtelières*, *Par suppression de chambre*>. En effet, ces deux dernières sections permettent de gérer les conséquences d'une fermeture temporaire d'hôtel ou de station. Elles sont donc proches de la section <*Démarrer*, *Gérer les ressources hôtelières*, *Par fermeture temporaire*> sans être identiques. Le degré d'alignement *partiellement aligné* est donc choisi.

- La section <*Gérer les ressources hôtelières*, *Arrêter*, *Par modification de réservations*> est partiellement alignée avec la section du système <*Gérer les ressources hôtelières*, *Arrêter*, *Par annulation de réservations*>. En, effet, on considère que la modification d'une réservation est proche mais non similaire à l'annulation suivie de la création d'une nouvelle réservation.

- La section <*Gérer les ressources hôtelières*, *Arrêter*, *Normalement*> est partiellement alignée avec la section du système <*Gérer les ressources hôtelières*, *Arrêter*, *En offrant des disponibilités*>. En effet, la section <*Gérer les ressources hôtelières*, *Arrêter*, *Normalement*> permet d'arrêter le processus de modification de ressources sans qu'aucune disponibilité ne soit ajoutée ou supprimée. Les mêmes disponibilités sont donc offertes. En revanche, la section <*Gérer les ressources hôtelières*, *Arrêter*, *En offrant des disponibilités*> permet éventuellement de modifier les disponibilités proposées. Ces deux sections sont donc proches, mais ont chacune leurs particularités d'où on considère que leur degré d'alignement est *partiellement aligné*.

La section <*Démarrer*, *Gérer les ressources hôtelières*, *En ajoutant des chambres*> issue du système est ainsi totalement alignée avec la section <*Démarrer*, *Gérer les ressources hôtelières*, *En ajoutant des chambres*> issue des processus et partiellement alignée avec les sections <*Démarrer*, *Gérer les ressources hôtelières*, *Par modification de chambre*> et <*Démarrer*, *Gérer les ressources hôtelières*, *Par fermeture temporaire*>. Il en est de même pour les autres sections issues du système. La Figure 135 présente la carte pivot affinant la section ab2.

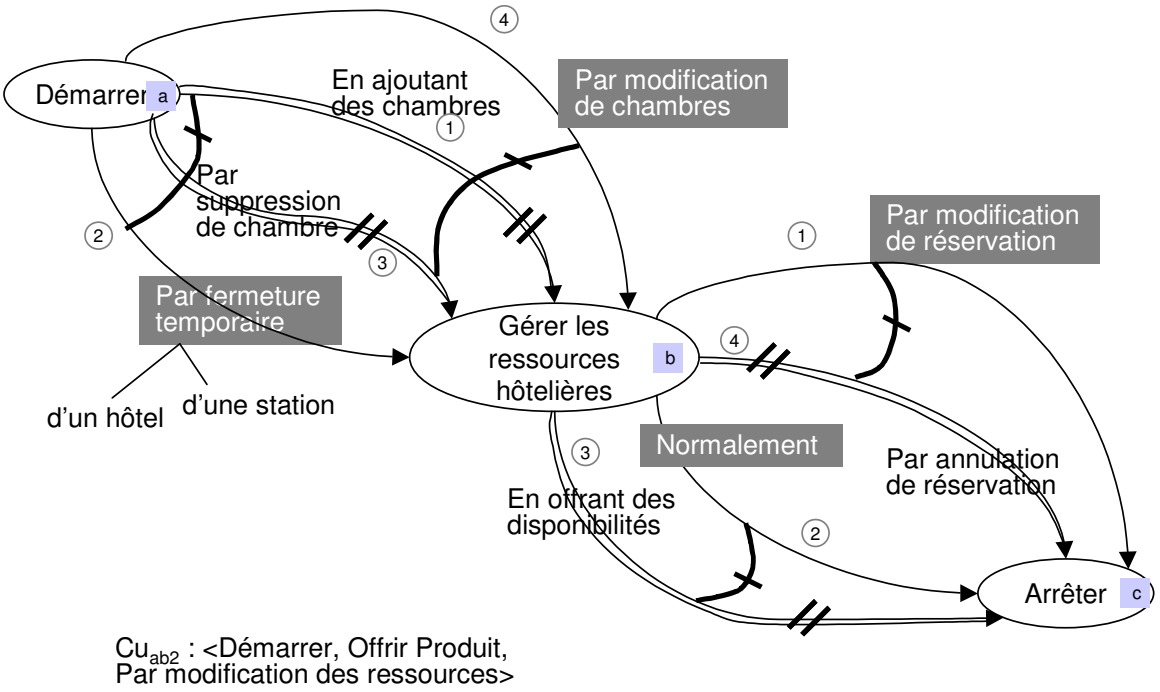


Figure 135 : Carte pivot affinant la section ab2

Les intentions de la carte pivot sont documentées dans le Tableau 39.

Code	Intention	états
a	Démarrer	Hôtel.état="ouvert", Hôtel.état="fermé_temp"
b	Gérer les ressources Hôtelières	Station.état="ouverte", Station.état="fermée_temp", Hôtel.état="ouvert", Hôtel.état="fermé_temp", Chambre.état="fermée", old.Chambre.prixch, new.Chambre.prixch, old.Hôtel.nbchambres, new.Hôtel.nbchambres, Réservation.état="créée", Réservation.état="facturée", new.Période
c	Arrêter	old.Réservation.Chambre, new.Réservation.Chambre, old.Réservation.Hôtel, new.Réservation.Hôtel, new.Disponibilité, Réservation.état="créée", Réservation.état="annulée" Réservation.état="facturée", new.DisponibilitéHôtel

Tableau 39 : Description des intentions de la carte pivot Cu_{ab2}

Les sections de la carte Cu_{ab2} sont décrites dans le Tableau 40.

Code	Pré-condition	Post-condition	Acteur	Ressource	Règle de gestion
ab1	Hôtel,	Chambre, Hôtel.nb-chambre= Hôtel.nb-chambre+1	Hôtelier	Hôtel, Station	<-, Chambre.état="ouverte">, <old.Hôtel.nbchambres, new.Hôtel.nbchambres>
ab2	Hôtel OR Station	Hôtel.état="fermé_temp" AND Station.état="fermée_temp" AND Chambre.état="fermée" AND new.Période	Hôtelier	-	[<-, new.Période> AND <Station.état="ouverte", Station.état="fermée_temp">] OR [<-, new.Période> AND <Hôtel.état="ouvert", Hôtel.état="fermé_temp">]
ab3	Hôtel	Chambre.état="fermée", Hôtel.nb-chambre= Hôtel.nb-chambre-1	Hôtelier	Hôtel, Station	<Chambre.état="ouverte", Chambre.état="fermée">, <old.Hôtel.nbchambres, new.Hôtel.nbchambres>
bc1	Réservation.état <= "facturée" AND Chambre.état="fermée" AND Chambre1.état="ouverte" AND Disponibilité.DateDeb <= Demande.DateDeb AND Disponibilité.DateFin >= Demande.DateFin	Réservation.état="créée"	Hôtelier	Hôtel, Station	<old.Réservation.Chambre, new.Réservation.Chambre>
bc2	Chambre.état="fermée" AND Non Réservation.état<=facturée	rien	Hôtelier	Hôtel, Station	-
bc3	Chambre.état="fermée" new.Hôtel.nb-chambre<>old Hôtel.nb-chambre	<-, new.Disponibilité> OR <-, new.DisponibilitéHôtel>	Hôtelier	Hôtel, Station	<old.Disponibilité, new.Disponibilité> OR <old.DisponibilitéHôtel, new.DisponibilitéHôtel>
bc4	Réservation.état <= "facturée" AND Chambre.état="fermée" AND NON (Chambre1.état="ouverte" AND Disponibilité.DateDeb <= Demande.DateDeb AND Disponibilité.DateFin >= Demande.DateFin) AND new.Hôtel.nb-chambre<old Hôtel.nb-chambre	Réservation.état="annulée"	Hôtelier	Hôtel, Station	<Réservation.état="créée", Réservation.état="annulée"> OR <Réservation.état="facturée", Réservation.état="annulée">
ab4	Chambre	Chambre	Hôtelier	Hôtel, Station	<old.Chambre.prixch, new.Chambre.prixch>

Tableau 40 : Description des sections de la carte pivot Cu_{ab2}

La Figure 136 présente les classes utilisées par les sections et les intentions de la carte pivot Cu_{ab2}.

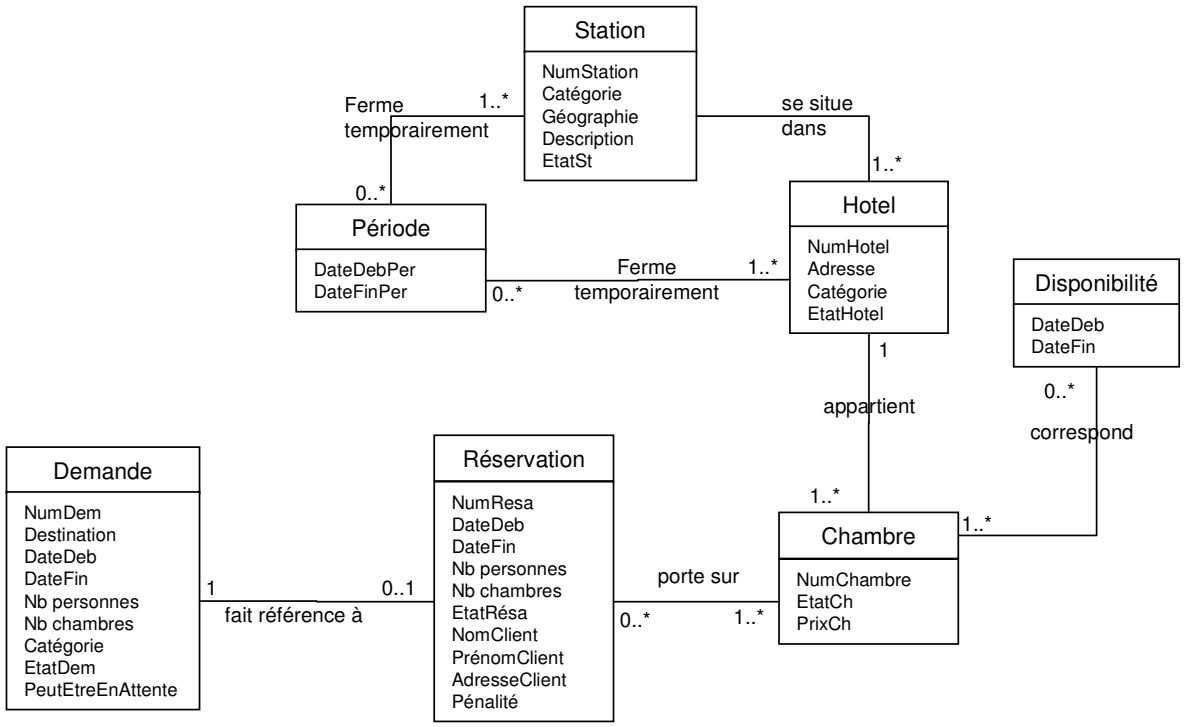


Figure 136 : Modèle de classes correspondant à la carte pivot Cu_{ab2}

Alors que toutes les sections de la carte pivot Cu sont blanches et considérées comme *totalement alignées*, la carte pivot Cuab2 contient des sections gris foncé précisant que

certaines sections des processus sont *partiellement alignées* avec celles du système. Ceci montre l'importance d'une étude de l'alignement à différents niveaux d'abstraction.

De plus, la carte pivot Cuab2 est conforme aux différentes mesures calculées au chapitre 4. En effet, cette carte pivot montre que certaines activités comme la fermeture temporaire d'hôtel ou de station, la modification de chambres ou la modification de réservations sont présentes dans les processus sans être prises en charge par le système. Ceci provient en partie du fait que les concepts de Chambre et de Station n'existent pas dans le système.

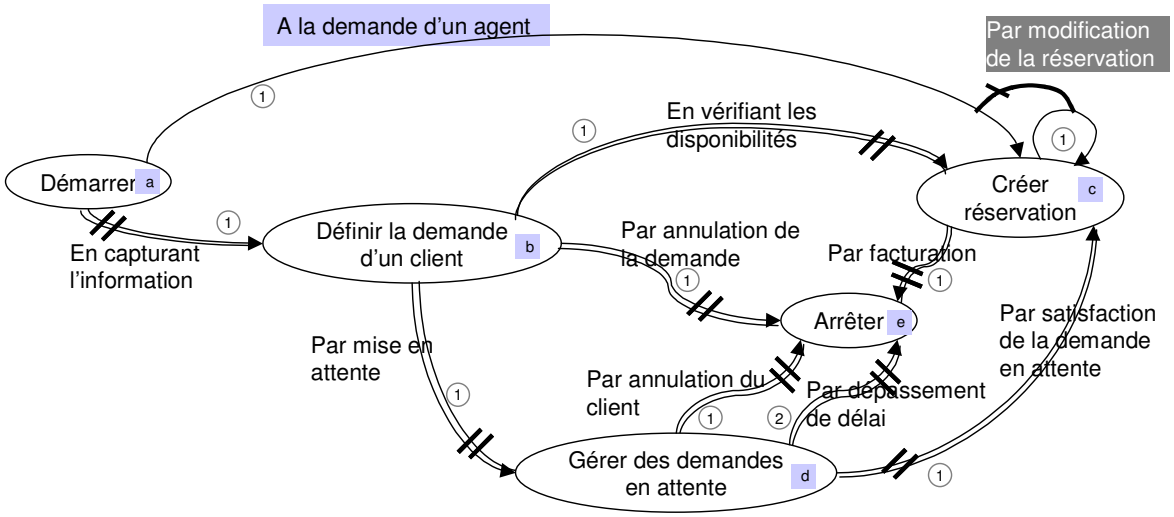
4.3.3 Construction de la carte pivot affinant la section bc1

De façon analogue, la construction de la carte pivot Cu_{bc1} se fait en fusionnant les cartes Cs_{bc1} et Cp_{bc1}. La carte Cp_{bc1} est considérée comme référent.

La carte Cp_{bc1} et la carte Cs_{bc1} comportent toutes deux cinq intentions. L'étude des intentions de la carte Cp_{bc1} montre que, pour chacune d'entre elles, il existe une intention de la carte Cs_{bc1} ayant le même nom et la même sémantique. Toutes ces intentions sont donc fusionnées et coloriées en blanc.

Concernant les sections, on remarque qu'il n'existe pas, dans la carte système Cs_{bc1}, de section identique à la section <*Créer réservation*, *Créer réservation*, *Par modification de la réservation*>. Chacune des autres sections de la carte Cp_{bc1} est identique à une section de la carte Cs_{bc1}. Une étude de la carte Cs_{bc1} montre que chaque section est identique à une section de la carte Cp_{bc1} hormis la section <*Démarrer*, *Créer réservation*, *A la demande d'un agent*>. Celle-ci est donc ajoutée dans la carte pivot et coloriée en gris clair comme l'indique la directive $DRI_{ab1.5.1.2.3}$. Chaque section coloriée en blanc correspond à une section système et une section processus identiques. On leur attribue donc le degré d'alignement totalement aligné. Concernant la section <*Créer réservation*, *Créer réservation*, *Par modification de la réservation*> coloriée en gris foncé, une étude montre qu'elle est partiellement alignée avec la section <*Démarrer*, *Créer réservation*, *A la demande d'un agent*> coloriée en gris clair. En effet, ces deux sections (i) permettent respectivement aux agents de modifier une réservation existante et d'en créer une nouvelle, mais (ii) correspondent en fait à la gestion des erreurs (dues en particulier à la fermeture d'hôtels, à la gestion des imprévus…).

La carte pivot affinant la section bc1 est présentée en Figure 137.



Cu_{bc1} : <Offrir Produit, Gérer les réservations,
En offrant des facilités de réservation >

Figure 137 : Carte affinant la section bc1

Les intentions sont décrites au Tableau 41.

Code	Intention	états
a	Démarrer	Station.état="ouverte", Hôtel.état="ouvert"
b	Définir la Demande d"un client	Demande.état = "créée"
d	Gérer des Demandes en attente	Demande.état="attente"
c	Créer Réservation	Demande.état="satisfaite", Réservation.état="créée", new.Disponibilité new.DisponibilitéHôtel
e	Arrêter	Réservation.état="facturée", Demande.état="annulée", Demande.état="sans_suite", Demande.état="hors delai"

Tableau 41 : Description des intentions de la carte pivot Cu_{bc1}

Les propriétés des sections sont spécifiées au Tableau 42.

Code	Pré-condition	Post-condition	Acteur	Ressource	Règle de gestion
ab1	Hôtel, Station	Demande.état = "créée"	Client	Hôtel	<-, Demande.état="créée">
de1	Demande.état="en attente" AND Demande.DateDeb > date-du-jour + 7j	Demande.état= "annulée"	Client	Hôtel	<Demande.état="attente", Demande.état="annulée">
bd1	Demande.état="créée" AND Demande.PeutEtreEnAttente = vrai AND (Disponibilité.DateDeb > Demande.DateDeb OR Disponibilité.DateFin < Demande.DateFin) (DisponibilitéHôtel.DateDeb > Demande.DateDeb OR DisponibilitéHôtel.DateFin < Demande.DateFin)	Demande.état="en attente"	Opérateur	Hôtel	<Demande.état="créée", Demande.état="attente">
be1	Demande.état="créée" AND (Demande.PeutEtreEnAttente = faux AND (Disponibilité.DateDeb > Demande.DateDeb OR Disponibilité.DateFin < Demande.DateFin))	Demande.état="sans_suite" Demande.état="annulée"	Opérateur	Hôtel	<Demande.état="créée", Demande.état="sans_suite"> <Demande.état="attente", Demande.état="annulée">
dc1	Demande.état="en attente" AND Disponibilité.DateDeb <= Demande.DateDeb AND Disponibilité.DateFin >= Demande.DateFin OR DisponibilitéHôtel.DateDeb <= Demande.DateDeb AND DisponibilitéHôtel.DateFin >= Demande.DateFin	Demande.état="satisfaite" AND Reservation.état="créée"	-	Hôtel	<Demande.état="en attente", Demande.état="satisfaite"> AND [<old.DisponibilitéHotel, new.DisponibilitéHôtel> OR <old.Disponibilité, new.Disponibilité>]
bc1	Demande.état="créée" AND Disponibilité.DateDeb <= Demande.DateDeb AND Disponibilité.DateFin >= Demande.DateFin OR DisponibilitéHôtel.DateDeb <= Demande.DateDeb AND DisponibilitéHôtel.DateFin >= Demande.DateFin	Demande.état="satisfaite" AND Reservation.état="créée"	Opérateur	Hôtel	<Demande.état="créée", Demande.état=satisfaite"> AND <-, Réservation.état="créée">
ce1	Réservation.état="créée"	Réservation.état="facturée"	-	Hôtel	<Réservation.état="créée", Réservation.état="facturée">
de2	Demande.état="en attente" AND Demande.DateDeb < date-du-jour + 7j	Demande.état="hors delai" Demande.état="annulée"	-	Hôtel	<Demande.état="attente", Demande.état="hors delai"> <Demande.état="attente", Demande.état="annulée">
cc1	Réservation.état="créée" OR Réservation.état="facturée"	<old.Réservation.Hôtel, new.Réservation.Hôtel> OR <old.Réservation.Chambre, new.Réservation.Chambre>	Opérateur	Hôtel	<old.Réservation.Hôtel, new.Réservation.Hôtel> OR <old.Réservation.Chambre, new.Réservation.Chambre>
ac1	Réservation.état="créée"	Réservation.état="créée"	Opérateur	Hôtel	<-, Réservation.état="créée">

Tableau 42 : Description des sections de la carte pivot Cu_{bc1}

La Figure 138 présente les classes utilisées par les sections et les intentions de la carte pivot Cu_{bc1}.

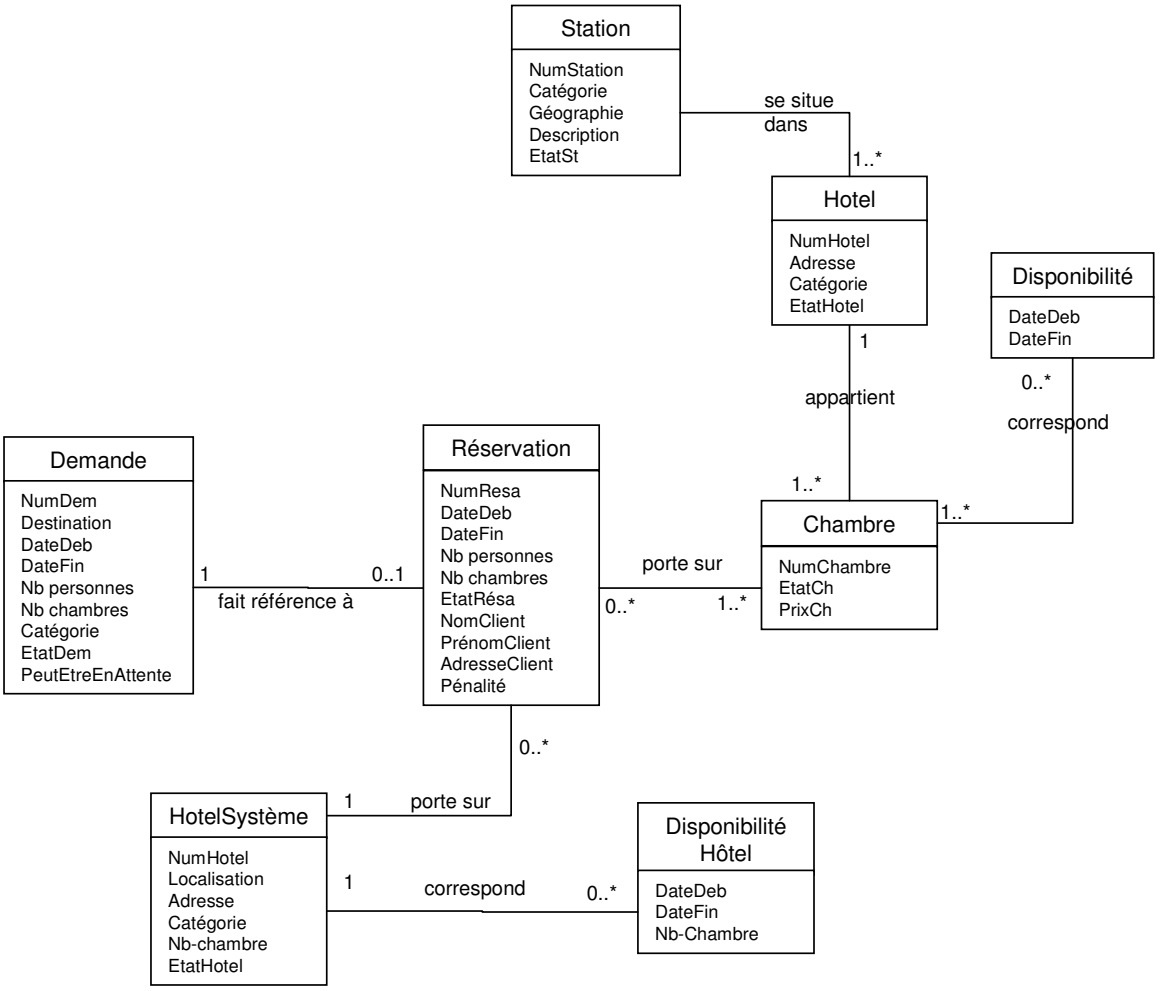


Figure 138 : Modèle de classes correspondant à la carte pivot Cu_{bc1}

La carte pivot Cubc1 montre que certaines activités des processus ne sont pas gérées par le système et vice-versa. C'est par exemple le cas de la modification d'une réservation qui est prévue dans les processus mais n'est pas implémentée dans le système. Le calcul des mesures au chapitre 4 avait déjà mis en évidence cette différence entre système et processus.

La carte pivot Cubc1 vient donc renforcer l'importance du concept d'affinement du méta-modèle pivot pour analyser la d'alignement à différents niveaux d'abstraction.

5. Découverte des exigences d'évolution

Le processus de la méthode ACEM propose trois façons différentes de découvrir des exigences d'évolution à partir :

- des dysfonctionnements du système ou des processus (DRI_4)
- des forces contextuelles (DRI_5)
- des ruptures de la relation d'alignement (DRI_6).

Chacune de ces possibilités nécessite un découpage des différentes cartes en blocs de base. Ainsi, avant de découvrir des exigences d'évolution selon chacun des trois cas, nous commençons par découper la carte de haut niveau du modèle pivot en blocs de base.

5.1. Découpage des cartes en blocs de base

Selon la méthode ACEM, un bloc de base de la carte pivot est constitué d'une intention et des différentes stratégies qui ont cette intention pour source ou pour cible. La carte pivot Cu compte quatre intentions. Elle se décompose donc en quatre blocs de base, comme le montre la Figure 139. Les classes associées à chaque bloc sont précisées dans des rectangles sous la partie de carte pivot correspondante.

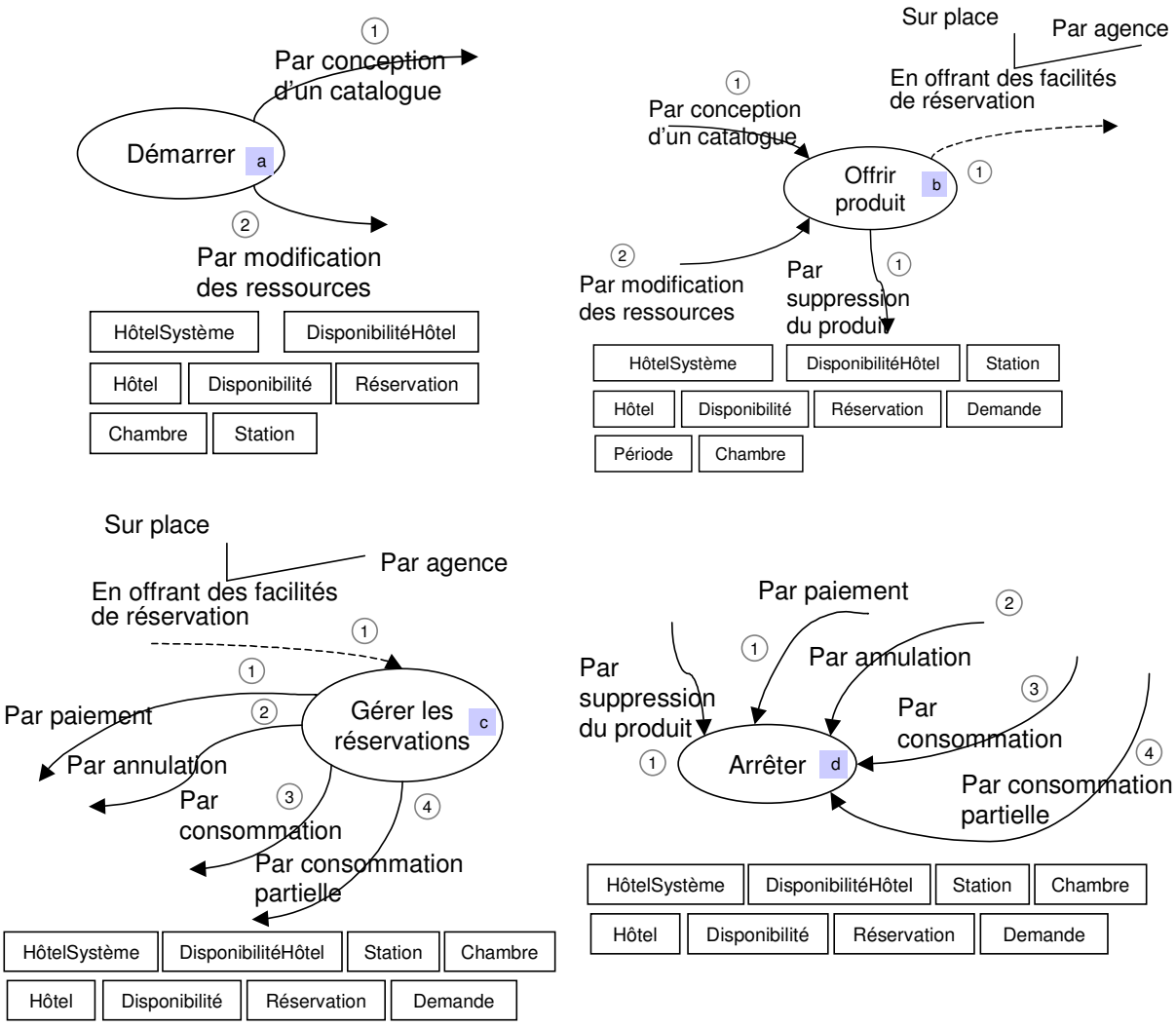


Figure 139 : Décomposition en blocs de base de la carte Cu

Comme la carte Cu est uniquement coloriée en blanc, chaque élément de chacun des blocs base est colorié en blanc.

Les autres cartes pivot sont décomposées en blocs de base de la même façon.

5.2. Découverte des exigences d'évolution

L'application des directives DRI_4, DRI_5 et DRI_6 de la méthode ACEM met en évidence des exigences d'évolution respectivement à partir (i) des dysfonctionnements du système ou des processus (ii) des forces contextuelles et (iii) des ruptures de la relation d'alignement.

Les trois sous-sections suivantes décrivent l'application de chacune de ces directives.

5.2.1 Découverte d'exigences d'évolution à partir des dysfonctionnements

Une analyse du système dans son utilisation normale montre que les disponibilités sont mal gérées par le système. En effet, les disponibilités sont gérées par hôtel et non par chambre. Ceci oblige les hôteliers à gérer manuellement la répartition des clients dans les différentes chambres de leur hôtel. La correction de ce dysfonctionnement du système permet, également, d'améliorer l'alignement.

Analyse de la correction du dysfonctionnement sur le bloc *Démarrer* de la carte Cu

Le bloc *Démarrer* de la carte pivot Cu définit les pré-conditions des différentes sections ayant pour source l'intention *Démarrer*. Aucune de ces pré-conditions ne correspond à la gestion des disponibilités. Le bloc *Démarrer* n'est donc pas affecté et ne fait pas l'objet d'exigence d'évolution relative à la correction de ce dysfonctionnement.

Analyse de la correction du dysfonctionnement sur le bloc *Offrir produit* de la carte Cu

L'analyse du bloc *Offrir produit* montre que pour corriger ce dysfonctionnement, il est nécessaire : (i) "d'introduire" le concept de chambre dans le système afin de tenir compte de cette nouvelle exigence et (ii) "d'améliorer" la gestion des disponibilités. Afin de mettre en œuvre ces deux types d'action, nous proposons d'appliquer les opérateurs comme suit :

- *AjouterClasse*(Hôtel) : permet d'ajouter la classe Chambre dans le modèle pivot, de la lier à la classe Hôtel et de préciser que le système manipule cette classe.
- *JoindreAttribut*(Chambre) permet d'ajouter les attributs NumChambre, EtatCh et PrixCh à la classe Chambre.
- *RemplacerClasse*(DisponibilitéHotel) : permet de substituer la classe Disponibilité à la classe DisponibilitéHôtel.
- *ChangerCible*(correspond) : permet de transformer le lien qui existe entre Disponibilité et HôtelSystème en un lien entre Disponibilité et Chambre.

L'application de ces opérateurs permet de corriger l'alignement entre le système et les processus. En effet, la classe Chambre ajoutée est en fait identique à la classe Chambre qui existait déjà dans le modèle pivot. Le résultat de l'opérateur *AjouterClasse* ne se matérialise donc pas concrètement par l'introduction d'une classe supplémentaire mais permet de préciser que le système et les processus manipulent la même classe.

- *EnleverAttribut*(HôtelSystème) : permet de supprimer l'attribut 'Nb-Chambre', devenu inutile, de la classe HôtelSystème.
- Les opérateurs *modifierPostCondition*(ab_1), *modifierPostCondition*(ab_2) et *modifierPréCondition*(bc_1) permettent de respectivement changer les post-conditions et les pré-conditions des sections $Cu.ab_1$, $Cu.ab_2$ et $Cu.bc_1$ afin de préciser que le système gère désormais des chambres et les disponibilités par chambre.
- *ModifierEtatIntention*(Offrir produit) : permet de modifier les états des classes qu'utilisent l'intention *Offrir produit*.

Analyse de la correction du dysfonctionnement sur le bloc *Gérer les réservations* de la carte Cu

Les exigences d'évolution découvertes sur les éléments du bloc *Offrir produit* ont un impact sur ceux du bloc *Gérer les réservations*.

- *ModifierPostCondition*(bd_1), *ModifierPostCondition*(cd_2), *ModifierPostCondition*(cd_4), permet de changer les post-conditions des différentes sections afin de tenir compte des modifications correspondant aux exigences d'évolution préalablement découvertes.
- *ModifierEtatIntention*(Gérer les réservations) permet de modifier les états des classes qu'utilisent l'intention *Gérer les réservations*.

Il est clair que ces exigences d'évolution ont un impact aux niveaux d'affinement supérieurs. Il est donc nécessaire d'étudier les différentes cartes pivot affinant les sections modifiées afin de découvrir de nouvelles exigences (cf. Annexe 2).

La Figure 140 présente le modèle de classes associé à la carte pivot Cu après application des opérateurs.

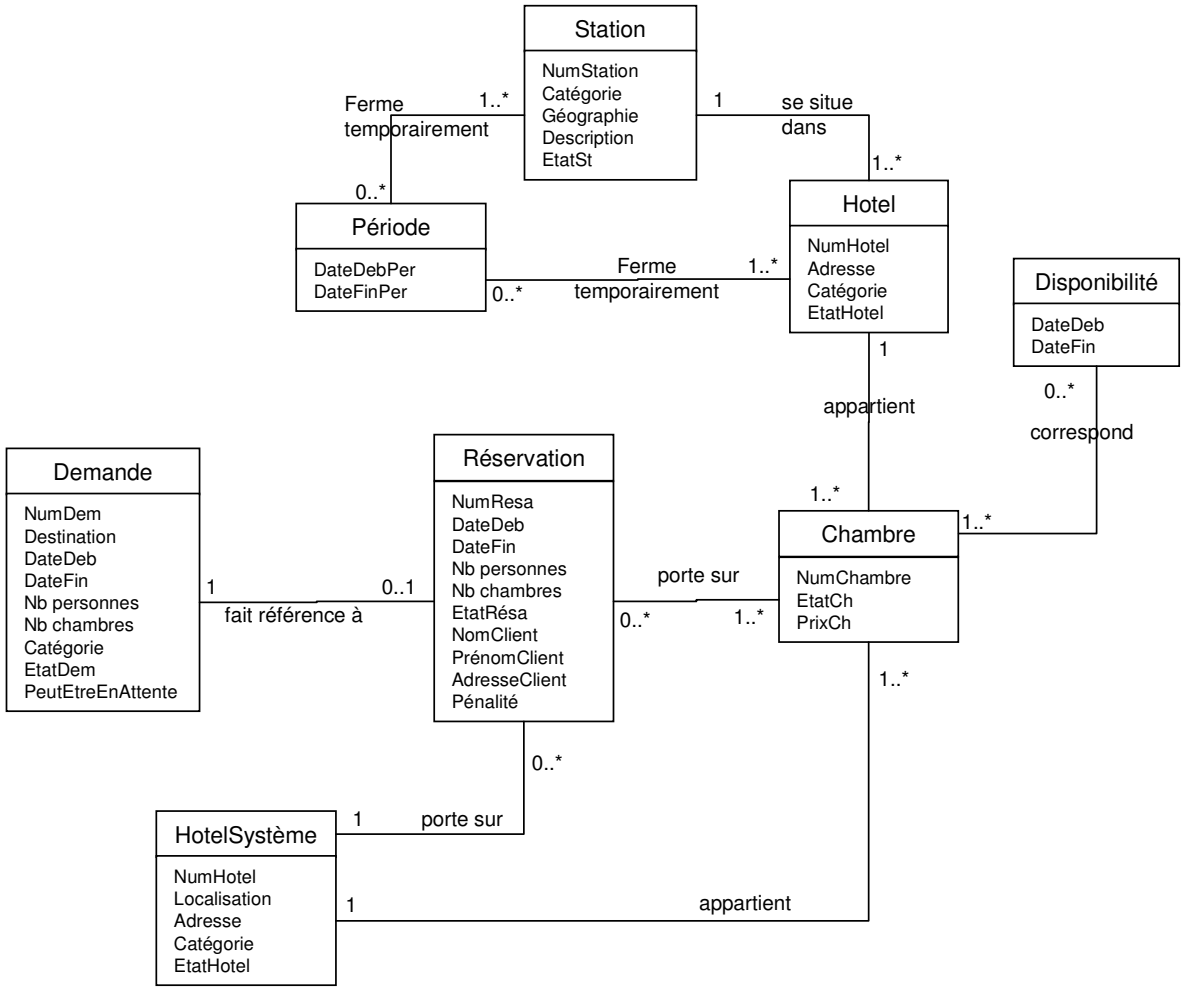


Figure 140 : Modèle de classes associé à la carte pivot Cu après application des opérateurs

Concernant la partie intentionnelle, les évolutions ne sont pas visibles graphiquement dans la mesure où elles n'affectent que les propriétés des sections et des intentions.

5.2.2 Découverte des exigences d'évolution par analyse de la relation d'alignement

La DRI_6 de la méthode ACEM propose d'identifier les ruptures d'alignement de deux façons différentes : en s'appuyant sur les métriques ou en analysant le modèle pivot. Dans la mesure où les métriques ont été appliquées sur les modèles As-Is du système et des processus, nous les utilisons pour mettre en évidence un certain nombre de ruptures dans la relation d'alignement.

Bien que l'analyse des dysfonctionnements du système concernant la gestion des disponibilités ait permis d'améliorer l'alignement, il reste cependant de nombreuses disparités entre le système et les processus. Par exemple, l'activité de modification des caractéristiques des chambres d'hôtel n'est pas prise en charge par le système. L'ajout du concept de chambre faisant partie des exigences d'évolution du système, il est désormais plus facile de corriger cette rupture de l'alignement. Pour cela, on choisit le type d'action "introduire" dans la mesure où, en terme intentionnel, le besoin est complètement nouveau pour le système.

La modification des caractéristiques des chambres n'étant pas visible dans la carte pivot Cu, il est nécessaire d'étudier les cartes affinant les différentes sections de la carte Cu. En particulier, l'analyse du bloc *Gérer les ressources hôtelières* de la carte pivot Cu_{ab2} permet de choisir l'opérateur *Ajouter* afin d'exprimer le besoin d'insérer dans la carte la stratégie *Par modification de chambres*. Cette stratégie existe dans la carte pivot mais est coloriée en gris foncé précisant qu'elle est propre aux processus. Son introduction dans la carte système permet de colorer cette stratégie en blanc, signe qu'elle existe dans le système et dans les processus. L'introduction de cette stratégie a un impact sur la post-condition de la section à laquelle elle appartient ainsi que sur les états de l'intention *Gérer les ressources hôtelières*. L'ajout de cette stratégie n'a en revanche aucun impact sur le bloc *Démarrer* de la carte Cu_{ab2}.

L'emploi de la directive DRI_8 permet de guider l'analyse de l'impact de cette exigence d'évolution sur le reste de la carte. Cette analyse aboutit à la création d'une nouvelle stratégie d'arrêt pour terminer le processus de gestion des ressources hôtelières. Comme le montre la section bc2 de la carte présentée en Figure 141, il est nécessaire de pouvoir terminer le processus normalement lorsque la modification des caractéristiques de la chambre n'a aucun impact sur les disponibilités ou les réservations en cours. Cette stratégie existe déjà dans la représentation des processus. L'application de l'opérateur *AjouterStratégie* se traduit par la coloration en blanc de la stratégie *Normalement*.

La Figure 141 présente la carte Cu_{ab2} après application des opérateurs *AjouterStratégie* pour ajouter les stratégies *Par modification de chambres* et *Normalement*.

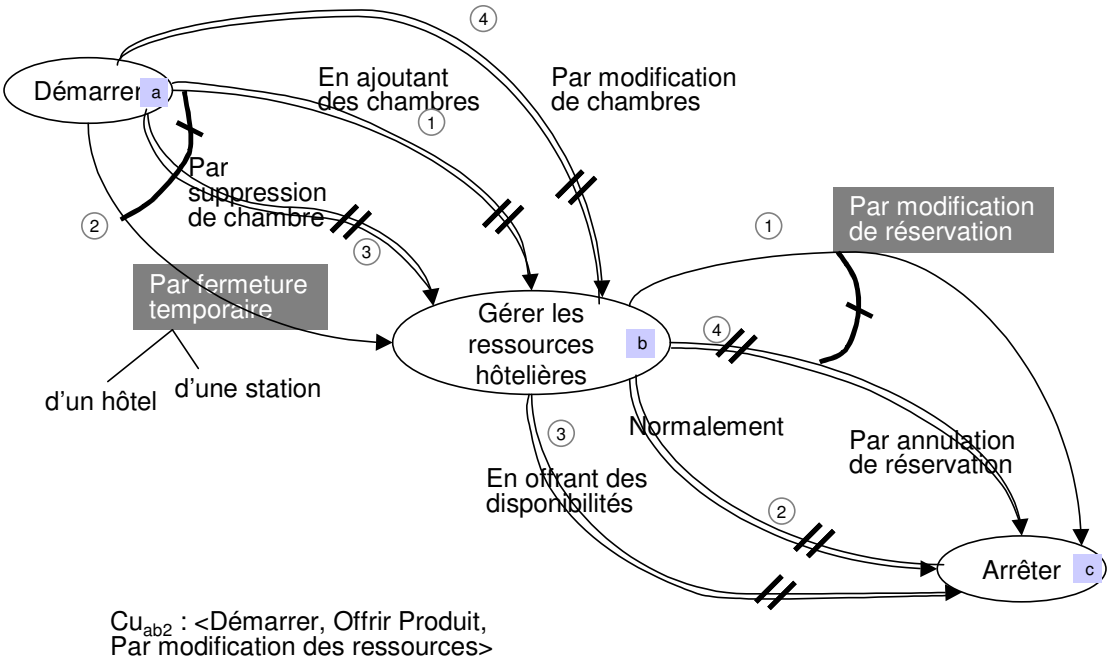


Figure 141 : Carte Cu_{ab2} après application des opérateurs

5.2.3 Découverte des exigences d'évolution à partir des forces contextuelles

La directive DRI_5 de la méthode ACEM guide la découverte des exigences d'évolution par analyse des forces contextuelles. Cette directive est décrite comme un plan composé de deux sous-directives pour (i) définir les forces contextuelles et (ii) spécifier des exigences d'évolution. La réalisation de chacune de ces sous-directives est décrite dans la suite de cette section.

5.2.3.1 Définition des forces contextuelles

Les stations de ski françaises sont en compétition avec de nombreuses stations étrangères et plus particulièrement depuis l'élargissement de l'Union Européenne. Les hôteliers français souhaitent être davantage compétitifs (i) en offrant par exemple des produits plus complexes sous forme de packages incluant des activités touristiques et culturelles et (ii) en fidélisant le client.

La fidélisation des clients passe aussi par leur satisfaction. Or une étude statistique menée auprès d'un échantillon représentatif de 1000 clients montre que seuls 13% sont prêts à attendre lorsque leur demande n'est pas immédiatement satisfaite (mettre leur demande en attente) et que près de 60% des clients ayant eu l'expérience de la mise en attente de leurs demandes ne sont pas satisfaits. Une analyse plus poussée montre que les clients qui ont accepté de mettre leurs demandes en attente pensaient que celles-ci seraient satisfaites à terme. En fait, le taux de satisfaction des demandes en attente est très faible car il repose uniquement sur les annulations des réservations. Les dirigeants du groupement d'hôtels ont décidé de supprimer ce service qui donne une image négative et de le remplacer par la mise en vente promotionnelle des disponibilités créées par les réservations annulées.

L'arbre de la Figure 142 présente la décomposition des forces contextuelles sous forme de buts et de sous-buts (ces buts sont ensuite nommés buts contextuels). Cette décomposition résulte d'une discussion avec les différentes parties prenantes, d'entretiens menés avec des utilisateurs, et d'enquêtes de satisfaction menées auprès des clients. Le choix qui a été fait ici est celui du groupement d'hôtels, mais bien sur, d'autres décompositions sont possibles.

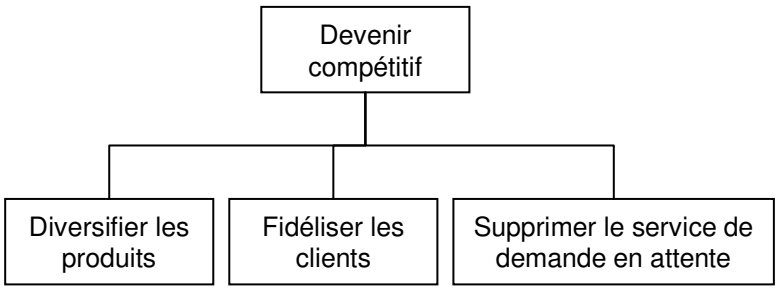


Figure 142 : Arbre des forces contextuelles de changement

5.2.3.2 Spécification des exigences d'évolution

Comme l'indique la Figure 122, la méthode ACEM préconise d'analyser les forces contextuelles pour découvrir les exigences d'évolution. Cette analyse prend la forme d'une étude systématique de l'impact des forces contextuelles sur les différents blocs de base des cartes unifiées. Cette analyse commence par les quatre blocs de la carte Cu présentés en Figure 139, à savoir le bloc *Démarrer*, le bloc *Offrir produit*, le bloc *Gérer les réservations*, et le bloc *Arrêter*. Ces quatre analyses d'impact sont présentées tour à tour ci après.

Analyse de l'impact des forces contextuelles sur le bloc *Démarrer* de la carte Cu

Le bloc *Démarrer* de la carte définit les pré-conditions des différentes sections ayant pour source l'intention *Démarrer*. Aucune de ces pré-conditions n'est un obstacle à la mise en œuvre des forces contextuelles. Le bloc *Démarrer* n'est donc pas affecté et ne fait pas l'objet d'exigence d'évolution relative aux forces contextuelles.

Analyse de l'impact des forces contextuelles sur le bloc *Offrir produit* de la carte Cu

L'analyse de l'impact des forces contextuelles sur les éléments du bloc *Offrir produit* fait apparaître un besoin d'évolution sous la pression de la force « Diversifier les produits ». Il faut donc choisir une évolution des éléments de ce bloc d'un autre type que "maintenir".

En l'occurrence, il est décidé "d'améliorer" l'intention *Offrir produit* sans pour autant la modifier drastiquement. Parmi la collection d'opérateurs d'écarts permettant d'exprimer le besoin de changement, l'opérateur *Remplacer* est choisi. Il a été décidé de substituer l'intention *Offrir des packages* à l'intention *Offrir produit*. L'idée sous-jacente est pour les hôtels de proposer dorénavant des packages incluant des activités touristiques en plus des chambres. Ceci requiert de développer des partenariats, par exemple avec des restaurant, des cinémas, des magasins de sport et toute autre entreprise offrant des services susceptibles d'être intégrés aux packages. Ce changement a donc des répercussions sur des cartes de plus bas niveau en particulier sur celles affinant les sections ayant pour source ou pour cible *Offrir des packages*.

La modification de l'intention a des répercussions sur le reste des éléments du bloc en particulier sur les pré et post-conditions des sections ayant l'intention *Offrir des packages*

comme source ou comme cible. Il est donc nécessaire de modifier au moins les pré et post-conditions de ces sections. Ces exigences seront exprimées en utilisant les opérateurs *Modifier*, *Joindre* ou *Enlever*.

Aucun changement de la structure du produit n'est requis pour la section Cu.bc1 dans la mesure où la structure du produit n'interagit avec la façon dont il est vendu. En revanche, concernant les sections Cu.ab1, Cu.ab2 et Cu.bd1, la modification des pré et post-conditions ne suffit pas. En effet, la manière d'introduire les produits et les packages dans le catalogue doit changer dans la mesure où cela peut maintenant impliquer des partenariats avec d'autres acteurs ou fournisseurs appartenant à d'autres secteurs que l'hôtellerie. La même remarque s'applique pour les sections de modification et de suppression de packages du catalogue. En conclusion de cette analyse, les stratégies *Par conception d'un catalogue*, *Par modification des ressources* et *Par suppression du produit* doivent être respectivement remplacées par les stratégies *Par conception de package*, *Par gestion des ressources* et *Par suppression d'un package*. Ces nouvelles exigences d'évolution sont exprimées à l'aide de l'opérateur *RemplacerStratégie*.

La modification de l'intention *Offrir produit* a également des conséquences sur les objets manipulés. Ainsi, les classes Package, Activité et DisponibilitéActivité sont introduites afin de pouvoir dorénavant gérer l'enregistrement des packages proposés aux clients. La classe Disponibilité est renommée en DisponibilitéChambre, afin de ne pas confondre les disponibilités des activités et celles des chambres.

Le Tableau 43 résume les nouvelles exigences d'évolution découvertes par analyse de l'impact des forces contextuelles de changement sur les éléments du bloc *Offrir produit* de la carte Cu.

Code	Opérateur	Elément As-Is	Elément To-Be
Cu-1	*RemplacerIntention*	*Offrir produit*	*Offrir des packages*
Cu-2	*RemplacerStratégie*	*Par conception d'un catalogue*	*Par conception de package*
Cu-3	*RemplacerStratégie*	*Par modification des ressources*	*Par gestion des ressources*
Cu-4	*RemplacerStratégie*	*Par suppression du produit*	*Par suppression d'un package*
Cu-5	*AjouterClasse*	-	*Package*
Cu-6	*AjouterClasse*	-	*Activité*
Cu-7	*JoindrePrécondition*	-	*Package*
Cu-8	*JoindrePrécondition*	-	*Activité*
Cu-9	*AjouterClasse*	-	*DisponibilitéActivité*
Cu-10	*RenommerClasse*	*Disponibilité*	*DisponibilitéChambre*

Tableau 43 : Exigences d'évolution découvertes sur le bloc *Offrir produit*

Analyse de l'impact des forces contextuelles sur le bloc *Gérer les réservations* de la carte Cu

Le Tableau 44 présente les exigences d'évolution découvertes par analyse de l'impact des forces contextuelles de changement sur les éléments du bloc *Gérer les réservations*.

De manière globale, le changement engendré par action de la force contextuelle « Fidéliser les clients » sur l'intention *Gérer les réservations* est de type "étendre". Il est choisi d'exprimer cette exigence d'évolution en remplaçant cette intention par l'intention *Gérer la relation client* (en utilisant l'opérateur *RemplacerIntention*). Par ailleurs, l'analyse de l'impact de la force contextuelle « Diversifier les produits » sur l'intention *Gérer les réservations* conduit à "améliorer" la réalisation de cette intention. Ce nouvel écart est déjà exprimé par l'exigence

d'évolution précédente. Il faut cependant noter que des différences sont à prévoir pour ces deux exigences d'évolution lorsqu'elles seront exprimées à des niveaux d'abstraction inférieurs.

La force contextuelle « Supprimer le service de demande en attente » nécessite de remettre en cause l'intention *Gérer les réservations*. Cependant, l'impact de cette exigence n'est pas visible à ce niveau. Cela explique l'absence de changement correspondant à l'application de ce but contextuel sur cette intention.

Remplacer l'intention *Gérer les réservations* par *Gérer la relation client* a plusieurs conséquence :

- La cible des stratégies *Par annulation*, *Par consommation*, *Par consommation partielle* et *Par paiement* doit changer. Cette exigence est exprimée en utilisant l'opérateur *ChangerIntentionCible*. En effet, il est dorénavant entendu que ces quatre stratégies proposent des façons de gérer la relation client mais pas d'arrêter le processus.

- De nouvelles stratégies sont nécessaires pour terminer le processus. Une discussion avec les dirigeants et les utilisateurs a conduit à l'identification de deux manières alternatives de sortir de la relation client : soit par exclusion, soit à la demande du client. L'opérateur *AjouterStratégie* permet de spécifier la nécessité d'introduire cette alternative sous forme de stratégie.

- La gestion de la relation client implique avant tout d'avoir des clients. La stratégie *Par attraction de la clientèle* est donc ajoutée. Mais il faut aussi pouvoir garder les clients. Ceci peut se faire par exemple en adaptant les packages du catalogue afin de satisfaire de nouveaux besoins des clients. Ceci se traduit par l'ajout de la stratégie *Par marketing*. Une autre possibilité réside dans le fait de mieux gérer les informations concernant les clients. La stratégie *En gérant l'information sur les clients* est ajoutée pour exprimer cette nouvelle exigence.

- La nouvelle exigence de gestion de l'information client nécessite d'introduire une classe Client dans les classes du modèle pivot. L'introduction de cette classe a des répercussions sur la classe Réservation. En effet, la classe Réservation contenait plusieurs attributs concernant le client : NomClient, PrénomClient et AdresseClient. Ceux-ci sont introduits dans la classe Client qui est elle-même liée à la classe Réservation par une relation. Par ailleurs, la pénalité en cas d'annulation tardive d'une réservation ou d'une consommation partielle est dorénavant imputée au client et non plus à la réservation. Ainsi, l'attribut Pénalité est supprimé de la classe Réservation. Il est ajouté dans la classe Client.

- Finalement, la prise en compte de la force contextuelle « Fidéliser les clients » requiert (i) d'introduire la stratégie *En récompensant le client* qui permet d'offrir des points de fidélité au client et (ii) d'introduire la stratégie *Par paiement avec des points de fidélité*. Cette dernière forme un paquet avec la stratégie *Par paiement normalement* en utilisant l'opérateur *RetyperSegment*.

Code	Opérateur	Elément As-Is	Elément To-Be	But contextuel
Cu-11	*RemplacerIntention*	*Gérer les réservations*	*Gérer la relation client*	"Fidéliser les clients", "Diversifier les produits"
Cu-12	*ChangerIntentionSource*	*Par annulation*	*Par annulation*	
Cu-13	*ChangerIntentionSource*	*Par consommation*	*Par consommation*	
Cu-14	*ChangerIntentionSource*	*Par consommation partielle*	*Par consommation partielle*	
Cu-15	*ChangerIntentionSource*	*Par paiement*	*Par paiement*	
Cu-16	*AjouterStratégie*	-	*Par exclusion*	
Cu-17	*AjouterStratégie*	-	*A la demande du client*	
Cu-18	*AjouterStratégie*	-	*Par attraction de la clientèle*	"Fidéliser les clients"
Cu-19	*AjouterStratégie*	-	*Par marketing*	"Fidéliser les clients"
Cu-20	*AjouterStratégie*	-	*En gérant l'information sur les clients*	"Fidéliser les clients"
Cu-21	*AjouterStratégie*	-	*En récompensant le client*	"Fidéliser les clients"
Cu-22	*AjouterStratégie*	-	*Par paiement avec des points de fidélité*	"Fidéliser les clients"
Cu-23	*RetyperSegment*	*Par paiement avec des points de fidélité* et *Par paiement normalement*	*Par paiement avec des points de fidélité* et *Par paiement normalement*	"Fidéliser les clients"
Cu-24	*AjouterClasse*	-	*Client*	"Fidéliser les clients"
Cu-25	*EnleverAttribut*	*NomClient*	-	
Cu-26	*EnleverAttribut*	*PrénomClient*	-	
Cu-27	*EnleverAttribut*	*AdresseClient*	-	
Cu-28	*EnleverAttribut*	*Pénalité*	-	

Tableau 44 : Exigences d'évolution découvertes sur le bloc *Gérer les réservations*

Analyse de l'impact des forces contextuelles sur le bloc *Arrêter* de la carte Cu

Une étude du bloc *Arrêter* confirme que les deux stratégies *Par exclusion* et *A la demande du client* sont bien des stratégies d'arrêt. Aucune nouvelle exigence d'évolution n'est identifiée à leur égard.

La Figure 143 présente la carte Cu après application des opérateurs présentés dans les tableaux 43 et 44.

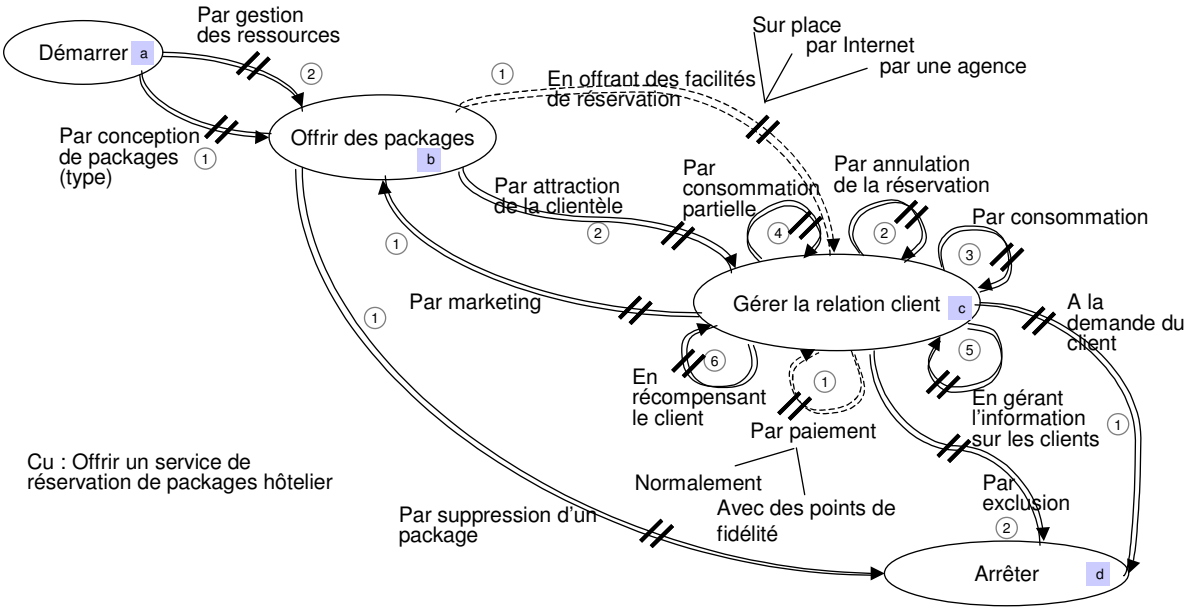


Figure 143 : Carte Cu après application des opérateurs

La Figure 144 présente le modèle de classes associé à la carte Cu. Ce modèle est celui obtenu par application des opérateurs matérialisant les exigences d'évolution qui viennent d'être découvertes par analyse d'impact des forces contextuelles de changement. Chaque fois que cela était possible, les exigences d'évolution ont été traduites de la même façon dans le système et les processus afin que l'alignement soit tout de même amélioré ou du moins que les exigences d'évolution sur le système et sur les processus ne soient pas divergentes.

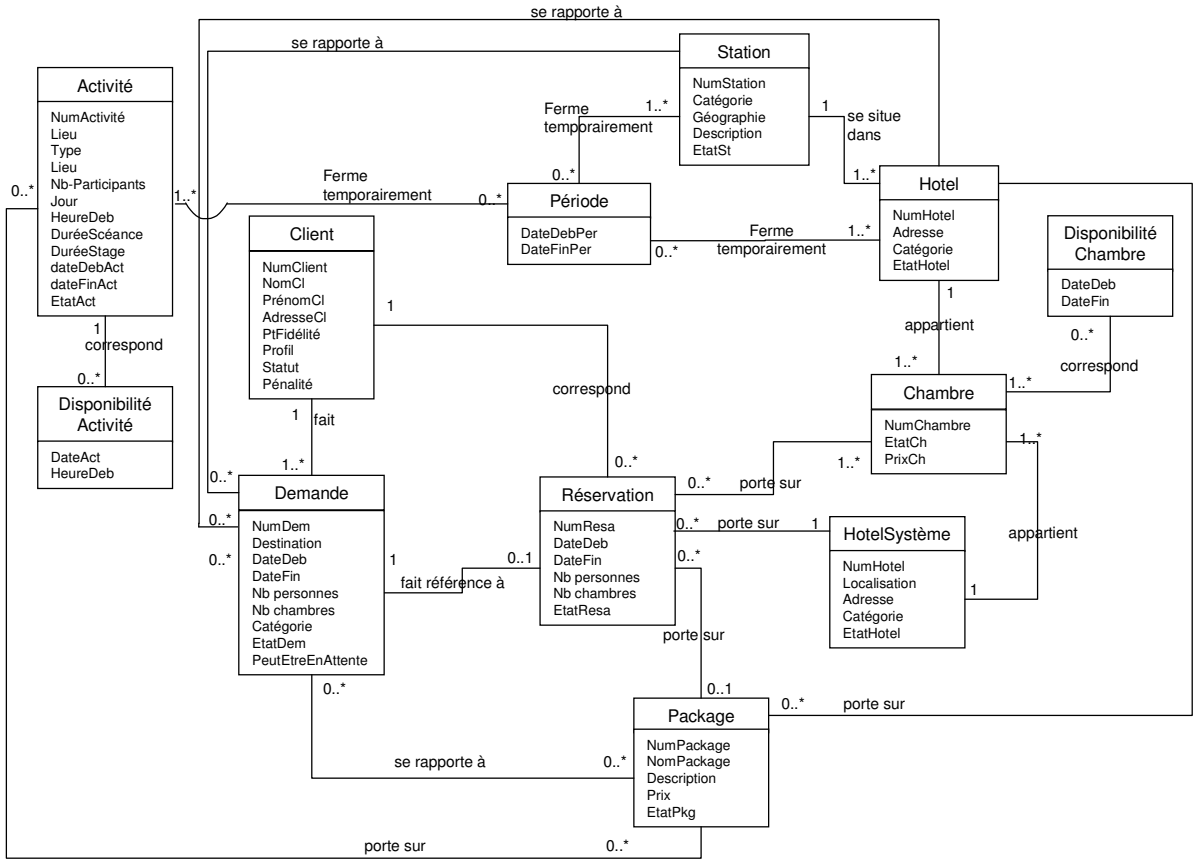


Figure 144 : Modèle de classes associé à la carte Cu

Une analyse analogue est appliquée aux cartes Cu_{ab2} et Cu_{bc1}. Cette analyse permet de mettre en évidence les exigences d'évolution présentées dans le Tableau 45

Carte affinant Cu.ab2			
Code	**Operateur**	**As-Is élément**	**To-Be élément**
Cu.ab2-1	*RemplacerIntention*	*Gérer les ressources hôtelières*	*Gérer les ressources*
Cu.ab2-2	*RemplacerStratégie*	*Cuab2.bc3 En offrant des Disponibilités*	*En offrant des Disponibilités de package*
Cu.ab2-3	*JoindrePrécondition*	Cu_{ab2}.bc1	OR Package.état="fermé"
Cu.ab2-4	*JoindrePrécondition*	Cu_{ab2}.bc2	OR Package.état="fermé"
Cu.ab2-5	*JoindrePrécondition*	Cu_{ab2}.ab2	Activité
Cu.ab2-6	*JoindrePostcondition*	Cu_{ab2}.ab2	Activité.état="fermée"
Carte affinant Cu.bc1			
Code	**Operateur**	**As-Is élément**	**To-Be élément**
Cu.bc1-1	*RemplacerIntention*	*Créer réservation*	*Réserver un Package*
Cu.bc1-2	*SupprimerIntention*	*Gérer des demandes en attente*	-
Cu.bc1-3	*AjouterStratégie*	*ab1*	*Par réutilisation*
Cu.bc1-4	*RetyperOR*	*En capturant l'information par une nouvelle demande et par réutilisation*	*En capturant l'information par une nouvelle demande et par réutilisation*
Cu.bc1-5	*SupprimerStratégie*	*de1 Par annulation du client*	-
Cu.bc1-6	*SupprimerStratégie*	*bc1 Par mise en attente*	-
Cu.bc1-7	*RenommerStratégie*	*bd1 Par annulation de la demande*	*Par abandon*
Cu.bc1-8	*SupprimerStratégie*	*dc1 Par satisfaction de la demande en attente*	-
Cu.bc1-9	*RemplacerStratégie*	*bc1 En vérifiant les disponibilités*	*Par sélection dans une liste d'offre*
Cu.bc1-10	*SupprimerStratégie*	*de2 Par dépassement de délai*	-
Cu.bc1-11	*AjouterStratégie*	-	*C_{bc1}.bb1 Par contreproposition*
Cu.bc1-12	*AjouterStratégie*	-	*C_{bc1}.ac1 Par sélection guidée*
Cu.bc1-13	*AjouterStratégie*	-	*C_{bc1}.bb2 Par affinement*
Cu.bc1-14	*AjouterStratégie*	-	*C_{bc1}.bb3 Par formulation d'une nouvelle demande*
Cu.bc1-15	*SupprimerStratégie*	*ac1 A la demande d'un agent*	-
Cu.bc1-16	*AjouterStratégie*	-	*cc1 Par modification de la réservation*
Cu.bc1-17	*ModifierPrécondition*	Cu_{bc1}.bc1	DisponibilitéChambre.DateDeb <= Demande.DateDeb
Cu.bc1-18	*ModifierPrécondition*	Cu_{bc1}.bc1	AND DisponibilitéChambre.DateFin >= Demande.DateFin
Cu.bc1-19	*ModifierPostcondition*	Cu_{bc1}.bc1	new.DisponibilitéChambre, new.DisponibilitéActivité

Tableau 45 : Exigences d'évolution découvertes sur les cartes Cu_{ab2} et Cu_{bc1}

La méthode ACEM propose de répercuter les exigences d'évolution identifiées par analyse de l'alignement, par analyse des dysfonctionnements, par analyse des forces contextuelles et par analyse d'impact, sur les modèles de système et de processus. L'application des deux stratégies de propagation est présentée dans la prochaine section qui achève l'illustration de l'application de la méthode.

6. Répercussion des exigences d'évolution sur les modèles du système et des processus

La section précédente a illustré la découverte d'un certain nombre d'exigences d'évolution. Celles-ci peuvent être classées en trois catégories :

- Les exigences d'évolution qui affectent des parties propres au processus. Ces exigences peuvent corriger l'alignement en modifiant les processus ou résoudre des dysfonctionnements propres aux processus ou correspondre à de nouveaux besoins. Elles impactent les parties représentées en gris foncé dans le modèle pivot.

- Les exigences d'évolution qui sont appliquées sur des éléments présents uniquement dans le système. Ces exigences corrigent une rupture de l'alignement en modifiant le système, atténuent les dysfonctionnements propres au système ou traduisent de nouveaux besoins. Elles impactent les parties colorées en gris clair dans le modèle pivot.

- Les exigences d'évolution qui affectent les parties communes au système et aux processus. Elles impactent les parties documentées en blanc dans le modèle pivot.

Le positionnement des exigences dans chacune de ces catégories est utile pour construire les modèles futurs (To-Be) du système et des processus. En effet, la méthode ACEM préconise de construire le modèle de processus To-Be par répercussion des exigences d'évolution appartenant à la première et à la troisième catégorie. Par ailleurs, la méthode ACEM propose de répercuter les exigences d'évolution de la deuxième catégorie et celles de la troisième catégorie pour produire le modèle du futur système. L'application des directives proposées par la méthode ACEM est illustrée dans chacune des deux sections ci-dessous.

6.1. Construction du modèle de processus To-Be

La construction du modèle de processus To-Be se fait en propageant les exigences d'évolution sur le modèle de processus As-Is. Comme indiqué plus haut, seules doivent être prises en compte les exigences d'évolution qui ont un impact sur les processus, c'est-à-dire coloriées en gris foncé et en blanc. Dans l'exemple, aucun dysfonctionnement propre aux processus n'a été pris en compte pour définir des exigences d'évolution. De plus, dans toutes les situations de rupture d'alignement, il a toujours été choisi de transformer le système plutôt que les processus. Par conséquent, les seules exigences de notre exemple à répercuter sur le modèle de processus sont celles qui ont été mises en œuvre sur la partie blanche du modèle pivot.

Ces exigences sont les suivantes :

1. Toutes les activités concernant la gestion des demandes en attente doivent être supprimées.

2. L'activité de création de réservation permet désormais de créer des réservations sur des packages, c'est-à-dire des produits complexes comprenant une ou plusieurs chambres d'hôtel ainsi que des activités touristiques.

3. Les informations concernant le client sont gérées dans la base de données du groupement d'hôtel, permettant ainsi au client d'être reconnu lorsqu'il fait de nouvelles demandes.

4. Le client est récompensé quand il consomme un produit proposé par le groupement d'hôtels.

La Figure 145 présente le modèle de processus To-Be de création de réservation modélisé à l'aide de diagrammes d'activités UML. Ce modèle de processus est produit en répercutant chacune des exigences d'évolution mise en œuvre sur les cartes Cu et Cu_{bc1}, concernant la création d'une réservation.

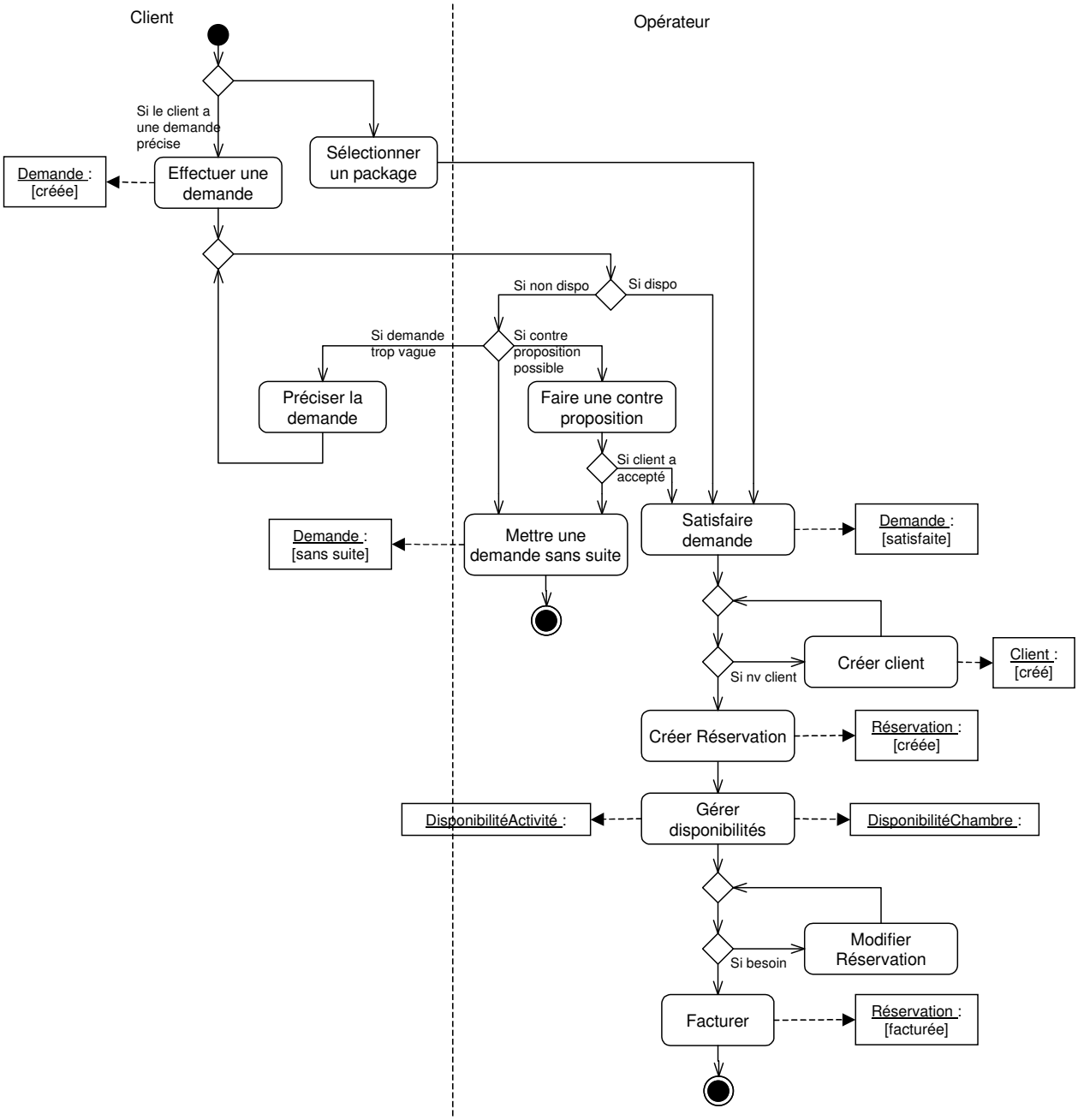


Figure 145 : Processus To-Be de création de réservation

On remarque que la structure de ce processus est globalement identique à celle du processus As-Is décrit en Figure 35. Quelques différences apparaissent cependant. Certaines activités ont ainsi été supprimées, par exemple “Mettre en attente”, “Mettre hors délai”, “Annuler Demande”. D’autres activités ont par ailleurs été ajoutées à partir des autres exigences d’évolution. Par exemple, l’ajout de l’activité “Sélectionner package“ permet désormais au client de choisir parmi les offres disponibles.

6.2. Construction du modèle de système To-Be

L’analyse des exigences d’évolution montre que certaines d’entre elles ont un impact uniquement sur le système. Dans notre exemple, il s’agit des exigences d’évolution qui viennent réparer des dysfonctionnements du système concernant la gestion des disponibilités

et celles qui ont été définies pour corriger l'alignement. La construction du modèle de système To-Be est donc faite par répercussion des exigences d'évolution impactant les parties en blanc et en gris clair du modèle pivot.

La Figure 146 présente un schéma de classes O* To-Be mettant en œuvre les exigences d'évolution identifiées précédemment. Dans ce nouveau schéma de classes, les classes Chambre et DisponibilitéChambre sont définies pour résoudre le dysfonctionnement du système relatif à la mauvaise gestion des disponibilités, telles que le demandent les exigences d'évolution correspondantes. L'introduction des classes Package, Activité et DisponibilitéActivité dans le schéma est faite par répercussion des exigences d'évolution définies par la volonté des dirigeants du groupement d'hôtels de proposer des produits complexes. Enfin, l'ajout de la classe Client permet de gérer dans le système les informations relatives au client, comme défini dans les exigences d'évolution qui ont amené à la création d'une gestion de la relation client.

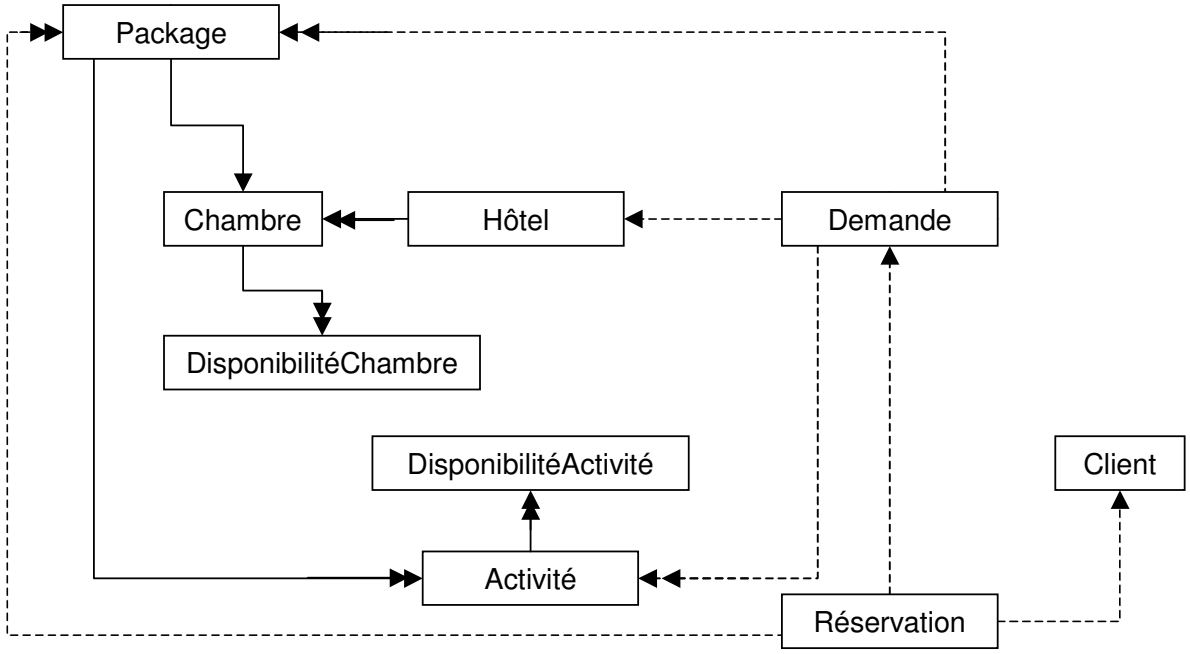


Figure 146 : Schéma de classes O* To-Be

Les exigences d'évolution découvertes sur le modèle pivot ont également des répercutions sur la partie dynamique du modèle O*. Ainsi, la Figure 147 présente l'événement EV3 du modèle de système To-Be. Cet événement correspond à la gestion d'une demande. Comme dans le modèle As-Is, la transaction spécifiée par cet événement intègre la satisfaction d'une demande (opération satisfaire sur la classe objet Demande) et la création d'une réservation (opération créer sur la classe Réservation) si des disponibilités le permettent et le classement de la demande sans suite dans le cas contraire. Cependant, cet événement ne fait plus appel à l'opération « réserver » de la classe Hôtel, mais à celle associée à classe Package. Ce changement implémente l'exigence de modification de la structure des produits (Cu-1). Dans cette nouvelle spécification, l'opération réserver de la classe Package déclenche les opérations créer et supprimer sur les classes DisponibilitéChambre et DisponibilitéActivité.

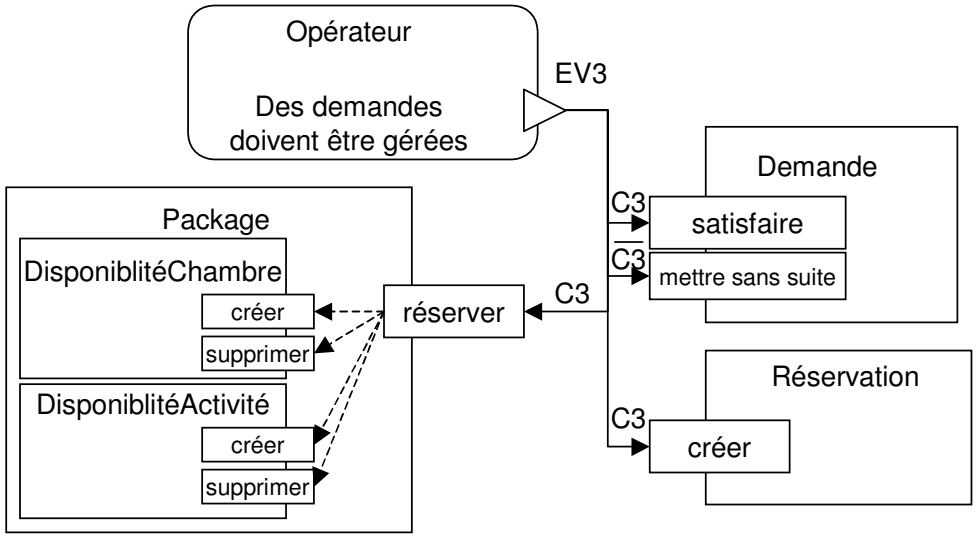


Figure 147 : Description de l'événement EV3 du modèle de système To-Be

Les autres événements du modèle de système To-Be sont obtenus de façon analogue en répercutant chaque exigence d'évolution sur le modèle de système.

7. Conclusion

Ce chapitre a présenté l'application de la méthode ACEM à l'étude de cas de la gestion des réservations d'hôtel. L'application a permis d'illustrer la correction et l'évolution de l'alignement entre le système d'information et les processus d'entreprise.

Une analyse rétrospective de cette application a permis de tirer les leçons suivantes :

- L'utilisation du modèle de la carte pivot permet de se focaliser sur l'essence du système et des processus tout en mettant en évidence les objets système et les objets de gestion respectivement utilisés par l'un et par l'autre.
- Le mécanisme d'abstraction/affinement permet (i) de modéliser de façon globale le système et les processus, (ii) de ne détailler l'analyse que des parties pour lesquelles cela s'avère nécessaire et (iii) d'étudier la relation d'alignement à plusieurs niveaux d'abstraction. Ainsi, certaines sections peuvent paraître totalement alignées à un niveau d'abstraction élevé mais être affinées par des cartes comportant certaines sections partiellement ou pas du tout alignées.
- Le mécanisme de coloration permet de représenter dans un même modèle les processus et le système, même si ceux-ci ne sont pas parfaitement alignés.
- La spécification des exigences d'évolution sous forme d'écarts permet de se concentrer sur les parties du système et des processus qui évoluent sans avoir à redéfinir les parties qui restent inchangées.
- La spécification des exigences d'évolution, contrairement à la stratégie de modification d'un cahier des charges permet de définir préalablement, et de manière explicite les évolutions requises, sans que celles-ci doivent nécessairement être adoptées de manière immédiate. La modélisation explicite des exigences d'évolution a pour autre effet

bénéfique de permettre des analyses complexes, comme l'analyse d'impact ou l'analyse des répercussions sur les modèles conceptuels.

- La méthode ACEM guide une analyse systématique des dysfonctionnements, des ruptures de l'alignement et des forces contextuelles. Sa mise en œuvre permet de corriger l'alignement sans forcément aboutir à un alignement parfait.

CHAPITRE 10 : CONCLUSION

1. Contributions

Dans cette thèse, parmi les différents aspects de l'ingénierie de l'alignement, nous avons abordé plus particulièrement l'évaluation, la correction et l'évolution de l'alignement entre un système d'information et des processus d'entreprise.

Nous avons tout d'abord proposé (i) une définition formelle de l'alignement entre un système d'information et des processus d'entreprise et (ii) un ensemble de métriques pour mesurer l'alignement. Nous avons ensuite proposé une démarche intentionnelle, la méthode ACEM (**A**lignment **C**orrection and **E**volution **M**ethod) qui guide la correction et l'évolution de l'alignement.

La *définition* formelle de la relation d'alignement que nous proposons repose sur l'utilisation de modèles pour représenter le système et les processus. Afin d'être indépendante des modèles utilisés, cette définition s'appuie sur les concepts de deux méta-modèles génériques ainsi que sur deux types de lien *représente* et *correspond*. Le lien *correspond* exprime une égalité entre des concepts similaires de modèles différents. Le lien *représente* précise qu'un concept d'un modèle a un impact sur un concept de l'autre modèle. Nous définissons la relation d'alignement comme l'ensemble des liens *correspond* ou *représente* entre un élément du modèle de système et un élément du modèle de processus.

Pour appréhender la complexité de la relation d'alignement, nous avons défini dix *métriques* qui mesurent chacune un aspect différent mais complémentaire de l'alignement. Ces métriques permettent d'évaluer le degré d'alignement pour un projet donné, à un instant précis. Elles sont définies au niveau générique en utilisant les concepts des méta-modèles BPRAM et SRAM et les liens *correspond* et *représente* puis instanciées pour deux méta-modèles spécifiques.

La méthode ACEM utilise un *modèle pivot* intentionnel pour représenter conjointement le système d'information et les processus d'entreprise. Ainsi, système et processus sont représentés dans les mêmes termes ce qui permet de résoudre le problème de discordance conceptuelle. Le modèle pivot permet de représenter les buts de l'organisation et les buts que permettent d'atteindre les fonctionnalités du système tout en précisant les objets manipulés par le système et les processus d'entreprise. Les concepts centraux du méta-modèle pivot sont ceux *d'intention*, de *stratégie*, de *section*, *d'affinement*, de *classe* de *coloration* et de *relation d'alignement*. Ces éléments permettent de représenter explicitement et de raisonner sur la relation d'alignement à différents niveaux d'abstraction. En effet, pour chaque section du système nous précisons son degré d'alignement avec une section des processus et vice-versa. De plus, le mécanisme d'affinement permet de représenter une section par un graphe complexe d'intentions et de stratégies, c'est-à-dire une carte, permettant ainsi d'affiner la relation d'alignement elle-même.

La méthode ACEM contient une démarche formellement décrite qui permet de guider les ingénieurs d'alignement dans la correction et l'évolution de l'alignement tout en leur laissant une grande liberté de choix. Cette démarche est décrite sous forme de *directives* instanciant le méta-modèle de Carte. Certaines directives aident à naviguer dans les cartes ; d'autres indiquent comment réaliser les différentes intentions.

La méthode ACEM propose de spécifier explicitement les *exigences d'évolution*, traduisant des évolutions et des corrections de l'alignement, sous la forme d'opérateurs d'écart. Ces opérateurs permettent de construire les modèles To-Be (c'est-à-dire correspondant à la situation future) par comparaison avec les modèles As-Is (représentant la situation courante) sans spécifier à nouveau ce qui reste inchangé. La méthode ACEM guide les ingénieurs d'alignement pour (i) rétablir l'alignement entre système et processus et/ou (ii) faire évoluer conjointement ces deux entités, maintenant ainsi l'alignement face aux changements.

2. Perspectives

Le travail présenté dans cette thèse peut être poursuivi dans plusieurs directions :

- **Extension et affinement des métriques pour évaluer l'alignement** :

  L'évaluation de l'alignement ne tient pas compte de la gestion partielle par le système de concepts comcomplexes comme les buts, les processus ou les classes. En effet, dans le calcul de la mesure ces concepts sont comptabilisés de la même façon qu'ils soient partiellement ou pas du tout gérés par le système. Nous avons fait le choix délibéré de définir des métriques pour lesquelles une valeur égale à 1 correspond à un alignement parfait. De nouvelles métriques pourraient éventuellement être introduites afin de différencier la prise en compte partielle d'un concept composé d'une absence totale de gestion de ce concept par le système.

- **Automatisation des processus de génération des métriques et d'évaluation de l'alignement :**

  Notre définition des métriques à un niveau générique conduit à utiliser un processus pour générer des métriques spécifiques propres aux méta-modèles utilisés par les organisations. Un environnement informatisé générant des métriques spécifiques et calculant des mesures pour un projet donné faciliterait le travail de l'ingénieur d'alignement.

- **Enrichissement de la méthode ACEM pour prendre en compte les exigences d'évolution non fonctionnelles :**

  L'ingénierie des exigences distingue les exigences fonctionnelles qui définissent les services à fournir par le système d'information ou le processus futur des exigences non-fonctionnelles qui contraignent la manière dont le système ou les processus doivent satisfaire les exigences fonctionnelles ou la manière de le développer (par exemple ce qui touche à la sécurité ou la flexibilité du système). La méthode ACEM au travers des forces contextuelles ne permet de prendre en compte que les exigences fonctionnelles. La méthode ACEM pourrait être enrichie par la gestion des exigences d'évolution relatives à des exigences non-fonctionnelles.

- **Enrichissement des directives relatives à la propagation des écarts sur les modèles de l'organisation :**

L'exemple du chapitre 9 a montré que les répercussions sur les modèles de système et de processus des écarts spécifiés sur le modèle pivot se faisait de façons différentes en fonction du méta-modèle utilisé. En effet, ces répercussions utilisent les liens qui existent entre le méta-modèle pivot et le méta-modèle utilisé par l'organisation. On peut néanmoins se demander si des règles ou des directives génériques ne pourraient pas être identifiées permettant ainsi de faciliter le travail de l'ingénieur d'alignement dans cette étape.

- **Intégration de la méthode ACEM à d'autres approches d'ingénierie de l'alignement :**

Le méta-modèle pivot reposant sur les concepts du méta-modèle de Carte permet un couplage direct entre le système et le processus explicitant ainsi la relation d'alignement entre ces deux entités. La méthode ACEM pourrait être intégrée à d'autres approches d'ingénierie de l'alignement comme par exemple, celles qui s'intéressent aux objectifs stratégiques de l'organisation. Un projet de collaboration entre le Centre de Recherche en Informatique de l'Université Paris 1 et le laboratoire de recherche australien NICTA est en cours d'élaboration afin d'associer les travaux entrepris par Bleistein et Cox [Bleistein05a], [Bleistein05b] sur l'alignement entre la stratégie d'entreprise et les besoins et les travaux sur l'évaluation et l'évolution de l'alignement présentés dans cette thèse.

REFERENCES

[Aerts04] A.T.M. Aerts, J.B.M.Goossenaerts, D.K. Hammer, J.C. Wortmann. Architectures in context: On the evolution of business, application software, and ICT platform architectures Information and Management, Vol. 41, N°6, pp. 781-794, 2004.

[Al-Jadir03] Al-Jadir L. (2003) "Once Upon a Time a DTD Evolved into Another DTD" Object Oriented Information Systems, Lecture Notes in Computer Science, Vol.2817, pp.226-237, 2003.

[Al-Jadir95] L. Al-Jadir, T. Estier, G. Falquet, M. Léonard, "Evolution Features of the F2 OODBMS", *Proc. 4th Int. Conf. on Database Systems for Advanced Applications (DASFAA'95)*, Singapore, 1995. World Scientific Press, Advanced Database Research and Development, vol. 5.

[Andany97] J. Andany, M. Leonard, and C. Palisser, "Management of Schema Evolution in Databases", Proc. VLDB Conf., 1991, Morgan Kaufmann, pp. 161-170.

[Anderson02] Anderson S., Felici M., "Quantitative Aspects of Requirements Evolution". *COMPSAC 2002*, England, 2002, IEEE Computer Society, pp. 27-32

[ARC] http://www.ecommerce.uqi.uq.edu.au/arc/index.htm

[Arsanjani01] A. Arsanjani, J. Alpigini Using Grammar-oriented Object Design to Seamlessly Map Business Models to Component-based Software Architectures. In Procs of the International Symposium of Modelling and Simulation, Pittsburgh, PA, USA, pp 186-191. May 16-18, 2001

[Avison04] Avison, D., Jones, J., Powell, P., Wilson, D. (2004) Using and validating the strategic alignment model. Journal of Strategic Information Systems, 13, 223-246.

[Banerjee87] J. Banerjee, W. Kim, H.-J. Kim, H.F. Korth, Semantics and Implementation of Schema Evolution in Object Oriented Databases, Proceedings of the ACM-SIGMOD Annual Conference, pp 311-322, San Francisco, CA, May 1987.

[Barclay97] D.W. Barclay, Y.E. Chan, D.G.Copeland, S.L. Huff, Business Strategic Orientation, Information Systems Strategic Orientation and Strategic Alignment. Information Systems Research, Vol. 8, No2, 1997

[Beeson03] I. Beeson S. Al Mahamid, Survey of strategic alignment indicators in manufacturing companies in the South-West of England, Proceedings of the CEMS Research Student Conference, 2003.

[BenAchour99] C. Ben Achour Extraction des besoins par analyse de scénarios textuels, thèse de doctorat, Université Paris VI, janvier 1999.

[Bergeron99] F. Bergeron, L. Raymond, S. Rivard. Conceptualizing and Analyzing Fit in Information Systems Research : An Empirical Comparison of Perspectives. Cahier du GreSI No. 99-03, HEC Montréal, Montreal, 1999.

[Bezivin01] J. Bezivin From Object Composition to Model Transformation with the MDA, Proceedings of TOOLS, USA, Santa Barbara, August 2001

[Bleistein05a] S. J. Bleistein, K. Cox, J. Verner Strategic Alignment in Requirements Analysis for Organizational IT : an Integrated Approach 20th ACM Symposium on Applied Computing, track on Organisation Engineering, 13-17 March 2005, Santa Fe, USA.

[Bleistein05b] S. J. Bleistein, K. Cox, and J. Verner, "Validating Strategic Alignment of Organizational IT Requirements Using Goal Modeling and Problem Diagrams," Journal of Systems and Software, 2005

[Bodhuin04] T. Bodhuin, R. Esposito, C. Pacelli and M. Tortorella, Impact Analysis for Supporting the Co-Evolution of Business Processes and Supporting Software Systems, Proceedings of BPMDS'04, Workshop on Creating and Maintaining the Fit between Business Processes and Support Systems, Riga, Latvia, 2004.

[Boehm78] Boehm, B. W, Brown, J. R, Kaspar, H, Lipow, M, MacLeod, G and Merritt, M. J.(1978) Characteristics of Software Quality, North Holland.

[Bonne04] J.C. Bonne, A. Maddaloni. *Convaincre pour urbaniser le SI*. Hermes, Lavoisier 2004

[BPMDS04] Workshop on Creating and Maintaining the Fit between Business Processes and Support Systems, Riga, Latvia, 2004

[Brancheau96] J. C Brancheau,.B. Janz and J. C. Wetherbe (1996). Key issues in information systems management: 1994-95 SIM Delphi results. MIS Quarterly, Vol. 20, n°2, pp. 225-242.

[Breton02] E. Breton,.Contribution à la représentation de processus par des techniques de métamodélisation. PhD thesis, Thèse de Doctorat de l'Université de Nantes. N°0366-066, 24 juin 2002.

[Broadbent93] M. Broadbent and P. Weill, "Improving Business and Information Strategy Alignment: Learning from the Banking Industry," IBM Systems Journal vol.32, n°.1, 1993

[Bubenko94] J. Bubenko. *Enterprise Modelling*. In Ingenierie de Systèmes d'Information, Vol 2, n° 6, pp. 657-678.1994.

[Bunge77] Bunge, M (1977) Treatise on Basic Philosophy: Ontology I. The Furniture of the World, Reidel.

[Bunge79] Bunge, M (1979) Treatise on Basic Philosophy: Ontology II. A World of Systems, Reidel.

[Camponovo04] G. Camponovo, Y. Pigneur Information Systems Alignment in Uncertain Environments, In Proceedings of the IFIP International Conference on Decision Support Systems (DSS2004), Prato, 2004

[Casati96] F. Casati, S. Ceri, B. Pernici, G. Pozzi Workflow Evolution. In Proc. of 15th Int. Conf. On Conceptual Modeling (ER'96), Cottbus, Germany, pp. 438-455, 1996

[Cavano88] J.P. Cavano, J.A. McCall, (1988) A framework for the management of quality. In: Proc of the ACM Software Assurance Quality Assurance Workshop, (ACM, New-York) pp. 133-139

[Chan97] Y. Chan, S. Huff, D. Barclay, D. Copeland, (1997) Business Strategic Orientation: Information Systems Strategic Orientation and Strategic Alignment. Information Systems Research, 8, 125-150.

[Chen76] P. Chen. The Entity-Relation Model - Towarda Unified View of Data. ACM Transactions on Database System, Vol. 1, N°1, pp. 9-36, March 1976.

[Chidamber94] S.R.Chidamber, C.F. Kemerer, A Metric Suite for Object Oriented Design. IEEE Transactions on Software Engineering, Vol. 20, N°6, pp476-493, June 1994.

[Ciborra97] C. Ciborra,. "De profundis? Deconstructing the concept of strategic alignment." Scandinavian Journal of Information Systems Vol. 9 n°1, pp. 67–82. 1997

[Clarke99] S. Clarke, W. Harrison, H. Ossher and P. Tarr. "Subject-Oriented Design: Towards Improved Alignment of Requirements, Design and Code", *Proceedings of Object-Oriented Programming, Systems, Languages and Applications* (OOPSLA), 1999

[Coakley96] J.R. Coakley, M.K. Fiegener, and D.M. White, "Assessing Strategic IT Alignment in A Transforming Organisation," Proceedings of the Association for Information Systems, Phoenix Arizona, 1996

[Croteau01] Croteau, A.-M. & Bergeron, F. (2001) An Information Technology Trilogy: Business Strategy, Technological Deployment and Organizational Performance. Journal of Strategic Information Systems, 10, 77-99.

[D'Souza99] D. F. D'Souza, A. C. Wills. Objects, Components, and Frameworks with UML The Catalysis Approach. Addison-Wesley, Object Technology Series. 2001.

[Dardenne93] A. Dardenne, A. Lamsweerde, S. Fickas *Goal-directed Requirements Acquisition*, Science of Computer Programming, 20, Elsevier, pp.3-50, 1993.

[Davenport00] T.H. Davenport Mission Critical: Realizing the Promise of Enterprise Systems. Harvard Business. School Press, MA, 2000.

[Davenport98] T.H. Davenport Putting the Enterprise into the Enterprise System. Harvard Business Review. Vol. 76, N°4, pp. 121-131, 1998.

[Delgado03] Delgado C., Samos J. and Torres M., "Primitive operations for schema evolution in ODMG databases," Object Oriented Information Systems, Lecture Notes in Computer Science, Vol.2817, pp.226-237, 2003.

[Denéckère01] R. Denéckère, *Approche d'extension de méthodes fondée sur l'utilisation de composants* génériques. Thèse de doctorat en informatique, Université Paris 1 – Sorbonne, janvier 2001.

[Dittrich94] K R. Dittrich, "Object-Oriented Data Model Concepts", Advances in Object-Oriented Database Systems, NATO ASI Series, Series F: Computer and System Science, Vol. 130, 29-45, Springer Verlag, New York, 1994.

[Dromey95] R. G. Dromey, (1995) 'A Model for Software Product Quality', IEEE Transactions on Software Engineering, pp. 146-162.

[Estublier00] J. Estublier and M. Nacer Schema Evolution in Software Engineering Databases -- A new Approach in Adele environment CAI Computer and Artificial Intelligence Journal. June 2000. Vol 19. pages 183-203.

[Etien05a] A. Etien, C. Salinesi, "Managing Requirements in a Co-evolution Context". *Proceedings of the IEEE International Conference on Requirements Engineering*, Paris, France, Sept. 2005

[Etien05b] A. Etien and C. Rolland "A Process for Generating Fitness Measures", *Proceedings of CAISE'05*, Porto, Portugal, June, 2005

[Etien05c] A. Etien, C. Rolland and C. Salinesi " Measuring the Business / System Alignment", *Proceeding of REBNITA*, Paris, France, August 2005

[Etien05d] A. Etien and C. Rolland "Measuring the fitness relationship", *Requirement Enginnering Journal,* Vol. 10, N°3, pp.184-197, 2005

[Etien04] A. Etien, C. Rolland, and C. Salinesi "Overview of a Gap-driven Evolution Process", *Proceedings of Australian Workshop on Requirements Engineering, AWRE*, Adelaid, Australia, December, 2004

[Etien03a] A. Etien, and C. Salinesi, "Towards a Systematic Definition of Requirements for Software Evolution: A Case-study Driven Investigation". *Proceedings of EMMSAD'03*, Velden, Austria, 2003.

[Etien03b] A. Etien, R. Deneckère and C. Salinesi "Extending Methods to Express Change Requirements", *Proceeding of EMSISE*, Geneva, Switzerland, Sept. 2003

[Finkelstein02] D. Bush, A. Finkelstein. Requirements Elicitation for Complex Safety Related Systems. In Procs London Communication Symposium. London, UK, Sept. 2002

[Fitzpatrick96] Ronan Fitzpatrick. Software quality definitions and strategic issues. Staffordshire University, 1996. http://www.comp.dit.ie/rfitzpatrick

[Giaglis99] Giaglis, G.M. On the Integrated Design and Evaluation of Business Processes and Information Systems. Communications of the AIS, vol 2, N°5, July 1999.

[Glass81] R.L. Glass, R.A. Noiseux; "Software Maintenance Guidebook", Prentice-Hall, ISBN 0-13-821728-9, 1981, 193 pages.

[Godet00] M. Godet, The Art of Scenarios and Strategic Planning: Tools and Pitfalls. Technological Forecasting and Social Change, vol. 65, pp. 3–22, 2000.

[Goedvolk00] H. Goedvolk, A. van Schijndel, V. van Swede, R. Tolido, 2000. The Design, Development and Deployment of ICT Systems in the 21st Century: Integrated Architecture Framework (IAF). Cap Gemini Ernst and Young.

[Gotel94] O. Gotel, A. Finkelstein, An Analysis of the Requirements Traceability Problem Proc. of First International Conference on Requirements Engineering, 1994, pages 94-101

[Green04] Green P.F., Rosemann M. (2004) Applying Ontologies to Business and Systems Modelling Techniques and Perspectives: Lessons Learned. Journal of Database Management, Vol 15(2), pp. 105-117

[Grundy96] Grundy, J.C., Venacle, J.R.: Towards an integrated environment for method engineering, In W. W. Cotterman and J. A. Senn (Eds.) Challenges and Strategies for Research in Systems Development, John Wiley&Sons, Chistester, pp. 45-62, 1996.

[Han97] J. Han, Supporting Impact Analysis and Change Propagation in Software Engineering Environments, Proceedings of Workshop on Software Technology and Engineering Practice (STEP'97 / CASE'97), IEEE Computer Society Press, London, UK, July 1997, pages 172-182.

[Harker93] : Harker S.D.P., Eason K.D. & Dobson J.E., "The Change and Evolution of Requirements as a Challenge to the Practice of Software Engineering", IEEE Symposium on Requirements Engineering, RE'93, San Diego, CA, Jan. 4-6, 1993, 266-272.

[Hearnden03] D. Hearnden Software Evolution with the Model-Driven Architecture, Internal report, University of Queensland, Australia.

[Henderson89] Henderson, J., Venkatramen, N., 1989. Strategic Alignment: A Model for Organisational Transformation, in: Kochan, T., Unseem, M. (Eds.), 1992. Transforming Organisations. OUP, New York.

[Henderson93] J. C. Henderson, N. Venkatraman, Strategic alignment: Leveraging information technology for transforming organizations, IBM Systems Journal, Vol. 32, No 1, 1993, pp4-16, reprint in IBM Systems Journal, Vol. 38, No2, 1999, pp 472-484.

[Hu05] Q. Hu, C. D. Huang, Aligning IT with Firm Business Strategies Using the Balance Scorecard System, Proceedings of the 38th Hawaii International Conference on System Sciences, Track 8 - Volume 08, Hawaii, 2005.

[IBM03] R. Adams Renner, D. Latimore and D. Wong, Business and IT operational models in financial services: Beyond strategic alignment, *IBM Institute for Business Value study*, 2003 http://www-1.ibm.com/services/us/imc/pdf/g510-3267-business-and-it-operational-models-financial-services.pdf

[IEEE SGSET91] IEEE Standard Glossary of Software Engineering Terminology, IEEE Software Engineer Standard Collection, Spring 1991 Edition, Institute for Electrical and Electronic Engineers, New York, New York (1991) p.61.

[IRDS90] Information Technology-Information Resource Dictionary System (IRDS) – Framework, ISO/IEC International Standard, 1990.

[ISO/IEC 9126] Norme ISO : Software Engineering - Product quality - Part 1: Quality model 2001

[Jackson01] M. Jackson, *Problem Frames: Analysing and structuring Software Development Problem,* Addison-Wesley Publishing Company.

[Jackson95] M. Jackson. *Software Requirements and Specifications*. Addison-Wesley. 1995

[Jacobson99] I. Jacobson, G. Booch, J. Rumbaugh. Unified Software Development Process. Addison-Wesley, Object Technology Series. 1999.

[Jarke93] Jarke, M., Pohl, K.: Establishing visions in context: toward a model of requirements processes. In Proceeding 12th International Conference Information Systems, Orlando, Fl, 1993.

[Jarke99] Jarke M., Rolland C., Sutcliffe A., Domges R. The NATURE requirements Engineering. Shaker Verlag, Aachen 1999.

[Kaplan92] R.S. Kaplan, and D.P. Norton (1992) "The Balanced Scorecard: Measures That Drive Performance," Harvard Business Review, Vol. 70, N° 1, pp. 71-79.

[Kardasis98] Kardasis, P. and Loucopoulos, P. Aligning Legacy Information Systems to Business Processes, Proceedings of CAiSE*98, Pisa, Italy, 1998, pp. 25 – 40

[Knoll94] K. Knoll, S. L. Jarvenpaa, Information technology alignment or "fit" in highly turbulent environments: the concept of flexibility, Proceedings of the computer personnel research conference on Reinventing IS, Alexandria, Virginia, United States, 1994.

[Kolber00] A.B. Kolber, C.K. Estep, D.C. Hay, D. Struck, G.S.W. Lam, J.D. Funk, J.D. Healy, J. Hall, J.A. Zachman, K. Anderson Healy, M. Eulenberg, N.A. Fishman, R.G. Ross, T. Moriarty, W.L. Selkow Organizing Business Plans The Standard Model for Business Rule Motivation, The Business Rules Group., November 2000, http://neptune.irit.fr/Biblio/02-10-04.pdf

[Kradolfer00] M. Kradolfer. A Workflow Metamodel Supporting Dynamic, Reuse-based Model Evolution. PhD thesis, Department of Information Technology, University of Zurich, Switzerland, mai 2000, chap. 4, pp. 59-73

[Krishna04a] A. Krishna, A.K. Ghose, S. Vilkomir, "Co-Evolution of Complementary Formal and Informal Requirements", *Proceedings of International Workshop on Principles of Software Evolution* (IWPSE'04), September, 2004, Kyoto, Japan, pp. 159-164

[Krishna04b] A. Krishna, S.A. Vilkomir, A.K. Ghose: A Case Study of Combining I* Framework and the Z Notation Proceedings of the International Conference on Enterprise Information Systems, Porto, Portugal, April, 2004, pp. 192-200

[Kumar92] K. Kumar et R.J. Welke "Methodology Engineering: A Proposal for Situation-Specific Methodology Construction" in Chalenges and Strategies for Research in Systems Development, W.W. Cotterman et J.A. Senn (Eds), Wiley, 1992.

[Lämmel04] R. Lämmel "Coupled Software Transformations", Proceedings of International Workshop on Software Evolution Transformation SET2004, 2004, pp31-35

[Lamsweerde00] A. van Lamsweerde, Requirements Engineering in the Year 2000: A research perspective. International Conference on Software Engineering, Limerick, Ireland, 2000, pp 5-19.

[Lamsweerde01] A. van Lamsweerde. *Goal-Oriented Requirements Engineering: A Guided Tour.* Invited Paper for RE'01 - 5th IEEE International Symposium on Requirements Engineering, Toronto, August, 2001, pp. 249-263

[Lamsweerde03] A. van Lamsweerde, From System Goals to Software Architecture In Formal Methods for Software Architectures, M. Bernardo & P. Inverardi (eds), LNCS 2804, Springer-Verlag, 2003, 25-43

[Landtsheer03] R. de Landtsheer, E. Letier, A. van Lamsweerde, "Deriving Tabular Event-Based Specifications from Goal-Oriented Requirements Models", *Proceedings of RE'03, IEEE International Conference on Requirements Engineering*, Montgomery Bay, USA, September 2003.

[Landtsheer04] R. de Landtsheer, E. Letier, A. van Lamsweerde, "Deriving Tabular Event-Based Specifications from Goal-Oriented Requirements Models", *Requirements Engineering Journal*, Vol. 9, 2004, pp. 104-120.

[Lausen97] S Lausen, G Vossen. 1997, Models and languages of Object/Oriented Databases, Addison-Wesley, Harlow, England, 1997

[Lee97] Lee S.P., Rolland C., Brunet J.. Abstraction in an Object-Oriented Analysis Method. in Malaysian Journal of Computer Science, Vol. 10, No 1, June 1997, p. 53-63.

[Lehman80] M.M. Lehman; "Programs, life cycles, and laws of software evolution" in Proc. IEEE, Vol. 68, No. 9, September 1980, pp. 35-41.

[Lewis98] P. Lewis, S. Goodman, P. Fandt. *Challenges in the 21st Century Management*. South-Western, 1998.

[Longépé01] C. Longépé. *Le projet d'urbanisation du SI.* Collection Informatique et Entreprise, Dunod 2001

[Lubars93] Lubars, M., Potts, C., Richer, C.: A review of the state of the practice in requirements modeling.Proc. IEEE Symp. Requirements Engineering, San Diego 1993.

[Luftman00] J Luftman, Assessing business-IT alignment maturity. Communications of the Association for Information Systems, Vol. 4, N°14, pp. 1-50, 2000.

[Luftman04] J. Luftman, E. R. Maclean, "Key issues for IT executives". *MIS Quarterly Executive*, 3, 2004, pp. 89-104.

[Luftman96] J. N. Luftman, "Competing in the Information Age", Oxford University Press, 1996

[Lyytinen87] Lyytinen K., Different perspectives on information systems: problems and solutions, ACM Computing Surveys, Vol 19, N°1, 1987.

[Maes99] Maes, R., 1999. A Generic Framework for Information Management. Prime Vera Working Paper, Universiteit Van Amsterdam http://primavera.fee.uva.nl/PDFdocs/99-03.pdf (page consultée le 24 novembre 2005)

[Maes00] Maes, R., Rijsenbrij, D., Tuuijens, O., & Goedvolk, H. (2000, June). Redefining business-IT alignment through a unified framework. Primavera Working Paper Series 2000-19, Universiteit van Amsterdam, Cap Gemini.

[Martiin94] Marttiin, P., Methodology Engineering in CASE shells: Design Issue and Current Practice, PhD thesis, Computer science and information systems reports, Technical report TR-4, 1994

[Marvie04] R. Marvie, Vers des patrons de méta-modélisation, "Interopérabilité des systèmes distribués: des modèles aux intergiciels", Special issue of Technique et Science Informatique (TSI), Hermès Sciences, Vol. 23, N°10, pp. 1355–1382, 2004.

[Matheron94] J.P. Matheron, Comprendre Merise, Eyrolles, ISBN : 2-212-07502-2

[McGraw97] Karen Mc Graw, Karan Harbison, User Centered Requirements, The Scenario-Based Engineering Process. Lawrence Erlbaum Associates Publishers, 1997.

[McKeen03] Mckeen, J. D. & Smith, H. (2003) Making IT happen: critical issues in IT management, Chichester; Hoboken, NJ, Wiley.

[Mens00] T. Mens. *Conditional graph rewriting as a domain-independent formalism for software evolution*. Proc. Int. Agtive '99 Conference, LNCS 1779: 127-143, Springer-Verlag, 2000.

[Mens01] T. Mens. Transformational Software Evolution by Assertions. In T. Mens and M. Wermelinger, editors, International Special Session on Formal Foundations of Software Evolution, Lisboa, Portugal, March 2001. Co-located with the European Conference on Software Maintainance and Reengineering (CSMR 2001)

[Meta03] META Group Research on Requirements Realization and Relevance, report, 2003

[Mitleton00] E. Mitleton-Kelly and M-C Papaefthimiou "Co-evolution & an enabling infrastructure: a solution to legacy?", Systems engineering for business process change', Peter Henderson, Springer-Verlag, 2000, ch. 14

[MOF] http://www.omg.org/cgi-bin/apps/doc?formal/02-04-03.pdf

[Munford81] E. Munford Participative Systems Design: Structure and Method systems, Objectives, Solution, Vo. 1, North-Holland, pp. 5-19, 1981.

[Nurcan99] S. Nurcan, J. Barrios, G. Grosz, C. Rolland. "Change process modelling using the EKD-Change Management Method" Proceedings of ECIS 99, Denmark, 1999.

[OMG03] OMG. Unified Modeling Language v1.5 Specification. Report formal/03-03-01, March 2003.

[Opdhal98] A. L. Opdhal, K. Pohl, Workshop Summary REFSQ'98. Proceedeings of the Fourth International Workshop on Requirements Engineering : Foundations of Software Quality, REFSQ'98, Pisa, Italy Presses Universitaires de Namur, (eds, E. Dubois, A.L. Opdhal, K. Pohl), pp.1-11, 1998.

[Parikh86b] G. Parikh; "Software Maintenance Notes (1)",in ACM SIGSOFT, Software Engineering Notes, Vol. 11, N. 2, April 1986, pp. 49-57.

[Plihon94] Plihon V., The OMT Methodology, The OOA Methodology (Coad/yourdon), The SA/SD Methodology, The Entity Relationship Methodology, The O* Methodology, The OOD Methodology, NATURE Deliverable DP2, 1994.

[Plihon95] Plihon V., Rolland C., Modelling Ways-of-Working, Proc 7th Int. Conf. on Advanced Information Systems Engineering (CAISE), Springer Verlag (Pub.), 1995.

[Plihon96] Plihon V., Un environnement pour l'ingénierie des méthodes, Thèse de doctorat de l'Université Paris 1, janvier 1996.

[Plihon97] Plihon, V., Rolland, C.: Using a generic approach to support the construction of methods. In the Proceedings of the 8th International Conference on Database and Expert Systemes Applications (DEXA'97), Toulouse, France, September 1-7, 1997.

[Pohl96] Pohl K (1996) Process-Centered Requirements Engineering, John Wiley & Sons Inc

[Potts94] C. Potts, K. Takahashi, A.I. Anton Inquiry-based requirements analysis. In IEEE Software Vol. 11, N°2, pp. 21-32, 1994.

[Potts97] C. Potts. *Fitness for Use: The System Quality that Matters Most.* In Procs REFSQ'97: Third Int.Workshop on Requirements Engineering: Foundation for Software Quality. Barcelona, Spain: June 16-17, 1997

[Prakash99] Prakash, N., On Method Statics and Dynamics. Information Systems. Vol. 24, No.8, pp 613-637. 1999.

[Prat97] N. Prat,: Goal formalisation and classification for requirements engineering. In: Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ'97, Barcelona (1997), 145-156.

[Ralyté01] Ralyté J., "Ingénierie des méthodes à base de composants", Thèse de Doctorat, Université de Paris I, Janvier 2001.

[Ralyté04] Ralyté J, Rolland C, Deneckère R (2004) Towards a Meta-Tool for Change-Centric Method Engineering: a Typology of Generic Operators. Proceedings of CAISE'04, Riga, Latvia

[Ramesh01] B. Ramesh, M. Jarke, Toward Reference Models for Requirements Traceability IEEE Transactions on Software Engineering, Vol. 27, No. 1, January 2001, pages 58-93

[REBNITA05] 1st International Workshop on Requirements Engineering for Business Need and IT Alignment, Paris, Aout, 2005.

[Regev04] G. Regev, A. Wegmann, Remaining Fit: On the Creation and Maintenance of Fit, Proceedings of BPMDS Workshop on Creating and Maintaining the Fit between Business Processes and Support Systems, Riga, Latvia, 2004, pp131-137

[Reich03] B. H. Reich, K. M. Nelson, "In Their Own Words: CIO Visions About the Future of In-House IT Organizations". *The DATA BASE for Advances in Information Systems*, 34, 2003, pp. 28-44.

[Reichert98] M. Reichert and P. Dadam. ADEPTflex-Supporting Dynamic Changes of Workflows Without Loosing Control. Journal of Intelligent Information Systems 10(2), 1998.

[Renner03] R. A. Renner, D. Latimore and D. Wong, Business and IT operational models in financial services: Beyond strategic alignment, *IBM Institute for Business Value study*, 2003 http://www-1.ibm.com/services/us/imc/pdf/g510-3267-business-and-it-operational-models-financial-services.pdf

[Robert00] Le Petit Robert, Dictionnaire le Robert, France, 2000.

[Robson97] Robson, W.: *Strategic Management and Information Systems*, Second Edition, Financial Times, Prentice Hall, 1997

[Rolland01] C. Rolland, N. Prakash. *Matching ERP System Functionality to Customer Requirements. In Procs* of the 5th IEEE International Symposium on Requirements Engineering, Toronto, Canada. August 27-31, 2001.

[Rolland03] C. Rolland. "Ingénierie des Besoins : L'Approche L'Ecritoire", Journal Techniques de l'Ingénieur (TI), 2003

[Rolland04] C. Rolland, C. Salinesi, A. Etien, "Eliciting Gaps in Requirements Change". *Requirement Engineering Journal* Vol. 9, N°1, pp1-15, 2004

[Rolland88] C. Rolland, O. Foucault, G. Benci *Conception des systèmes d'information: La méthode REMORA*. Eyrolles, 1988.

[Rolland94] Rolland C., Prakash N., A Contextual Approach to modeling the Requirements Engineering Process, SEKE'94, 6th International Conference on Software Engineering and Knowledge Engineering, Vilnius, Lithuania, 1994.

[Rolland95] Rolland C., Souveyet C., Moreno M.. An Approach for Defining Ways-Of-Working, in the Information Systems Journal, 1995.

[Rolland96] C. Rolland "Conception de Bases de Données: Une Méthode Orientée Objet et Evénement", Techniques de l'Ingénieur, Ref H3248, Vol 6, 1996.

[Rolland98] C. Rolland, C. Ben Achour, C. Cauvet, J. Ralyte, A. Sutcliffe, N. A. M. Maiden, M. Jarke, P. Haumer, K. Pohl, E. Dubois, and P. Heymans "A Proposal for a Scenario Classification Framework", Requirements Engineering Journal (REJ), Vol. 3, N°1, pp. 23-47, 1998

[Rolland98] Rolland, C., Souveyet, C., Salinesi, C.: Guiding Goal Modelling using Scenarios, IEEE Trans. On Software Engineering (Special Issue on Scenario Management), 1998, 24(12): 1055-1071.

[Rosemann02] Rosemann, M., Green, P. Developing a meta model for the Bunge-Wand-Weber ontological constructs. Information Systems, Vol. 27, N°2, 75-91, 2002

[Rosemann04] M. Rosemann, I. Vessey, R. Weber, Alignment in Enterprise Systems Implementations: The Role of Ontological Distance, In Proceedings of the International Conference of Information Systems, Washington, D.C., USA, 12-15 December, pp. 11-19, 2004.

[Royce70] Royce W. W. (1970), Managing the Development of Large Software Systems; Proc. IEEE WESCON 08.

[Rubin96] KS Rubin, A Goldberg Object Behavior Analysis, Communications of the ACM, Vol. 35, n°9, 1992.

[Sabherwal01] R. Sabherwal, and Y. E. Chan. "Alignment Between Business and IS Strategies: A Study of Prospectors, Analyzers, and Defenders". *Information Systems Research*, Vol. 12, N°1, March 2001, pp. 11-33.

[Salinesi03] Salinesi C., Rolland C., « Fitting Business Models to Systems Functionality Exploring the Fitness Relationship ». *Proceedings of CAiSE'03*, Velden, Austria, 2003.

[Salinesi04a] C. Salinesi, A. Etien, I. Zoukar, A Systematic Approach to Express IS Evolution Requirements Using Gap Modelling and Similarity Modelling Techniques, *Proceedings of CAiSE,* Riga, Lettonie, June, 2004

[Salinesi04b] C. Salinesi, A. Etien and J. Wäyrynen , "Towards a Systematic Propagation of Evolution Requirements in IS Adaptation Projects" *Proceeding of Australian Conference on Information System ACIS*, Hobart, Australia, 2004

[Scheer98] A.W. Scheer, Business Process Engineering. Reference Models for Industrial Enterprises, Springer, Berlin, 1998.

[SchulzeKremer97] Schulze-Kremer, Steffen 1997. Adding Semantics to Genome Databases: Towards an Ontology for Molecular Biology. in Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology, T. Gaasterland, et al. (eds.), Halkidiki, Greece, June 1997. Palo Alto: AAAI Press, 272-275.

[Schwer99] S. Schwer, and C. Rolland "Theoritical Formalisation of the Meta-Modelling Approach", The NATURE of Requirements Engineering, Shaker Verlag, Aachen, pp. 361 - 376, 1999.

[Seligmann89] Seligmann P. S., Wijers G. M., Sol H. G., "Analyzing the structure of I. S. methodologies, an alternative approach", in Proc. of the 1st Dutch Conference on Information Systems, Amersfoort, The Netherlands, 1989.

[Soffer04a] P. Soffer Analyzing the Scope of a Change in a Business Process Model, Proceedings of BPMDS'04, Workshop on Creating and Maintaining the Fit between Business Processes and Support Systems Riga, Latvia, 2004.

[Soffer04b] P. Soffer, "Fit Measurement: How to Distinguish Between Fit and Misfit", note for BPMDS'04, Workshop on Creating and Maintaining the Fit between Business Processes and Support Systems Riga, Latvia, 2004

[Soffer04c] Soffer P, Wand Y (2004) Goal-Driven Analysis of Process Model Validity. Proceedings of CAiSE'04, Riga, Latvia.

[Soffer05] P. Soffer, C. Rolland, "Combining Intention-Oriented and State-Based Process Modeling" Proceedings of ER2005, Klagenfurt, Austria, 2005

[Standish 95] The Standish Group, Chaos. Standish Group Internal Report, http://www.standishgroup.com/chaos.html, 1995.

[Swanson76] E.B. Swanson; "The Dimensions of Maintenance", in Proc. 2nd International Conference on Software Engineering, 1976, pp. 492-497.

[SysML] http://www.sysml.org

[Tallon02] Tallon, P. P. & Kraemer, K. L. (2002) Executives' Perspectives on IT: Unraveling the Link between Business Strategy, Management Practices and IT Business Value. Americas Conference on Information Systems (ACIS2002). Dallas, TX, USA

[Tamzalit03] D. Tamzalit, M. Oussalah "A Conceptualization of OO Evolution", Object Oriented Information Systems, Lecture Notes in Computer Science, Vol.2817, pp.274-278, 2003.

[Teeuw97] Teeuw W. B., van den Berg H. (1997), On the Quality of Conceptual Models, Proceedings of the 16th International Conference on Conceptual Modeling (ER'97), Los Angeles, CA, November 1997.

[Terrasse03] M-N Terrasse, M. Savonnet, G. Becker, E. Leclercq, "UML-based Metamodeling for Information System Engineering and Evolution", Proceedings of the 9th International Conference on Object-Oriented Information Systems (OOIS'03), LNCS 2817, Springer Verlag, pp. 83-94, 2003

[UML] http://www.uml.org

[VanderAalst04] van der Aalst W.M.P. (2004) Business Alignment: Using Process Mining as a Tool for Delta Analysis. Proceedings of BPMDS'04, Riga, Latvia.

[Vilkomir04] S. Vilkomir, A. Ghose and A. Krishna. Combining agent-oriented conceptual modeling with formal methods, Proceedings of Australian Software Engineering Conference (ASWEC 2004), April 2004, Melbourne, Australia, IEEE Computer Society, pp. 147-157

[Wand04] Wand Y., Weber R. (2004) Reflection: Ontology in Information Systems, Journal of Database Management, Vol 15(2).

[Wand92] Wand Y, Weber R (1992) An Ontological Model of an Information System, IEEE Transactions on Software Engineering, November, pp. 1282-92

[Wand93] Wand Y, Weber R (1993) On the Ontological Expressiveness of Information Systems Analysisand Design Grammars. Journal of Information Systems, 3(4),217–237.

[Wan-Kadir04] W. M. N. Wan-Kadir , P. Loucopoulos, Relating evolving business rules to software design, Journal of Systems Architecture: the EUROMICRO Journal, v.50 n.7, p.367-382, July 2004

[Weber96] R. Weber and Y. Zhang (1996) An analytical evaluation of NIAM's grammar for conceptual schema diagrams. Information Systems Journal, 6, pp. 147–170.

[Weber97] Weber, R. (1997) Ontological Foundations of Information Systems, Buscombe Vicprint, Blackburn, Victoria.

[Wegmann05a] A. Wegmann, P. Balabko, L. S. Le, G. Regev, I. Rychkova "A Method and Tool for Business-IT Alignment in Enterprise Architecture" Proceedings of CAiSE`05 Forum, Porto, Portugal, June 2005, pp. 113-118.

[Wegmann05b] A. Wegmann, R.Regev, B. Loison, Business and IT Alignment with SEAM, Proceedings of REBNITA Requirements Engineering for Business Need and IT Alignment, Paris, France, August 2005.

[Wieringa03] R.J. Wieringa, H.M. Blanken, M.M. Fokkinga, and P.W.P.J. Grefen. ``Aligning application architecture to the business context." In Conference on Advanced Information System Engineering (CAiSE 03), pages 209–225. Springer, 2003. LNCS 2681.

[Woodfield97] S.N. Woodfield “The Impedance Mismatch Between Conceptual Models and Implementation Environments”, ER’97 Workshop on Behavioral Models and Design Transformations: Issues and Opportunities in Conceptual Modeling, 6 - 7 November 1997, UCLA, Los Angeles, California.

[Yu01] E. Yu. “Agent Orientation as a Modelling Paradigm”. *Wirtschaftsinformatik*. Vol. 43 n°2 April 2001. pp. 123-132.

[Yu97] E. Yu Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering, Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering (RE'97) Jan. 6-8, 1997, Washington D.C., USA. pp. 226-235.

[Zachman99] J. A. Zachman A framework for information systems architecture, IBM system Journal Volume 26, Number 3, Page 276 (1987)

[Zicari92] Roberto Zicari. A Framework for Schema Updates in an Object-Oriented Database System. Chapter 7 in F. Bancilhon, C. Delobel, and P. Kanellakis, editors, Building an Object-Oriented Database System—the Story of O2 Morgan Kaufmann Publishers, 1992.

[Zoukar05] I. Zoukar, "MIBE : Méthode d'Ingénierie des Besoins pour l'implantation d'ERP", Thèse, Université Paris 1, Avril 2005.

[Zowghi96] D. Zowghi, A.K. Ghose, P. Peppas A Framework for Reasoning about Requirements Evolution Proceedings of the 4th Pacific Rim International Conference on Artificial Intelligence (PRICAI96), Cairns, Australia, August 1996

ANNEXE 1

1. Description du processus

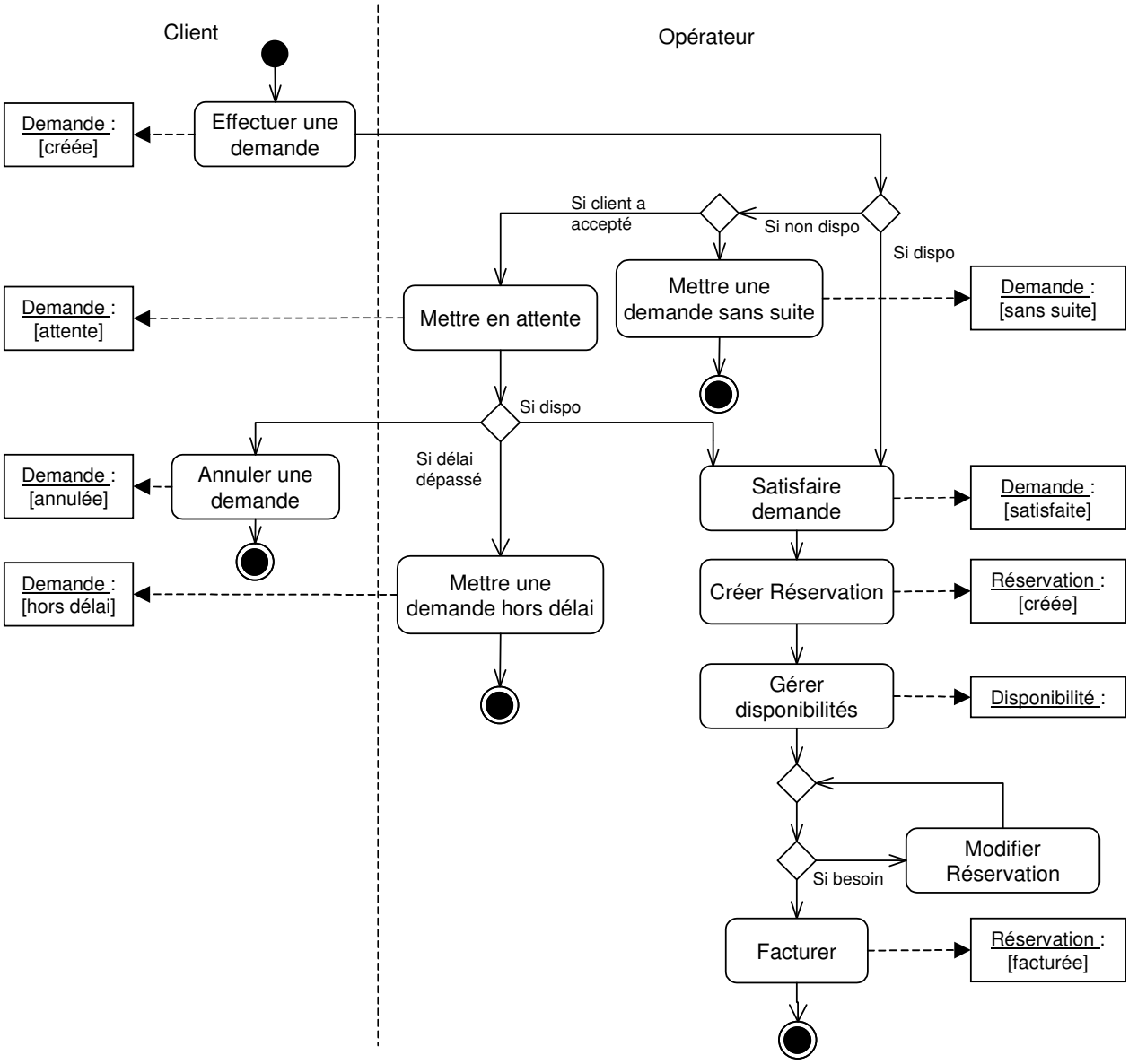


Figure 148 : Processus de création des réservations

Ce processus a déjà été présenté dans le corps de la thèse au chapitre 4, Figure 35.

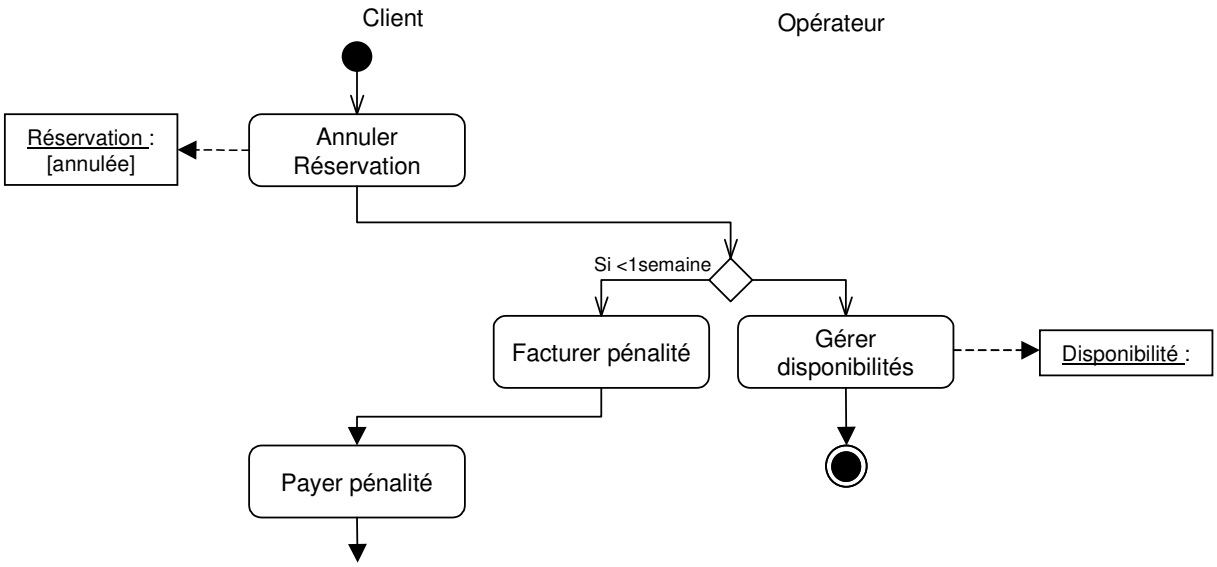


Figure 149 : Processus d'annulation de réservation par le client

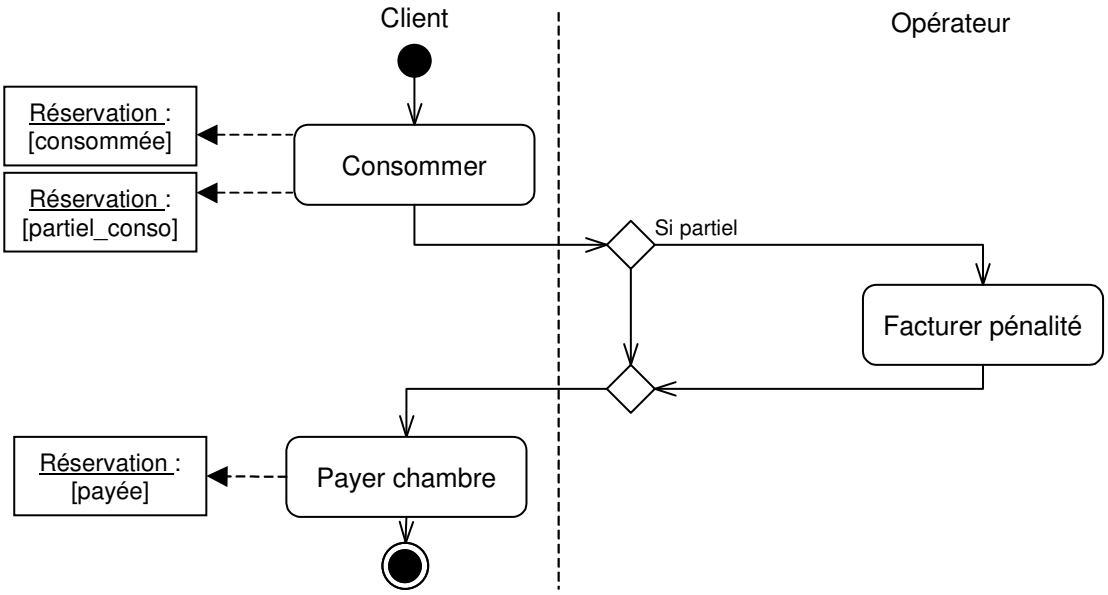


Figure 150 : Processus de consommation et de paiement

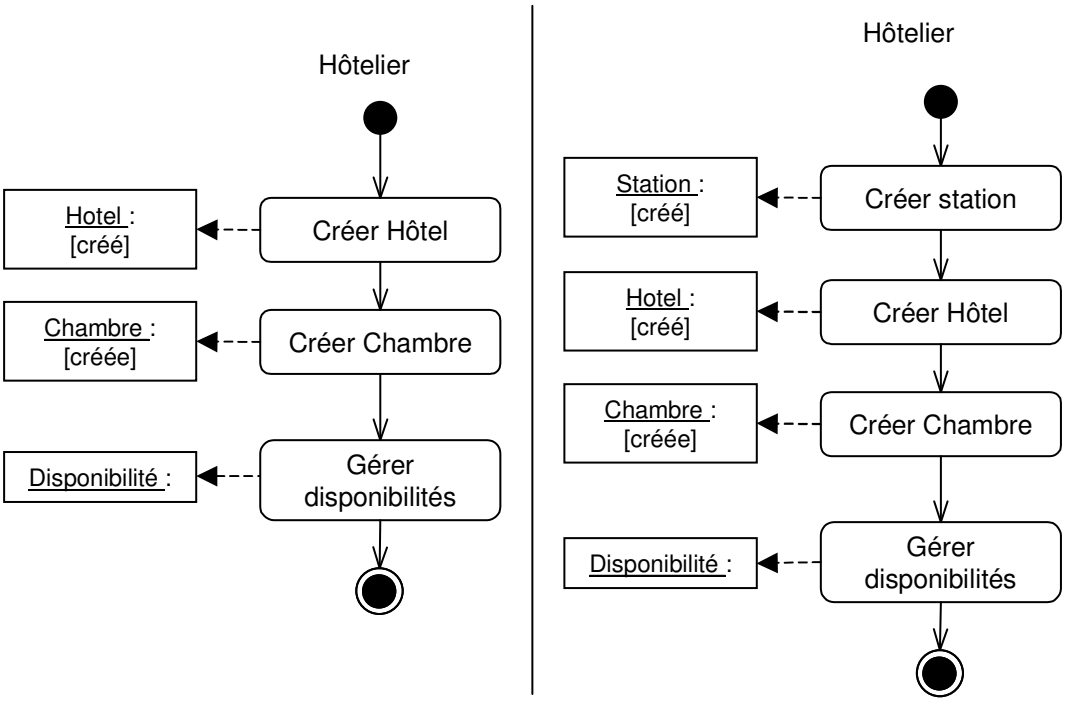


Figure 151 : Processus de création d'hôtel à gauche et de création de station à droite

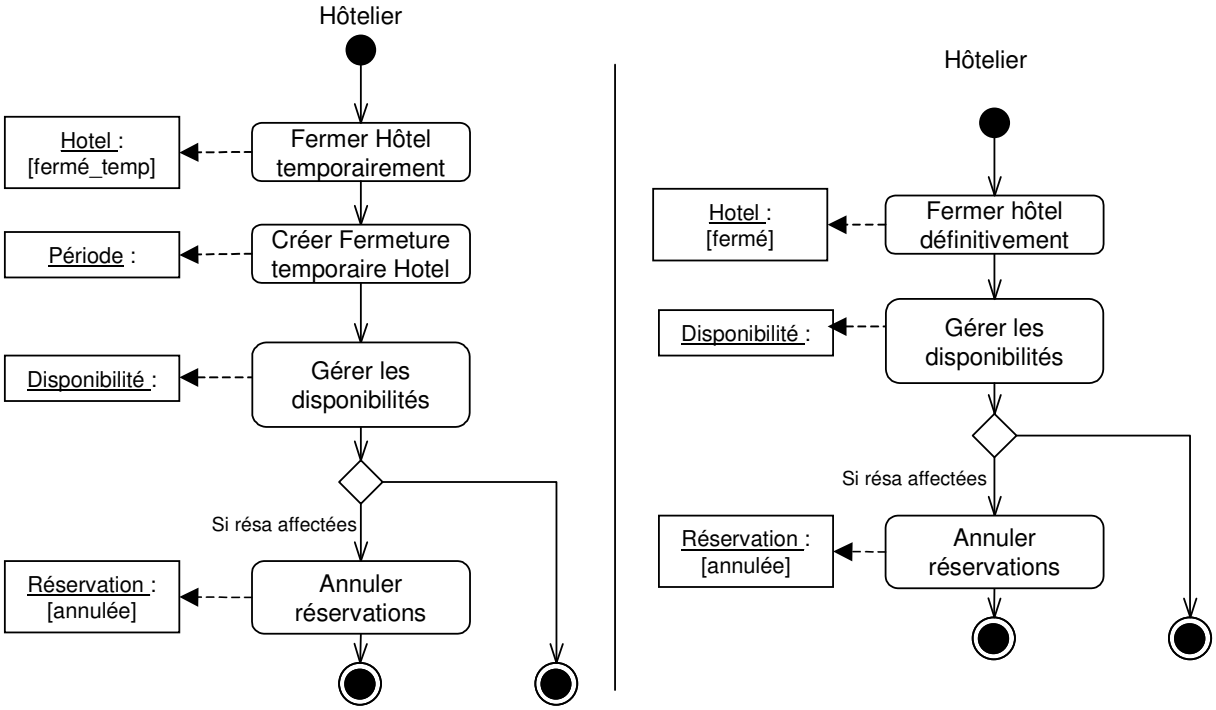


Figure 152 : Processus de fermeture temporaire (à gauche) et définitive (droite) d'un hôtel

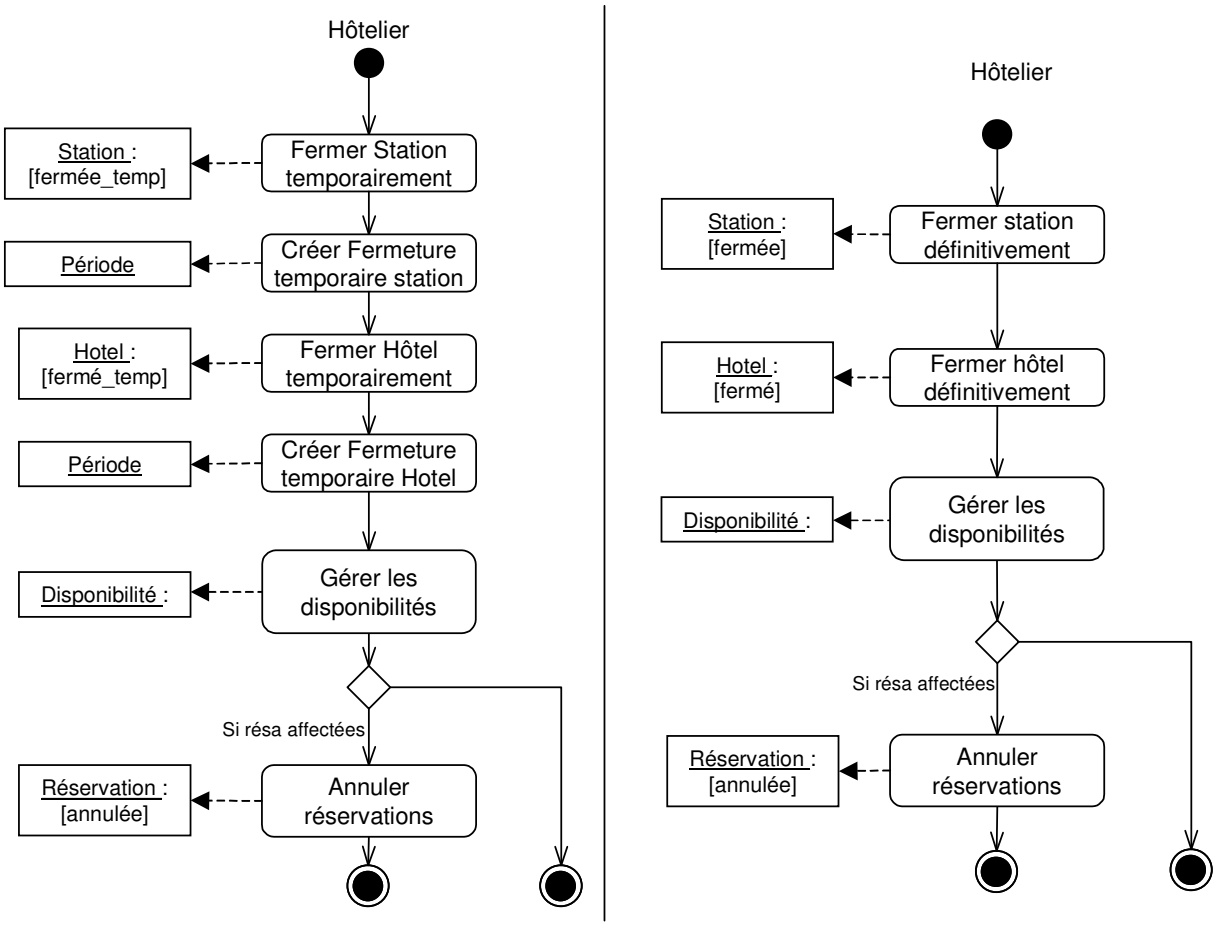


Figure 153 : Processus de fermeture temporaire (à gauche) et définitive (droite) d'une station

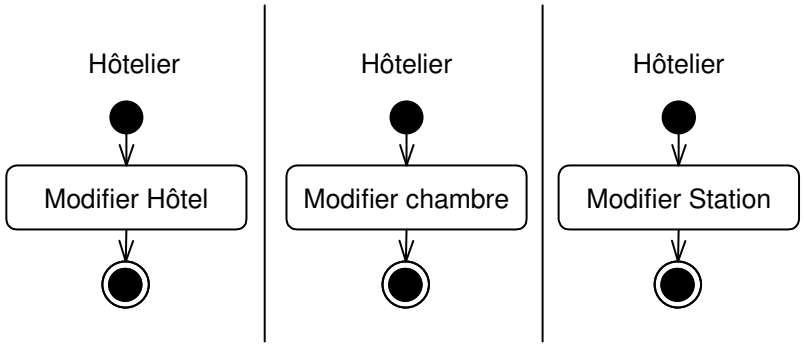


Figure 154 : Processus de modification d'un Hôtel (à gauche), d'une chambre (au centre) et d'une station (à droite)

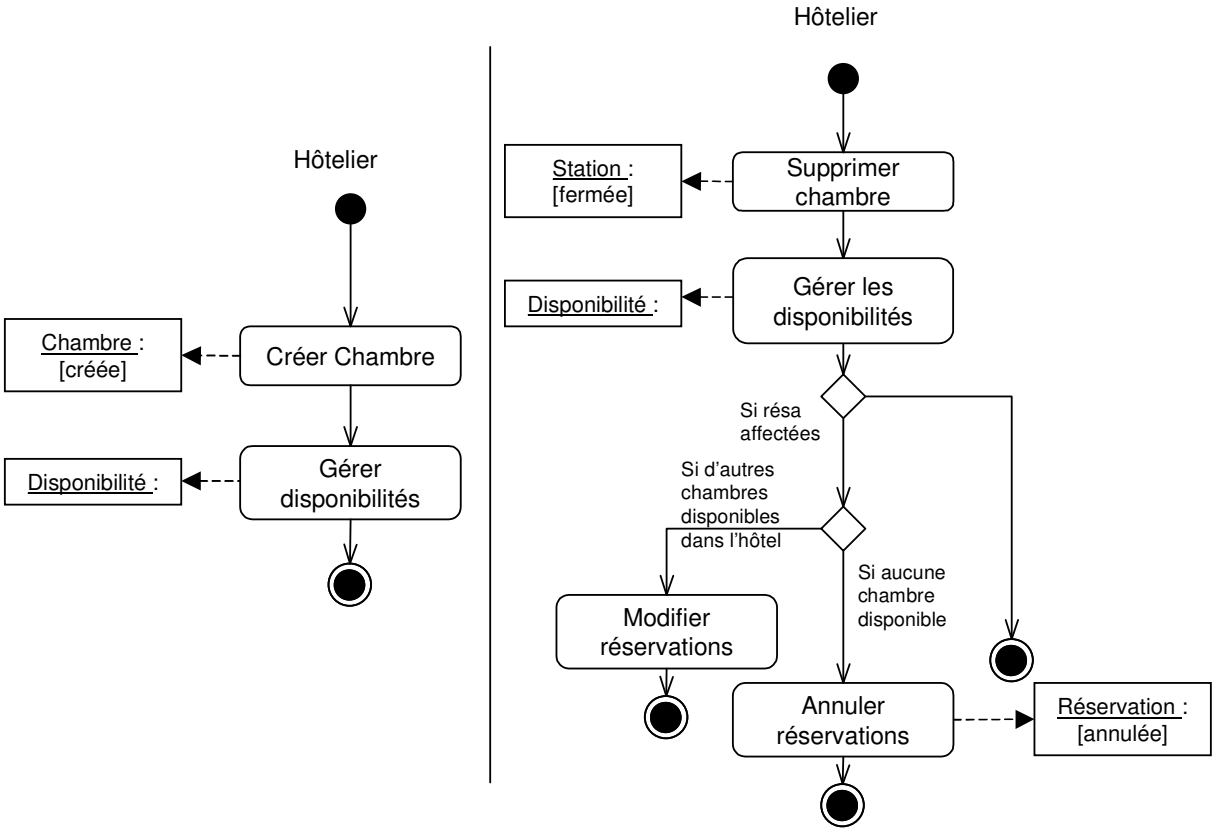
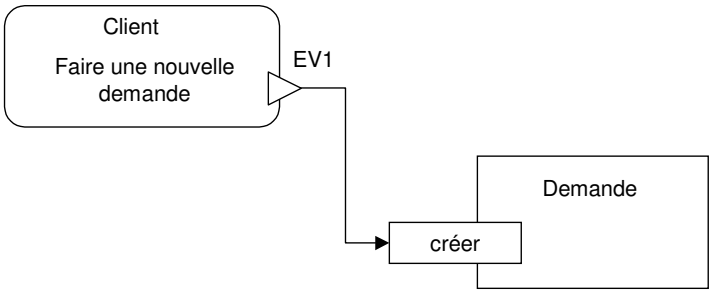


Figure 155 : Processus de création de chambre (à gauche) et de suppression de chambre (à droite)

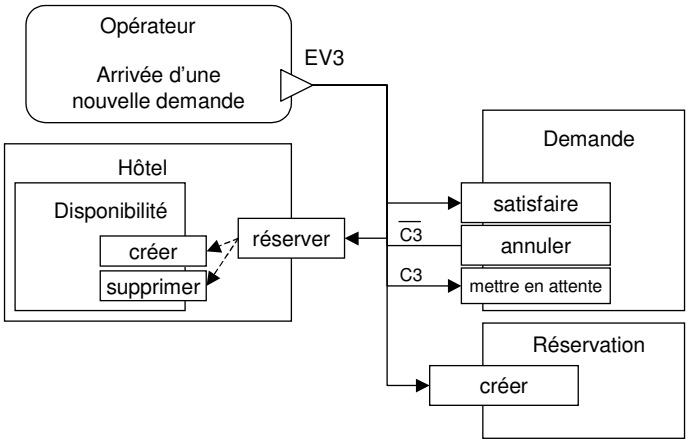
2. Description du système

2.1. Description dynamique



Acteur Client
Evénement
Faire une nouvelle Demande EV1
Message
Prénom : **string**
Nom : **string**
Date de naissance **date**
Adresse **string**
Téléphone : **string**
[Nom Station : **string**]
[Catégorie Hôtel : **string**]
Check in date : **date**
Check out date : **date**
Nombre de Chambres : **entier**
Déclenchement
Créer sur Demande

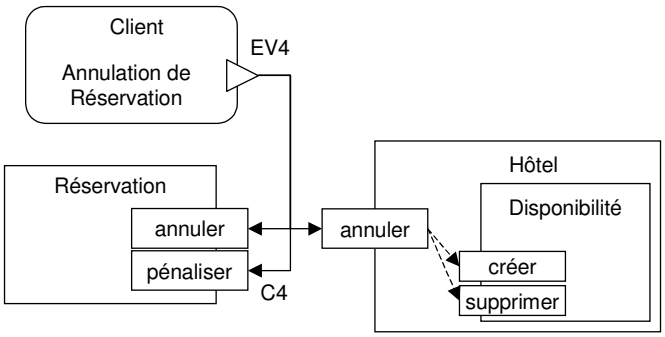
Figure 156 : Description de l'événement EV1



C3 : le client accepte de mettre sa demande en attente

Acteur Opérateur
Evénement
Arrivée d'une nouvelle Demande EV3
Message
Numéro Demande : **entier**
Numéro Hôtel: **entier**
Déclenchement
Créer sur Réservation
Réserver sur Hôtel
Satisfaire sur Demande
Annuler sur Demande si ¬ C3
Mettre en attente sur Demande si C3

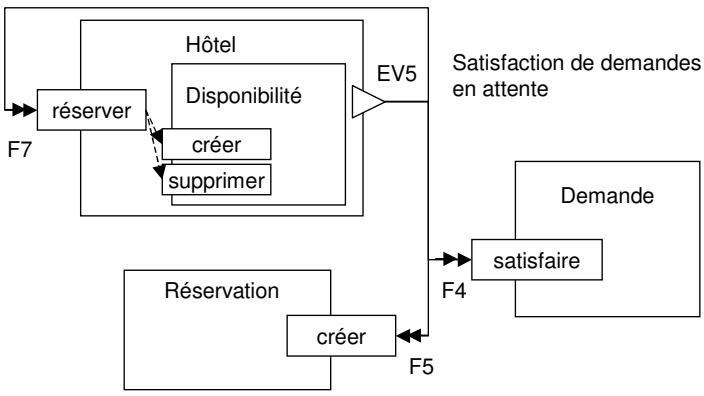
Figure 157 : Description de l'événement EV3



C4 : la réservation débute dans moins d'une semaine

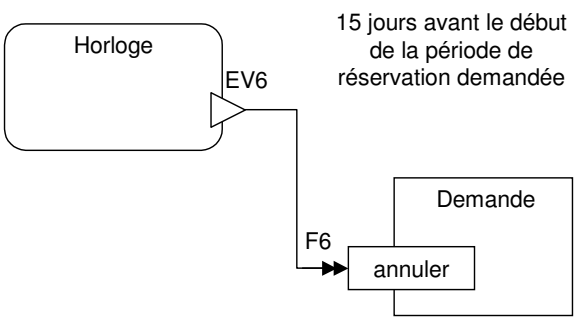
Acteur Client
Evénement
Annulation d'une Réservation EV4
Message
Numéro Réservation : **entier**
Déclenchement
Annuler sur Réservation
Rendre dispo sur Hôtel
Pénaliser sur Réservation ***si***
la réservation commence dans moins d'une semaine

Figure 158 : Description de l'événement EV4



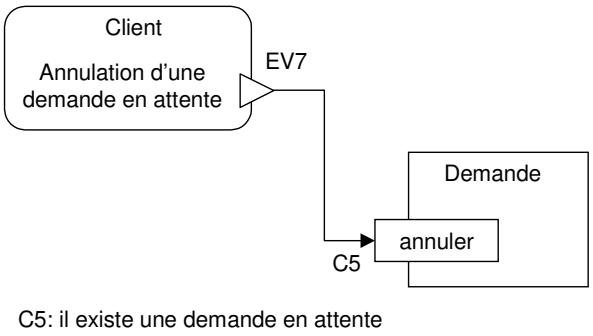
Classe Hôtel
Evénement
Satisfaction de demandes en attente EV5
Prédicat
Prec. Disponibilité.état = supprimée
Nouv. Disponibilité.état = créée
Déclenchement
Créer sur Réservation
Réserver sur Hôtel
Satisfaire sur Demande

Figure 159 : Description de l'événement EV5



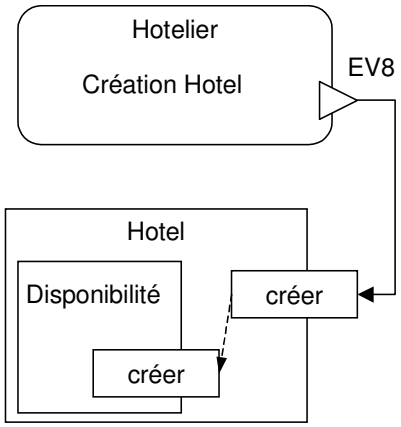
Class Horloge
Evénement
Annulation automatique des demandes en attente EV6
Predicat
Deux semaines avant la période demandée
Déclenchement
Annuler sur Demande ***pour chaque***
Demande en attente arrivant à échéance à J+15

Figure 160 : Description de l'événement EV6



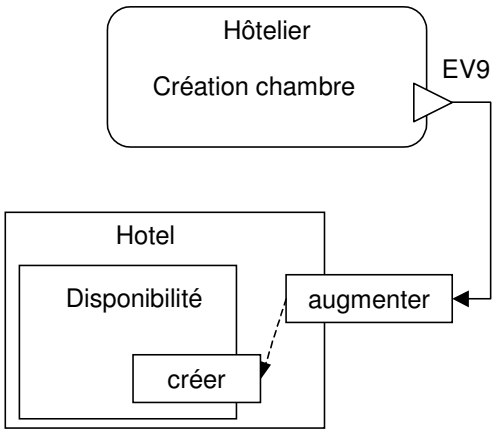
Acteur Client
Evénement
Annulation d'une Demande en attente EV7
Message
Numéro Demande : **entier**
Déclenchement
Annuler sur Demande ***si*** la demande est en attente

Figure 161 : Description de l'événement EV7



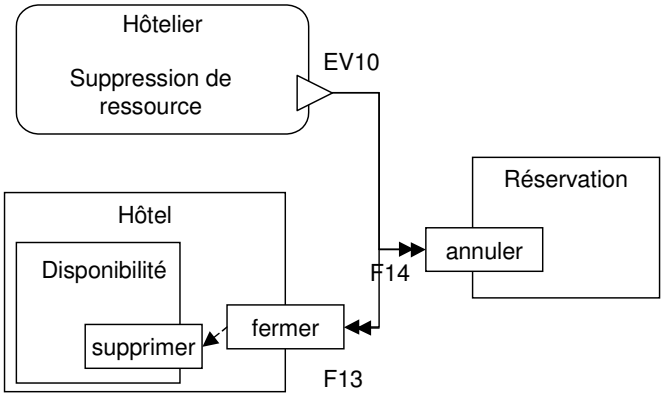
Acteur Hôtelier
Evénement
Création d'hôtel EV8
Message
Nom Hôtel : **string**
Adresse Hôtel : **string**
Catégorie Hôtel : **string**
Déclenchement
Ouvrir sur Hôtel
Créer sur Disponibilité

Figure 162 : Description de l'événement EV8



Acteur Hôtelier
Evénement
Création Chambre EV9
Message
Nombre de chambre : **entier**
Prix chambre : **réel**
Déclenchement
Augmenter sur Hôtel
Créer sur Disponibilité

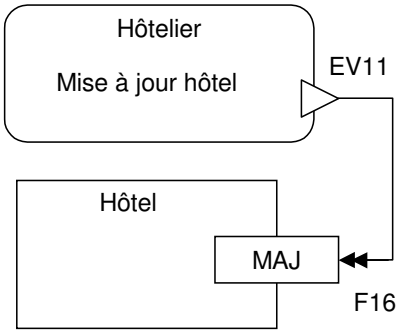
Figure 163 : Description de l'événement EV9



F14: Pour chaque réservation à modifier
F13: pour chaque hôtel à fermer

Acteur Hôtelier
Evénement
Suppression de ressource EV10
Message
Numéro Hôtel : entier
Déclenchement
Fermer sur Hôtel
Annuler sur Réservation

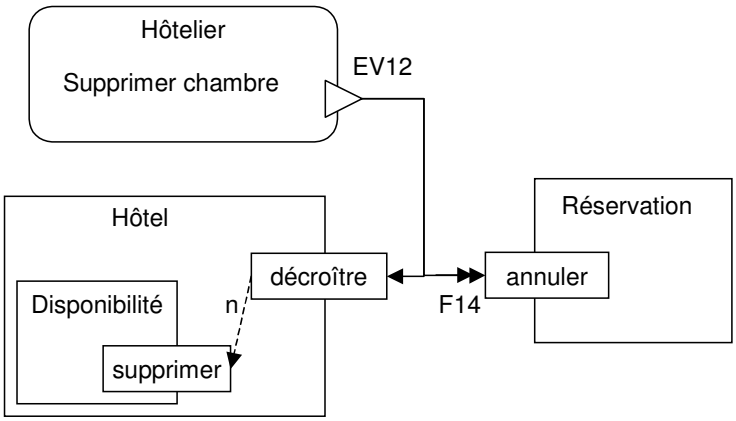
Figure 164 : Description de l'événement EV10



F16: pour chaque hôtel à mettre à jour

Acteur Hôtelier
Evénement
Mise à jour Hôtel EV11
Message
Nom Hôtel : **string**
Adresse Hôtel : **string**
Catégorie Hôtel : **string**
Prix chambre : **liste (réel)**
Déclenchement
Mettre à jour sur Hôtel

Figure 165 : Description de l'événement EV11

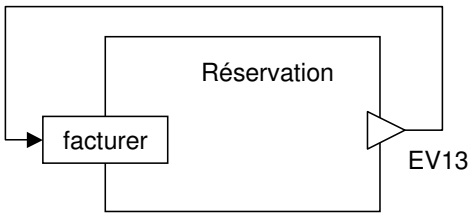


Acteur Hôtelier
Evénement
Suppression de Chambres EV12
Message
Hôtel numéro : entier
Nombre de chambres
Déclenchement
Décroître sur Hôtel
Annuler sur Réservation ***pour chaque*** Réservation concernée par ces Chambres

F14: Pour chaque réservation à modifier

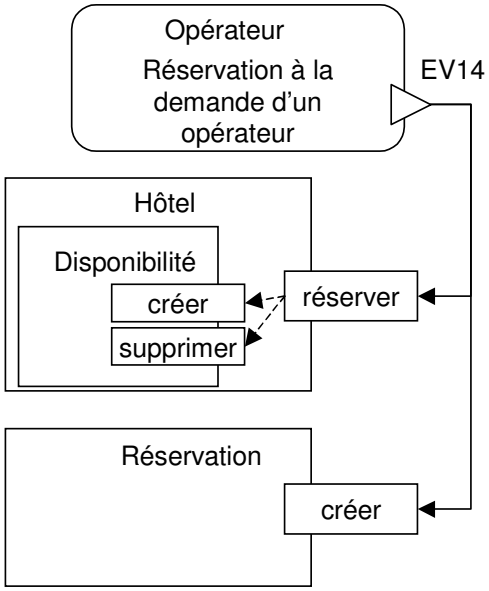
Figure 166 : Description de l'événement EV12

Le système se rend compte qu'une réservation a été créée, elle doit être facturée



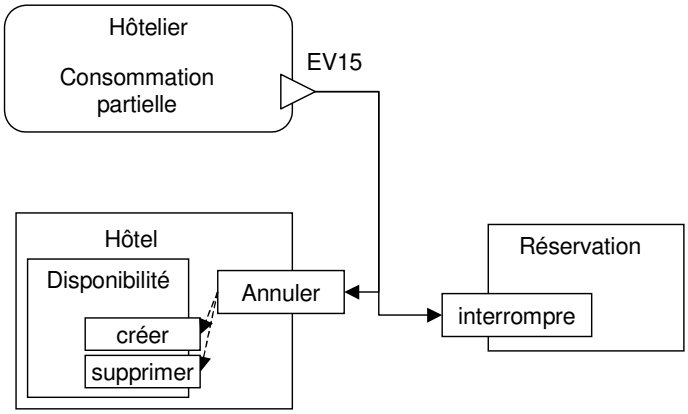
Classe Réservation
Evénement
Facturer Réservation EV13
Prédicat
Préc. état = s0, nouv. état = créée
Déclenchement
Facturer sur Réservation

Figure 167 : Description de l'événement EV13



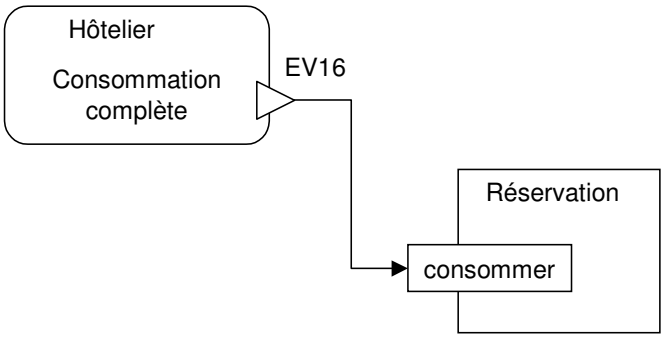
Acteur Opérateur
Evénement
Réservation à la demande d'un opérateur EV14
Message
Prénom : **string**
Nom : **string**
Date de naissance **date**
Adresse **string**
Téléphone : **string**
[Nom Station : **string**]
[Catégorie Hôtel : **string**]
Check in date : **date**
Check out date : **date**
Numéro de Chambres : **entier**
Déclenchement
Créer sur Réservation
Réserver sur Hôtel

Figure 168 : Description de l'événement EV14



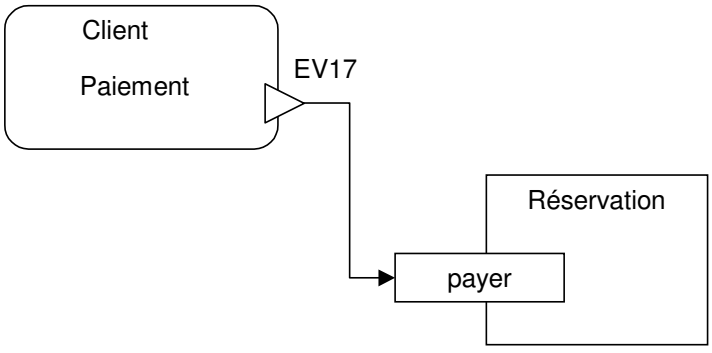
Acteur Hôtelier
Evénement
Consommation partielle EV15
Message
Numéro Hôtel : entier
Numéro Réservation : entier
Déclenchement
Rendre dispo sur Hôtel
Interrompre sur Réservation
Pénaliser sur Réservation

Figure 169 : Description de l'événement EV15



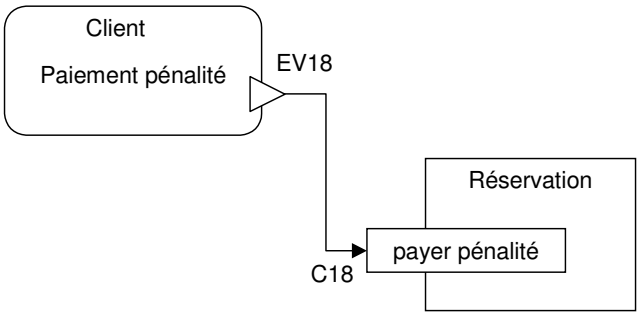
Acteur Hôtelier
Evénement
Consommation complète EV16
Message
Numéro Hôtel : entier
Numéro Réservation : entier
Déclenchement
Consommer sur Réservation

Figure 170 : Description de l'événement EV16



Acteur Client
Evénement
Paiement EV17
Message
Numéro Réservation : entier
Déclenchement
Payer sur Réservation

Figure 171 : Description de l'événement EV17



Acteur Client
Evénement
Paiement des pénalités EV18
Message
Numéro Réservation : entier
Déclenchement
Payer pénalité sur Réservation ***si*** il existe une pénalité

C18 : Il existe une pénalité

Figure 172 : Description de l'événement EV18

2.2. Description statique

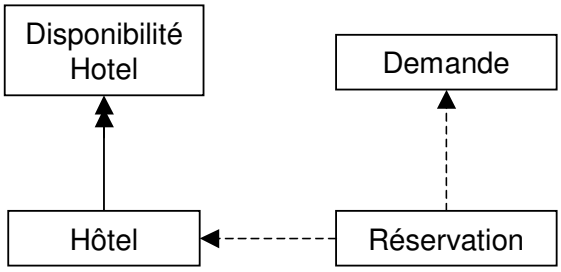


Figure 173 : Scéma de classe O*

Hôtel

Propriété

Nom : string
Adresse : string
Station : string
Catégorie : string
Disponibilité : set(comp(DisponibilitéHotel))
Prix des chambres : {prix des chambres}
Nombre chambre

Opérations

Ouvrir (créer un hôtel)
Fermer (fermer un hôtel)
Mettre à jour (mettre à jour les informations concernant un hôtel)
Réserver (réserver une chambre de l'hôtel)
Annuler (annuler une réservation)
Augmenter (créer une nouvelle chambre dans l'hôtel)
Décroître (enlever une chambre de l'hôtel)

Graphe de transition d'état

{ouvert, fermé provisoirement, fermé}

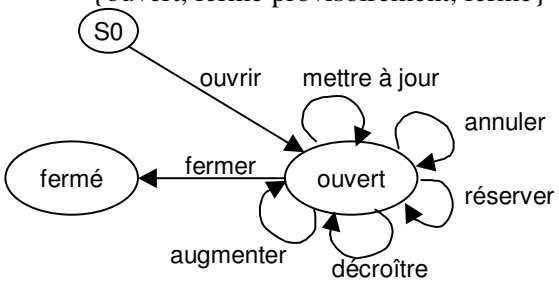


Figure 174 : Description de la classe objet Hôtel

DisponibilitéHotel

Propriété

DateDeb : date
DateFin : date

Opération

Créer
Supprimer

Graphe de transition d'état

{créée, supprimée}

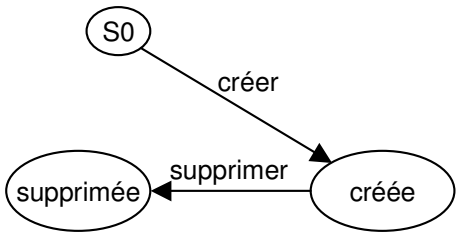


Figure 175 : Description de la classe objet DisponibilitéHôtel

Demande

- Propriétés
  - Numéro : entier
  - DateDeb : date
  - DateFin : date
  - Nb chambres : entier
  - Catégorie : string
  - Station : string
  - PeutEtreEnAttente : booléen
- Opérations
  - Créer
  - Satisfaire (la demande)
  - Annuler
  - Mettre attente
- Graphe de transition d'état
  - {créée, satisfaite, annulée, attente}

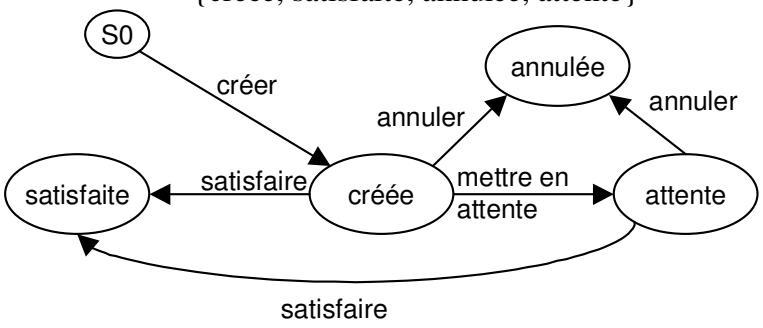


Figure 176 : Description de la classe objet Demande

Réservation

Propriétés

Numéro : entier
DateDeb : date
DateFin : date
Nb chambres : entier
Nb personnes : entier
PrénomClient : string
NomClient : string
AdresseClient : string
Pénalité : réel
Demande : ref(Demande)
Hôtel : ref(Hôtel)

Opérations

Créer
Pénaliser
Annuler
Interrompre (le client quitte l'hôtel avant la date de départ prévue)
Consommer (le client quitte l'hôtel à la date de départ qui était prévue)
Facturer (émettre la facture correspondant à la réservation)
Payer
Payer pénalité

Graphe de transition d'état

{créée, facturée, annulée, consommée, partiel_conso, payée}

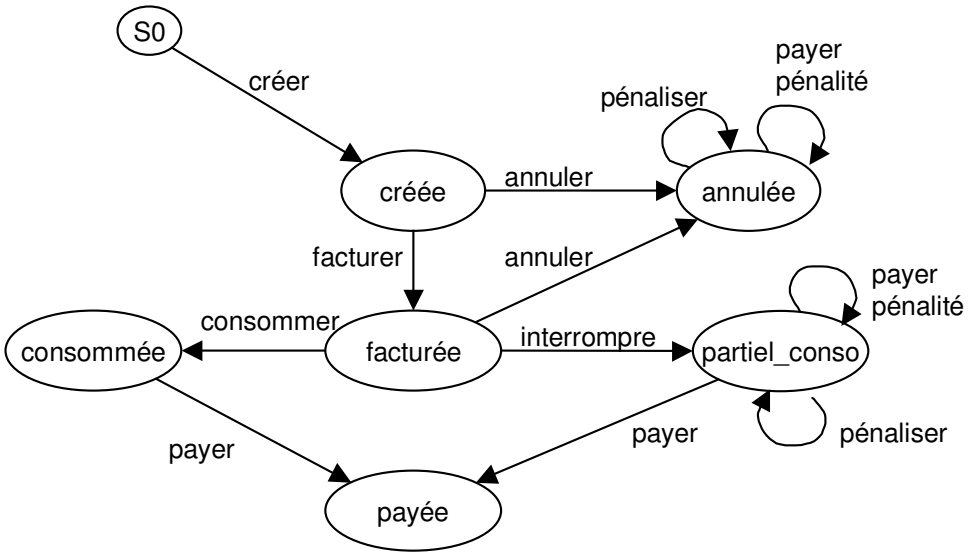


Figure 177 : Description de la classe objet Réservation

3. Calcul des métriques

Cette section présente les valeurs des métriques *Complétude de l'activité* et *Exactitude de l'activité* pour chacune des activités.

Activité	Complétude de l'activité	Exactitude de l'activité
Effectuer une demande	1	1
Mettre en attente	1	1
Mettre une demande hors délai	1	1/2 = 0,5
Mettre une demande sans suite	1	1/2 = 0,5
Satisfaire demande	1	1
Créer réservation	1	1
Gérer les disponibilités	0	0
Modifier réservation	1/3 = 0,33	0
Facturer	1	1
Annuler Réservation par le client	1	1
Facturer pénalité annulation	1	1
Payer pénalité	1	1
Consommer	1	1
Facturer pénalité dû à une consommation partielle	1	1
Payer Réservation	1	1
Modifier Hotel	0	0
Modifier Station	0	0
Modifier Chambre	0	0
Fermer hotel définitivement	0	0
Fermer station définitivement	0	0
Annuler réservation	1	1
Supprimer chambre	0	0
Créer Hotel	0	0
Créer Chambre	0	0
Créer Fermeture temporaire station	0	0
Créer Fermeture temporaire Hotel	0	0
Fermer Station temporairement	0	0
Créer nouvelles réservations par l'opérateur	1	1
Fermer Hotel Temporairement	0	0
Créer station	0	0

Tableau 46 : Complétude de l'activité et Exactitude de l'activité

Ce tableau permet de visualiser que :

- pour 13 des 30 activités (soit 43%) l'information qu'elles manipulent est gérée de la même façon par le système.
- les activités concernant la gestion des réservations sont majoritairement mieux gérées que les activités de gestion des produits
- la *Complétude de l'activité* et/ou l'*Exactitude de l'activité* est souvent nulle ce qui peut expliquer pourquoi certaines activités ne sont pas gérées par le système. Comme cela a été vu dans le corps du texte, ce n'est pas la seule raison. En effet, l'activité 'créer chambre' est prise en charge par le système mais sa *Complétute* et son *Exactitude* sont nulles.

ANNEXE 2

1. Description du As-Is

Cette section présente le modèle pivot dans son ensemble avant l'exécution des opérateurs d'écart.

1.1. Carte Cu de haut niveau

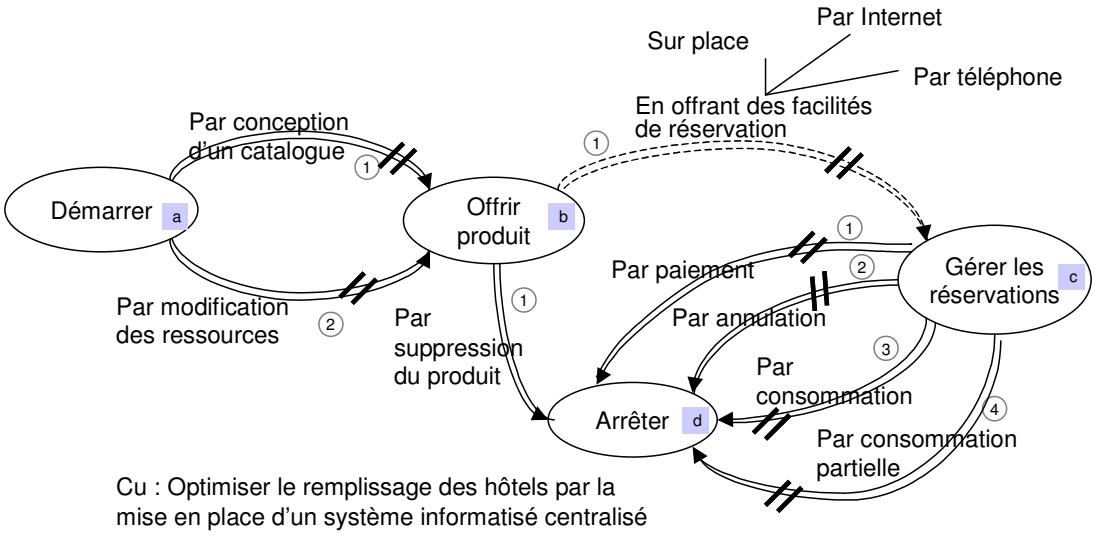


Figure 178 : Carte As-Is unifiée de haut niveau

1.2. Carte affinat Cu.ab1

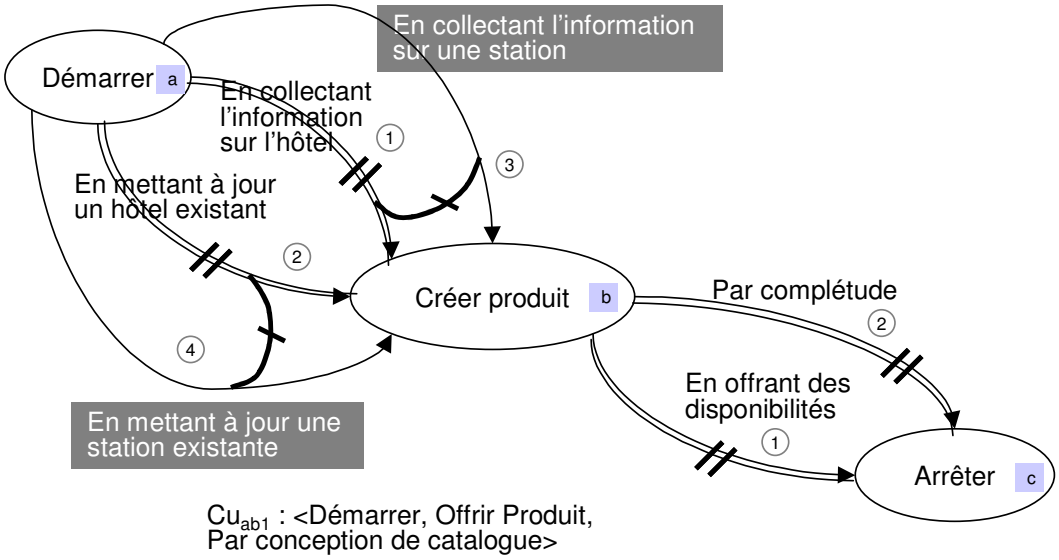


Figure 179 : Carte As-Is unifiée affinant Cu.ab1

Code	Intention	états
a	Démarrer	Station.état="ouverte", Hôtel.état="ouvert"
b	Créer produit	Hôtel.état="ouvert", Station.état="ouvert", Chambre.étatCh="libre"
c	Arrêter	new.Disponibilité, new.DisponibilitéHôtel

Tableau 47: Description des intentions de la carte Cu_{ab1}

Code	Pré-condition	Post-condition	Acteur	Ressource	Règle de gestion
ab1	Station	Hôtel.état="ouvert" AND Station.état="ouvert" AND Chambre.état="ouverte"	Hôtelier	Station	<-, Hôtel.état="ouvert">, <-, new.Disponibilité>, <-, new.DisponibilitéHôtel>, <-, chambre.état="ouverte">
ab2	Hôtel	Hôtel.état="ouvert"	Hôtelier	-	<old.Hôtel.adresse, new.Hôtel.adresse> OR <old.Hôtel.categorie, new.Hôtel.categorie>
ab3	-	Station.état="ouvert"	Hôtelier	-	<-, Station.état="ouvert">
ab4	Station	Station.état="ouvert"	Hôtelier	-	<old.Station.géographie, new.Station.géographie> OR <old.Station.categorie, new.Station.categorie> OR <old.Station.description, new.Station.description>
bc1	Hôtel.état="ouvert"	Chambre AND Disponibilité DisponibilitéHôtel	Hôtelier	Hôtel, Station, Chambre	<old.DisponibilitéHotel, new.DisponibilitéHôtel> OR <old.Disponibilité, new.Disponibilité>
bc2	Hôtel.état="ouvert"	-	Hôtelier	-	-

Tableau 48 : Description des sections de la carte Cu_{ab1}

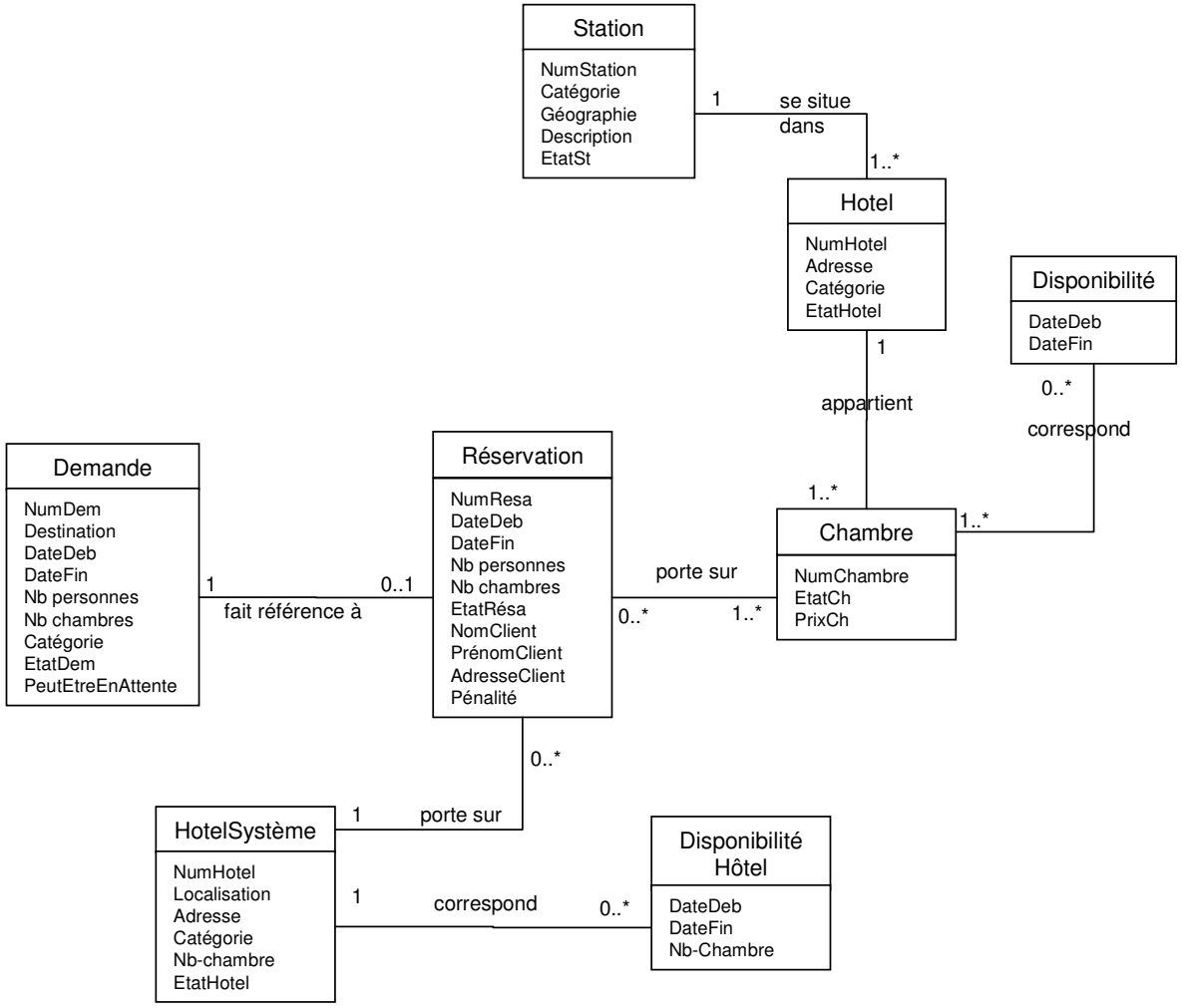
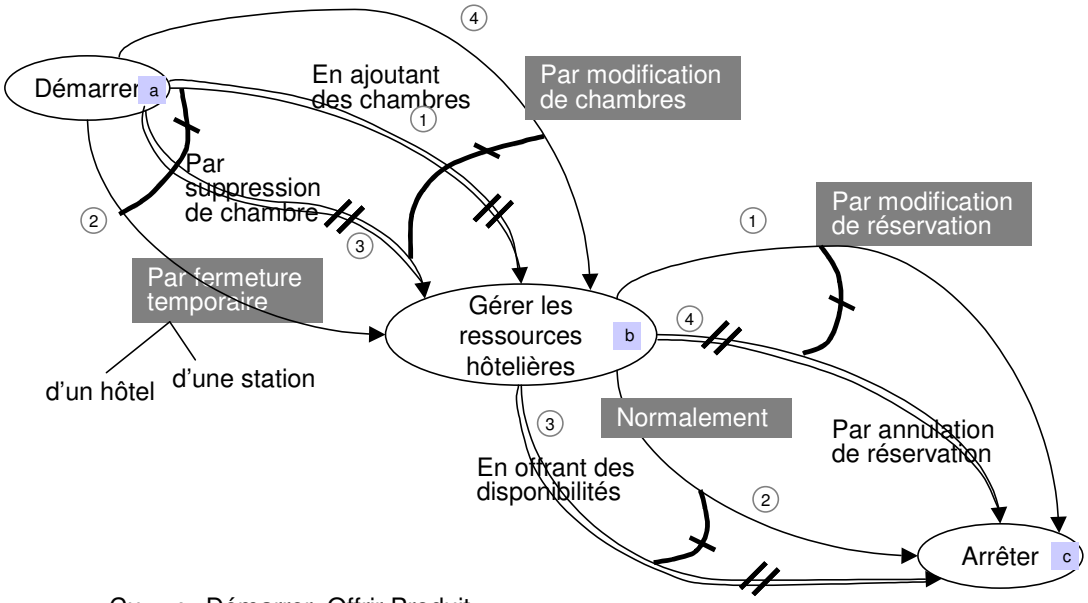


Figure 180 : Modèle de classes correspondant à la carte pivot Cu_{ab1}

1.3. Carte affinant Cu.ab2



Cu_{ab2} : <Démarrer, Offrir Produit,
Par modification des ressources>

Figure 181 : Carte As-Is unifiée affinant Cu.ab2

Code	Intention	états
a	Démarrer	Hôtel.état="ouvert", Hôtel.état="fermé_temp"
b	Gérer les ressources Hôtelières	Station.état="ouverte", Station.état="fermée_temp", Hôtel.état="ouvert", Hôtel.état="fermé_temp", Chambre.état="fermée", old.Chambre.prixch, new.Chambre.prixch, old.Hôtel.nbchambres, new.Hôtel.nbchambres, Réservation.état="créée", Réservation.état="facturée", new.Période
c	Arrêter	old.Réservation.Chambre, new.Réservation.Chambre, old.Réservation.Hôtel, new.Réservation.Hôtel, new.Disponibilité, Réservation.état="créée", Réservation.état="annulée" Réservation.état="facturée", new.DisponibilitéHôtel

Tableau 49 : Description des intentions de la carte pivot Cu_{ab2}

Code	Pré-condition	Post-condition	Acteur	Ressource	Règle de gestion
ab1	Hôtel,	Chambre, Hôtel.nb-chambre= Hôtel.nb-chambre+1	Hôtelier	Hôtel, Station	<-, Chambre.état="ouverte">, <old.Hôtel.nbchambres, new.Hôtel.nbchambres>
ab2	Hôtel OR Station	Hôtel.état="fermé_temp" AND Station.état="fermée_temp" AND Chambre.état="fermée" AND new.Période	Hôtelier	-	[<-, new.Période> AND <Station.état="ouverte", Station.état="fermée_temp">] OR [<-, new.Période> AND <Hôtel.état="ouvert", Hôtel.état="fermé_temp">]
ab3	Hôtel	Chambre.état="fermée", Hôtel.nb-chambre= Hôtel.nb-chambre-1	Hôtelier	Hôtel, Station	<Chambre.état="ouverte", Chambre.état="fermée">, <old.Hôtel.nbchambres, new.Hôtel.nbchambres>
bc1	Réservation.état <= "facturée" AND Chambre.état="fermée" AND Chambre1.état="ouverte" AND Disponibilité.DateDeb <= Demande.DateDeb AND Disponibilité.DateFin >= Demande.DateFin	Réservation.état="créée"	Hôtelier	Hôtel, Station	<old.Réservation.Chambre, new.Réservation.Chambre>
bc2	Chambre.état="fermée" AND Non Réservation.état<=facturée	rien	Hôtelier	Hôtel, Station	-
bc3	Chambre.état="fermée" new.Hôtel.nb-chambre<>old Hôtel.nb-chambre	<-, new.Disponibilité> OR <-, new.DisponibilitéHôtel>	Hôtelier	Hôtel, Station	<old.Disponibilité, new.Disponibilité> OR <old.DisponibilitéHôtel, new.DisponibilitéHôtel>
bc4	Réservation.état <= "facturée" AND Chambre.état="fermée" AND NON (Chambre1.état="ouverte" AND Disponibilité.DateDeb <= Demande.DateDeb AND Disponibilité.DateFin >= Demande.DateFin) AND new.Hôtel.nb-chambre<old Hôtel.nb-chambre	Réservation.état="annulée"	Hôtelier	Hôtel, Station	<Réservation.état="créée", Réservation.état="annulée"> OR <Réservation.état="facturée", Réservation.état="annulée">
ab4	Chambre	Chambre	Hôtelier	Hôtel, Station	<old.Chambre.prixch, new.Chambre.prixch>

Tableau 50 : Description des sections de la carte pivot Cu_{ab2}

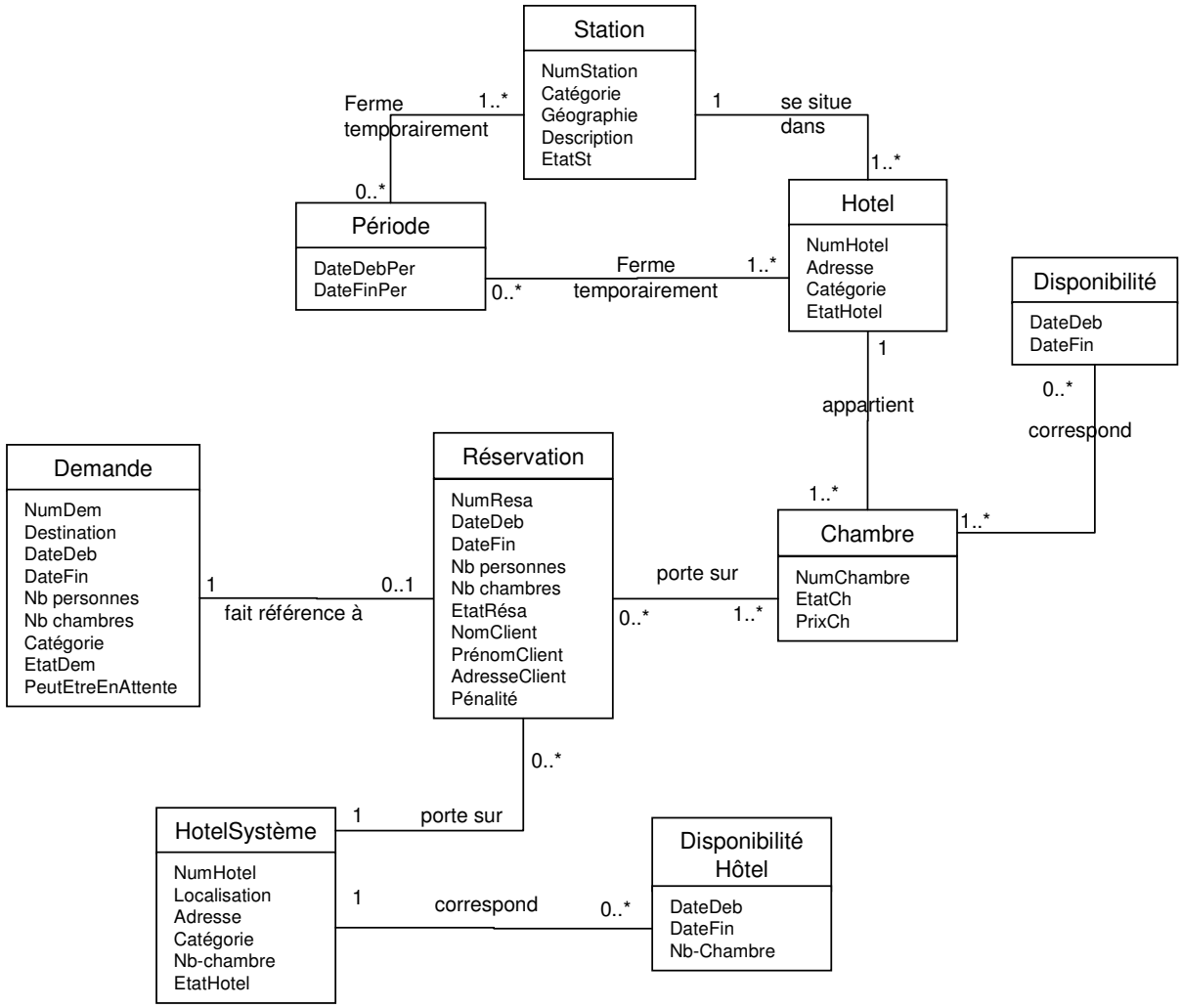


Figure 182 : Modèle de classes correspondant à la carte pivot Cu_{ab2}

1.4. Carte affinant Cu.bc1

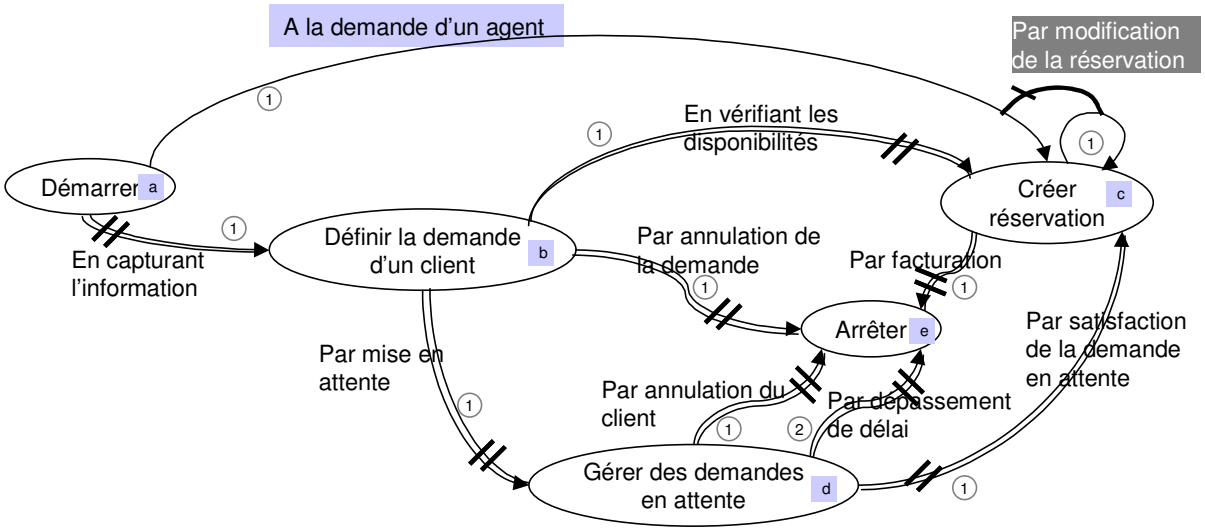


Figure 183 : Carte As-Is unifiée affinant Cu.bc1

Code	Intention	états
a	Démarrer	Station.état="ouverte", Hôtel.état="ouvert"
b	Définir la Demande d"un client	Demande.état = "créée"
d	Gérer des Demandes en attente	Demande.état="attente"
c	Créer Réservation	Demande.état="satisfaite", Réservation.état="créée", new.Disponibilité new.DisponibilitéHôtel
e	Arrêter	Réservation.état="facturée", Demande.état="annulée", Demande.état="sans_suite", Demande.état="hors delai"

Tableau 51 : Description des intentions de la carte pivot Cu_{bc1}

Les propriétés des sections sont spécifiées au Tableau 42.

Code	Pré-condition	Post-condition	Acteur	Ressource	Règle de gestion
ab1	Hôtel, Station	Demande.état = "créée"	Client	Hôtel	<-, Demande.état="créée">
de1	Demande.état="en attente" AND Demande.DateDeb > date-du-jour + 7j	Demande.état= "annulée"	Client	Hôtel	<Demande.état="attente", Demande.état="annulée">
bd1	Demande.état="créée" AND Demande.PeutEtreEnAttente = vrai AND (Disponibilité.DateDeb > Demande.DateDeb OR Disponibilité.DateFin < Demande.DateFin) (DisponibilitéHôtel.DateDeb > Demande.DateDeb OR DisponibilitéHôtel.DateFin < Demande.DateFin)	Demande.état="en attente"	Opérateur	Hôtel	<Demande.état="créée", Demande.état="attente">
be1	Demande.état="créée" AND (Demande.PeutEtreEnAttente = faux AND (Disponibilité.DateDeb > Demande.DateDeb OR Disponibilité.DateFin < Demande.DateFin))	Demande.état="sans_suite" Demande.état="annulée"	Opérateur	Hôtel	<Demande.état="créée", Demande.état="sans_suite"> <Demande.état="attente", Demande.état="annulée">
dc1	Demande.état="en attente" AND Disponibilité.DateDeb <= Demande.DateDeb AND Disponibilité.DateFin >= Demande.DateFin OR DisponibilitéHôtel.DateDeb <= Demande.DateDeb AND DisponibilitéHôtel.DateFin >= Demande.DateFin	Demande.état="satisfaite" AND Reservation.état="créée"	-	Hôtel	<Demande.état="en attente", Demande.état="satisfaite"> AND [<old.DisponibilitéHotel, new.DisponibilitéHôtel> OR <old.Disponibilité, new.Disponibilité>]
bc1	Demande.état="créée" AND Disponibilité.DateDeb <= Demande.DateDeb AND Disponibilité.DateFin >= Demande.DateFin OR DisponibilitéHôtel.DateDeb <= Demande.DateDeb AND DisponibilitéHôtel.DateFin >= Demande.DateFin	Demande.état="satisfaite" AND Reservation.état="créée"	Opérateur	Hôtel	<Demande.état="créée", Demande.état=satisfaite"> AND <-, Réservation.état="créée">
ce1	Réservation.état="créée"	Réservation.état="facturée"	-	Hôtel	<Réservation.état="créée", Réservation.état="facturée">
de2	Demande.état="en attente" AND Demande.DateDeb < date-du-jour + 7j	Demande.état="hors delai" Demande.état="annulée"	-	Hôtel	<Demande.état="attente", Demande.état="hors delai"> <Demande.état="attente", Demande.état="annulée">
cc1	Réservation.état="créée" OR Réservation.état="facturée"	<old.Réservation.Hôtel, new.Réservation.Hôtel> OR <old.Réservation.Chambre, new.Réservation.Chambre>	Opérateur	Hôtel	<old.Réservation.Hôtel, new.Réservation.Hôtel> OR <old.Réservation.Chambre, new.Réservation.Chambre>
ac1	Réservation.état="créée"	Réservation.état="créée"	Opérateur	Hôtel	<-, Réservation.état="créée">

Tableau 52 : Description des sections de la carte pivot Cu_{bc1}

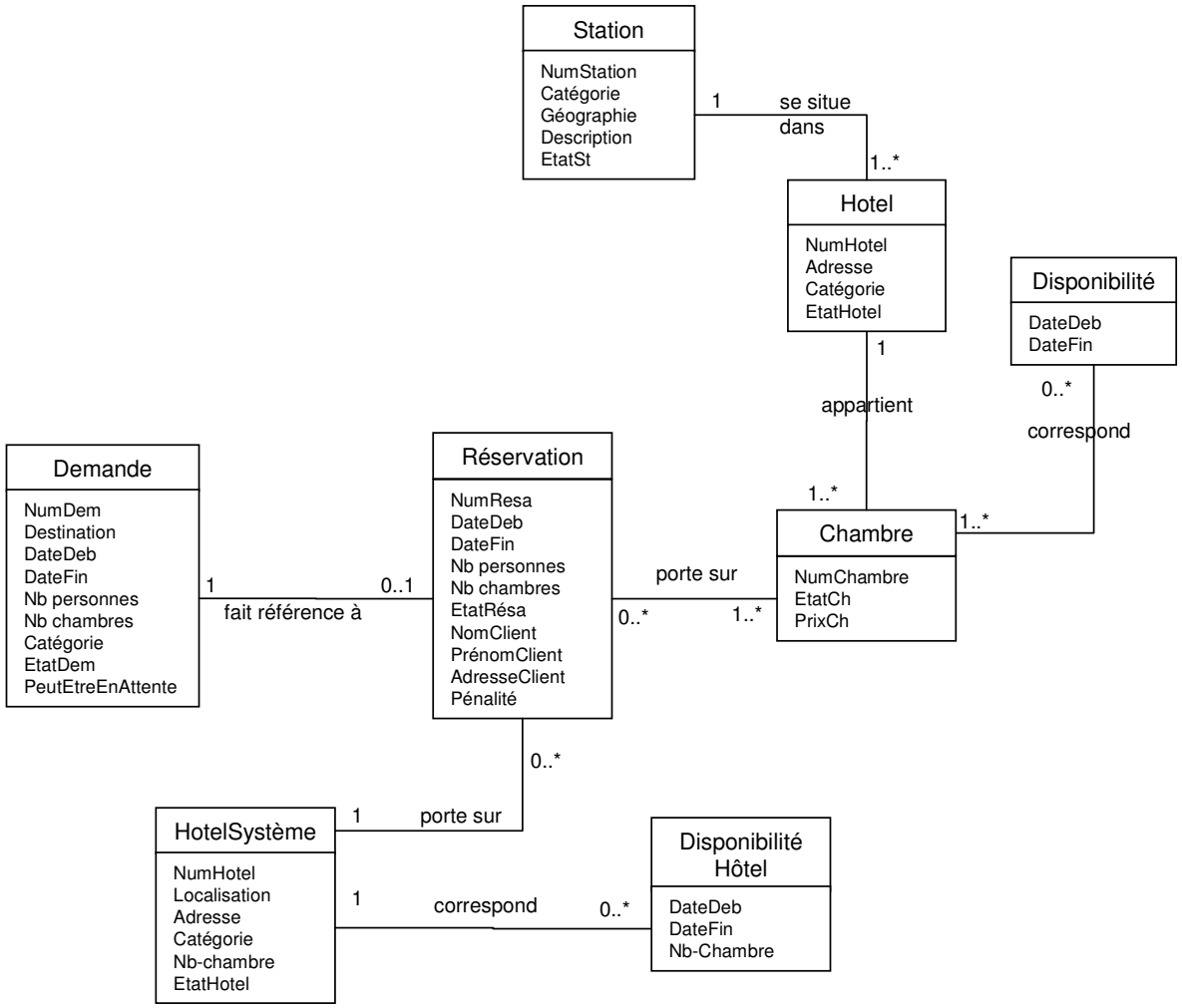


Figure 184 : Modèle de classes correspondant à la carte pivot Cu_{bc1}

1.5. Carte affinant Cu.bd1

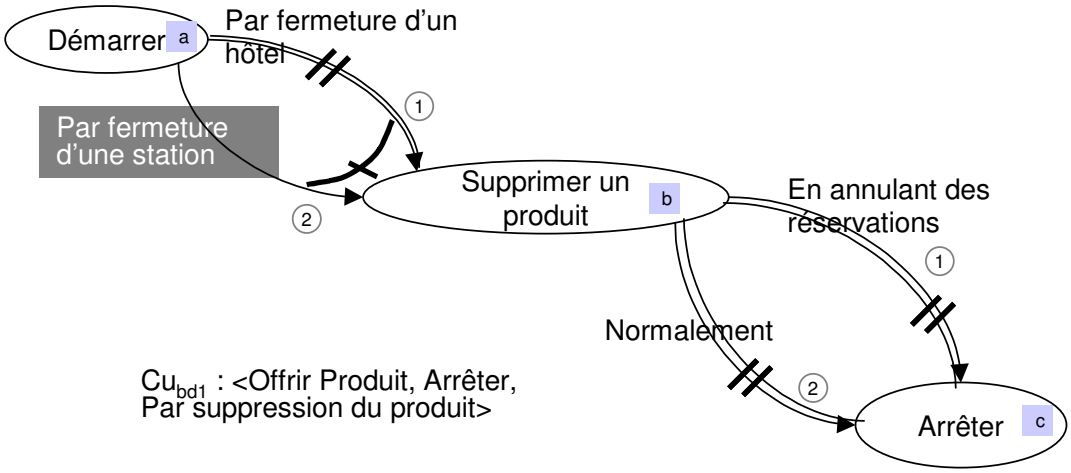


Figure 185 : Carte As-Is unifiée affinant Cu.bd1

Code	Intention	états
a	Démarrer	Hôtel.état="ouvert", Hôtel.état="fermé_temp", Station.état="fermée_temp", Station.état="ouverte"
b	Supprimer un produit	Hôtel.état="fermé", Station.état="fermée", Réservation.état="créée", Réservation.état="facturée"
c	Arrêter	Réservation.état="annulée", new.Disponibilité, new.DisponibilitéHôtel

Tableau 53 : Description des intentions de la carte pivot Cu_{bd1}

Code	Pré-condition	Post-condition	Acteur	Ressource	Règle de gestion
ab1	Hôtel	Hôtel.état="fermé"	Hôtelier	Station	<Hôtel.état="ouvert", Hôtel.état="fermé">
ab2	Station	Station.état="fermée"	Hôtelier	-	<Station.état="ouverte", Station.état="fermée">
bc1	Hôtel.état="fermé" OR Station.état="fermée"	Disponibilité DisponibilitéHôtel	Hôtelier	-	<old.DisponibilitéHotel, new.DisponibilitéHôtel> OR <old.Disponibilité, new.Disponibilité>
bc2	Réservation.état <= "facturée" AND Hôtel.état="fermé"	Réservation.état="annulée"	Hôtelier	Station, Hôtel, Chambre	<Réservation.état="créée", Réservation.état="annulée"> OR <Réservation.état="facturée", Réservation.état="annulée">

Tableau 54 : Description des sections de la carte pivot Cu_{bd1}

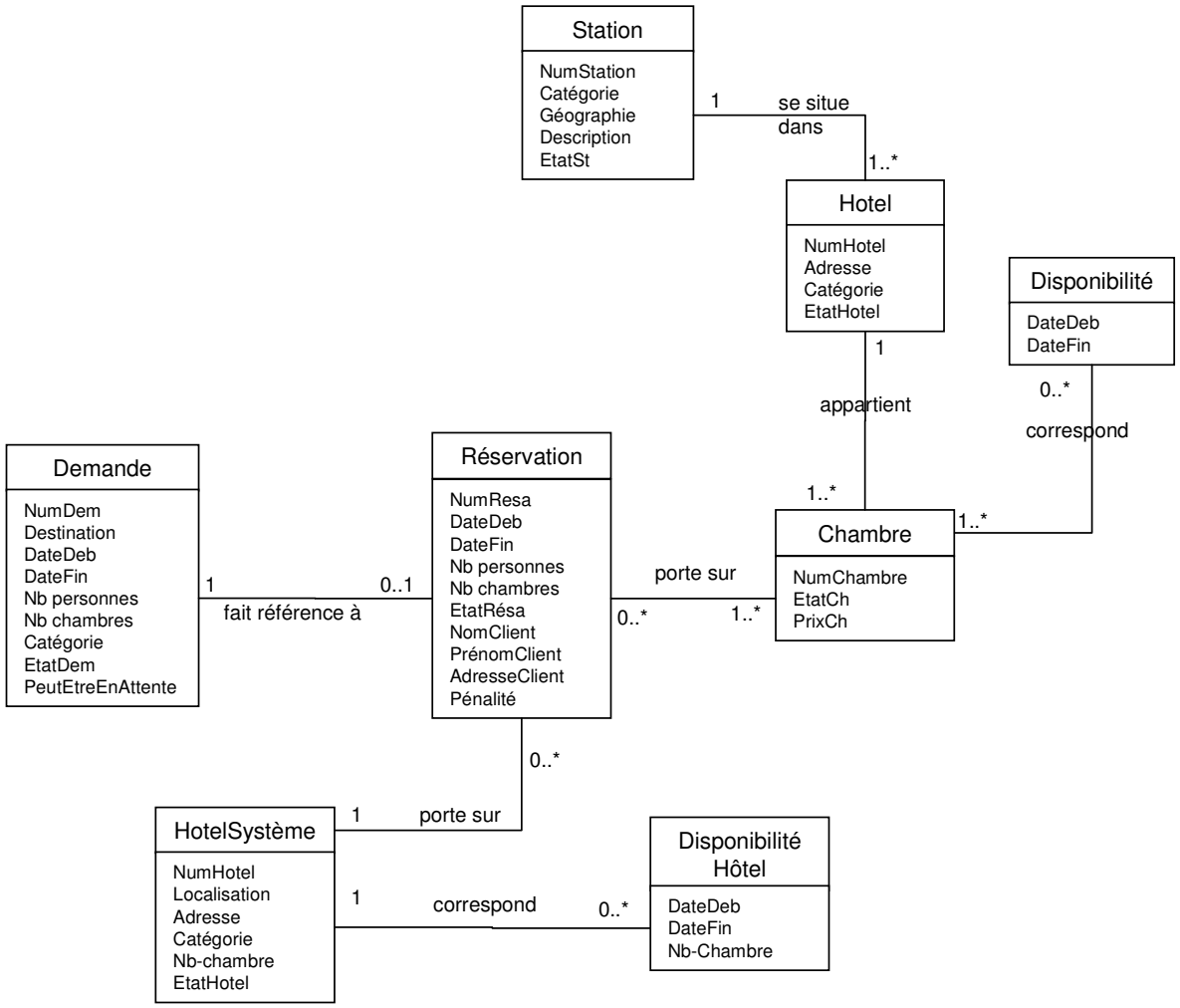


Figure 186 : Modèle de classes correspondant à la carte pivot Cu_{bd1}

1.6. Carte affinant Cu.cd2

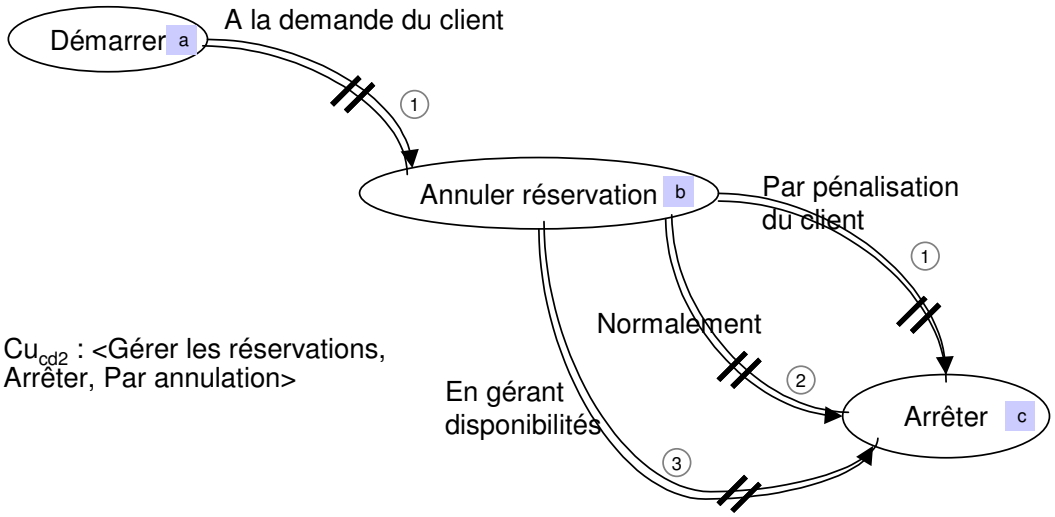


Figure 187 : Carte As-Is unifiée affinant Cu.cd2

Code	Intention	états
a	Démarrer	Réservation.état="créée", Réservation.état="facturée"
b	Annuler Réservation	Réservation.état="annulée"
c	Arrêter	new.Disponibilité, new.DisponibilitéHôtel, new.Réservation.Pénalité

Tableau 55 : Description des intentions de la carte pivot Cu_{cd2}

Code	Pré-condition	Post-condition	Acteur	Ressource	Règle de gestion
ab1	Réservation.état>="créée"	Réservation.état="annulée"	Client	-	<Réservation.état="créée", Réservation.état="annulée"> OR <Réservation.état="facturée", Réservation.état="annulée">
bc1	Réservation.état="annulée" AND Réservation.DateDeb > date-du-jour + 7j	new.Réservation.Pénalité> old.Réservation.Pénalité	Opérateur		<old.Réservation.Pénalité, new.Réservation.Pénalité>
bc2	Réservation.état="annulée" AND Réservation.DateDeb < date-du-jour + 7j	rien	-	-	-
bc3	Réservation.état="annulée"	Disponibilité	Opérateur	-	<old.DisponibilitéHotel, new.DisponibilitéHôtel> OR <old.Disponibilité, new.Disponibilité>

Tableau 56 : Description des sections de la carte pivot Cu_{cd2}

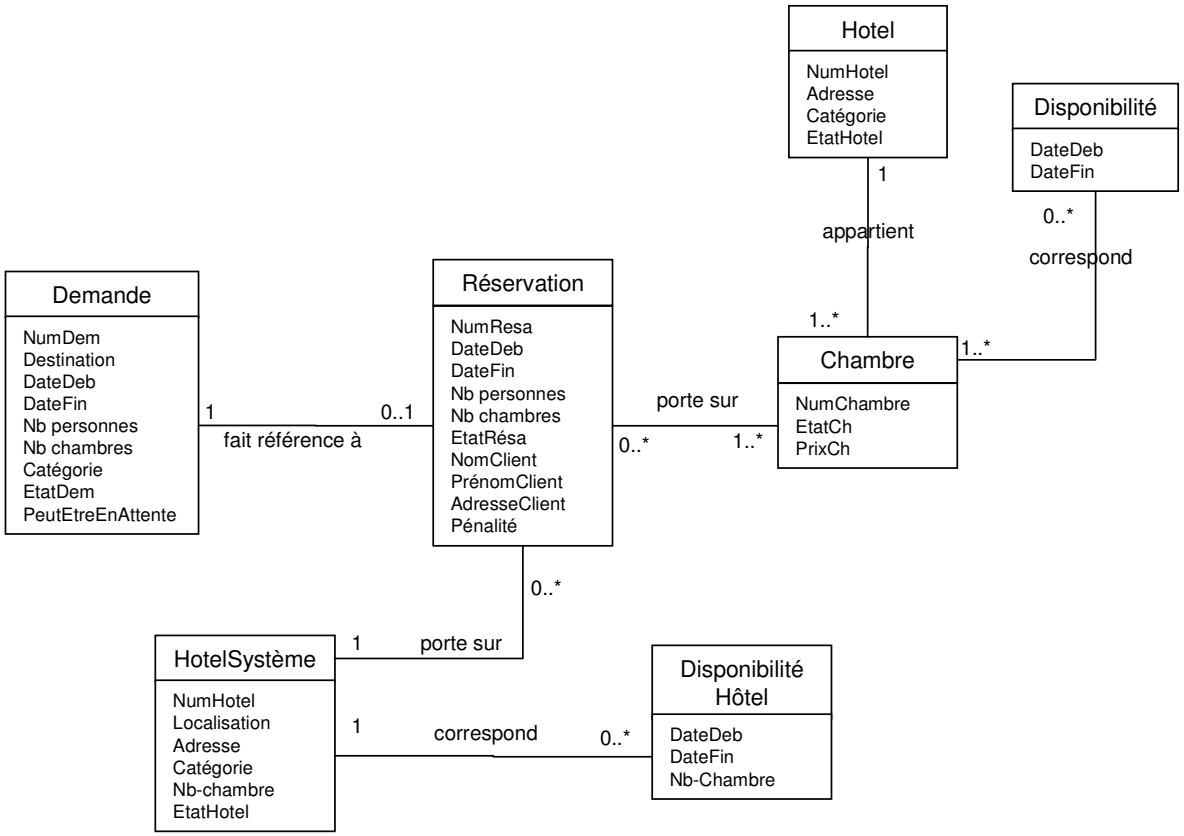


Figure 188 : Modèle de classes correspondant à la carte pivot Cu_{cd2}

2. Exigences d'évolution

Carte Cu			
Code	**Operateur**	**As-Is élément**	**To-Be élément**
Cu-1	*RemplacerIntention*	*Offrir produit*	*Offrir des packages*
Cu-2	*RemplacerStratégie*	*Par conception d'un catalogue*	*Par conception de package*
Cu-3	*RemplacerStratégie*	*Par modification des ressources*	*Par gestion des ressources*
Cu-4	*RemplacerStratégie*	*Par suppression du produit*	*Par suppression d'un package*
Cu-5	*JoindrePrécondition*	-	*Package*
Cu-6	*JoindrePrécondition*	-	*Activité*
Cu-7	*AjouterClasse*	-	*Package*
Cu-8	*AjouterClasse*	-	*Activité*
Cu-9	*AjouterClasse*	-	*DisponibilitéActivité*
Cu-10	*RenommerClasse*	*Disponibilité*	*DisponibilitéChambre*
Cu-11	*RemplacerIntention*	*Gérer les réservations*	*Gérer la relation client*
Cu-12	*ChangerIntentionSource*	*Par annulation*	*Par annulation*
Cu-13	*ChangerIntentionSource*	*Par consommation*	*Par consommation*
Cu-14	*ChangerIntentionSource*	*Par consommation partielle*	*Par consommation partielle*
Cu-15	*ChangerIntentionSource*	*Par paiement*	*Par paiement*
Cu-16	*AjouterStratégie*	-	*Par exclusion*
Cu-17	*AjouterStratégie*	-	*A la demande du client*
Cu-18	*AjouterStratégie*	-	*Par attraction de la clientèle*
Cu-19	*AjouterStratégie*	-	*Par marketing*
Cu-20	*AjouterStratégie*	-	*En gérant l'information sur les clients*
Cu-21	*AjouterStratégie*	-	*En récompensant le client*
Cu-22	*AjouterStratégie*	-	*Par paiement avec des points de fidélité*
Cu-23	*RetyperSegment*	*Par paiement avec des points de fidélité* et *Par paiement normalement*	*Par paiement avec des points de fidélité* et *Par paiement normalement*
Cu-24	*AjouterStratégie*	-	*Par Internet*
Cu-25			
Cu-26	*AjouterClasse*	-	*Client*
Cu-27	*SupprimerAttribut*	*NomClient*	-
Cu-28	*SupprimerAttribut*	*PrénomClient*	-
Cu-29	*SupprimerAttribut*	*AdresseClient*	-
Cu-30	*SupprimerAttribut*	*Pénalité*	-
Cu-31	*AjouterClasse*	-	*Chambre*
Cu-32	*AjouterComposant*	-	*NumChambre*
Cu-33	*AjouterComposant*	-	*Hôtel*
Cu-34	*AjouterComposant*	-	*EtatCh*
Cu-35	*AjouterComposant*	-	*PrixCh*
Cu-36	*RetyperAssociation*	*appartient*	*appartient*
Cu-37	*RemplacerClasse*	*DisponibilitéHotel*	*Disponibilité*
Cu-38	*ChangerCible*	*Disponibilité*	*Chambre*

Tableau 57 : Ecarts sur la carte de haut niveau Cu

Carte affinant Cu.ab1			
Code	**Operateur**	**As-Is élément**	**To-Be élément**
Cu.ab1-1	*RemplacerIntention*	*Créer produit*	*Créer des packages*
Cu.ab1-2	*AjouterStratégie*	-	*Par mise à jour d'un package existant*
Cu.ab1-3	*RemplacerStratégie*	*Cuab1.bc1 En offrant des Disponibilités*	*En offrant des Disponibilités de ressources*
Cu.ab1-4	*AjouterStratégie*	-	*En associant un produit hôtelier et des activités*
Cu.ab1-5	*DiviserStratégie*	*Cu_{ab1}.ab1 En collectant l'information sur l'hôtel*	*Cu_{ab1}.ab1 En collectant l'information sur l'hôtel* *Cu_{ab1}.ab5 En signant des partenariats*
Cu.ab1-6	*JoindrePrécondition*	Cu_{ab1}.ab2	*OR Activité*
Cu.ab1-7	*JoindrePrécondition*	Cu_{ab1}.ab2	*OR Package*
Cu.ab1-8	*JoindrePostCondition*	Cu_{ab1}.ab2	*OR Activité.état="ouvert"*
Cu.ab1-9	*JoindrePostCondition*	Cu_{ab1}.ab2	*OR Package.état="ouvert"*
Cu.ab1-10	*JoindrePostCondition*	Cu_{ab1}.ab3	*Activité.état="ouvert"*
Cu.ab1-11	*JoindrePrécondition*	Cu_{ab1}.bc1	*OR Package.état="ouvert"*
Cu.ab1-12	*JoindrePostCondition*	Cu_{ab1}.bc1	*Chambre*
Cu.ab1-13	*JoindrePostCondition*	Cu_{ab1}.bc1	*OR DisponibilitéChambre*
Cu.ab1-14	*JoindrePostCondition*	Cu_{ab1}.bc1	*OR DisponibilitéActivité*
Cu.ab1-15	*JoindrePrécondition*	Cu_{ab1}.bb1	*Activité.état="ouvert"*
Cu.ab1-16	*JoindrePrécondition*	Cu_{ab1}.bb1	*OR Hotel.état="ouvert"*
Cu.ab1-17	*JoindrePostCondition*	Cu_{ab1}.bb1	*Package.état="ouvert"*

Tableau 58 : Ecarts sur la carte affinant Cu.ab1

Carte affinant Cu.ab2			
Code	**Operateur**	**As-Is élément**	**To-Be élément**
Cu.ab2-1	*RemplacerIntention*	*Gérer les ressources hôtelières*	*Gérer les ressources*
Cu.ab2-2	*RemplacerStratégie*	*Cuab2.bc3 En offrant des Disponibilités*	*En offrant des Disponibilités de package*
Cu.ab2-3	*JoindrePrécondition*	Cu_{ab2}.bc1	OR Package.état="fermé"
Cu.ab2-4	*JoindrePrécondition*	Cu_{ab2}.bc2	OR Package.état="fermé"
Cu.ab2-5	*JoindrePrécondition*	Cu_{ab2}.ab2	Activité
Cu.ab2-6	*JoindrePostcondition*	Cu_{ab2}.ab2	Activité.état="fermée"

Tableau 59 : Ecarts sur la carte affinant Cu.ab2

Carte affinant Cu.bc1			
Code	**Operateur**	**As-Is élément**	**To-Be élément**
Cu.bc1-1	*RemplacerIntention*	*Créer réservation*	*Réserver un Package*
Cu.bc1-2	*SupprimerIntention*	*Gérer des demandes en attente*	-
Cu.bc1-3	*AjouterStratégie*	*ab1*	*Par réutilisation*
Cu.bc1-4	*RetyperOR*	*En capturant l'information par une nouvelle demande et par réutilisation*	*En capturant l'information par une nouvelle demande et par réutilisation*
Cu.bc1-5	*SupprimerStratégie*	*de1 Par annulation du client*	-
Cu.bc1-6	*SupprimerStratégie*	*bc1 Par mise en attente*	-
Cu.bc1-7	*RenommerStratégie*	*bd1 Par annulation de la demande*	*Par abandon*
Cu.bc1-8	*SupprimerStratégie*	*dc1 Par satisfaction de la demande en attente*	-
Cu.bc1-9	*RemplacerStratégie*	*bc1 En vérifiant les disponibilités*	*Par sélection dans une liste d'offre*
Cu.bc1-10	*SupprimerStratégie*	*de2 Par dépassement de délai*	-
Cu.bc1-11	*AjouterStratégie*	-	*C_{bc1}.bb1 Par contreproposition*
Cu.bc1-12	*AjouterStratégie*	-	*C_{bc1}.ac1 Par sélection guidée*
Cu.bc1-13	*AjouterStratégie*	-	*C_{bc1}.bb2 Par affinement*
Cu.bc1-14	*AjouterStratégie*	-	*C_{bc1}.bb3 Par formulation d'une nouvelle demande*
Cu.bc1-15	*SupprimerStratégie*	*ac1 A la demande d'un agent*	-
Cu.bc1-16	*AjouterStratégie*	-	*cc1 Par modification de la réservation*
Cu.bc1-17	*ModifierPrécondition*	Cu_{bc1}.bc1	DisponibilitéChambre.DateDeb <= Demande.DateDeb
Cu.bc1-18	*ModifierPrécondition*	Cu_{bc1}.bc1	AND DisponibilitéChambre.DateFin >= Demande.DateFin
Cu.bc1-19	*ModifierPostcondition*	Cu_{bc1}.bc1	new.DisponibilitéChambre, new.DisponibilitéActivité

Tableau 60 : Ecarts sur la carte affinant Cu.bc1

Carte affinant Cu.ad1			
Code	**Operateur**	**As-Is élément**	**To-Be élément**
Cu.ac1-1	RemplacerIntention	Supprimer un produit	Supprimer un Package
Cu.ac1-2	AjouterStratégie	-	Cac1.ab3 Par la fin d'un partenariat
Cu.ac1-3	ModifierPostcondition	Cu_{ac1}.ab1	new.DisponibilitéChambre, new.DisponibilitéActivité
Cu.ac1-4	JoindrePostcondition	Cu_{ac1}.ab1	Package.état="supprimé"
Cu.ac1-5	JoindrePostcondition	Cu_{ac1}.ab1	Activité.état="fermée"

Tableau 61 : Ecarts sur la carte affinant Cu.ad1

3. Modèles To-Be

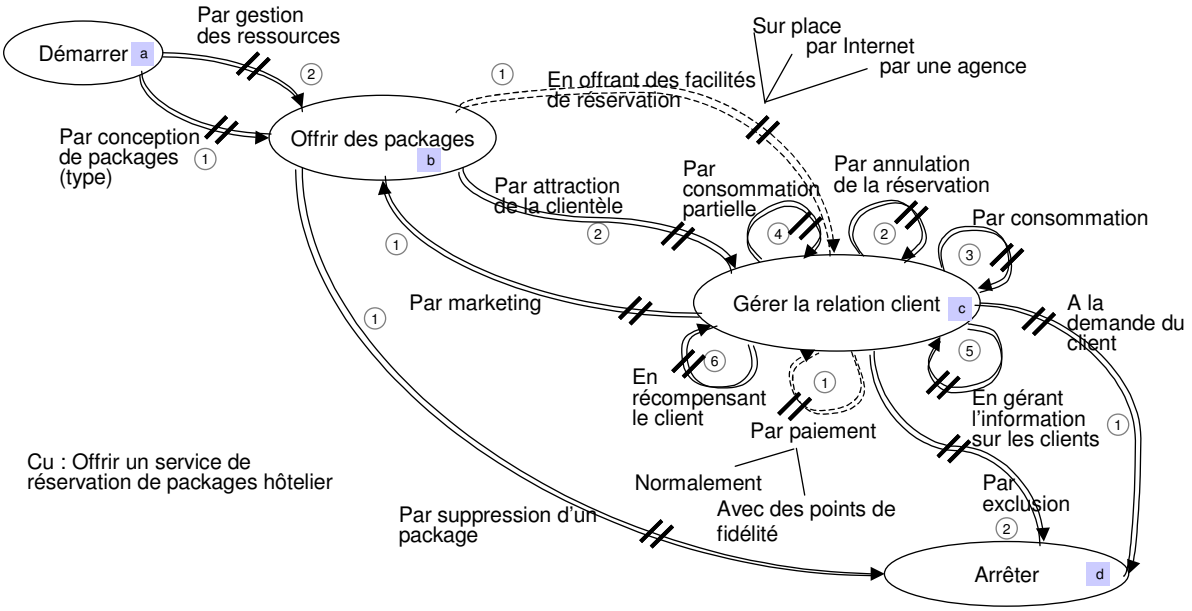


Figure 189 : Carte pivot To-Be du plus haut niveau

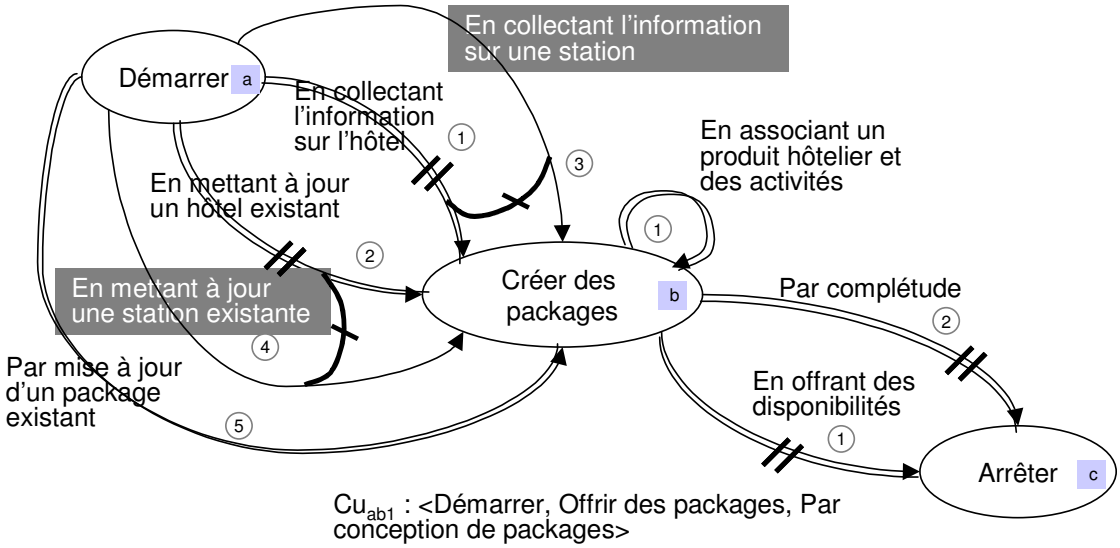


Figure 190 : Carte pivot To-Be affinant la section ab1

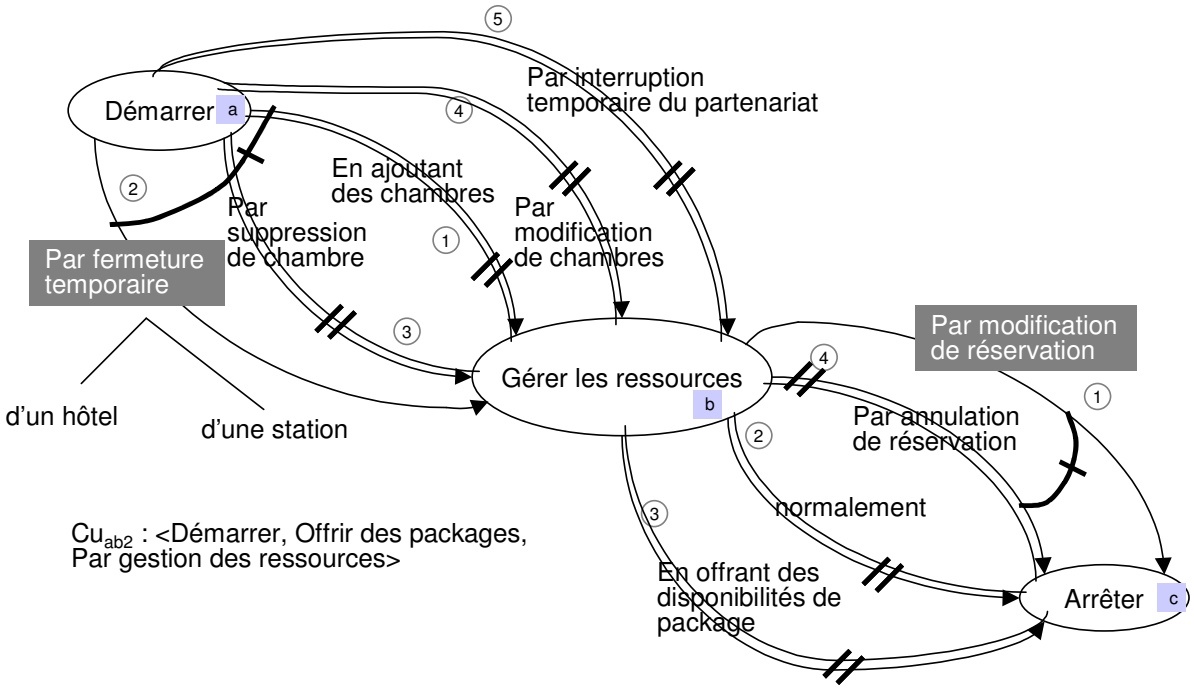


Figure 191 : Carte pivot To-Be affinant la section ab2

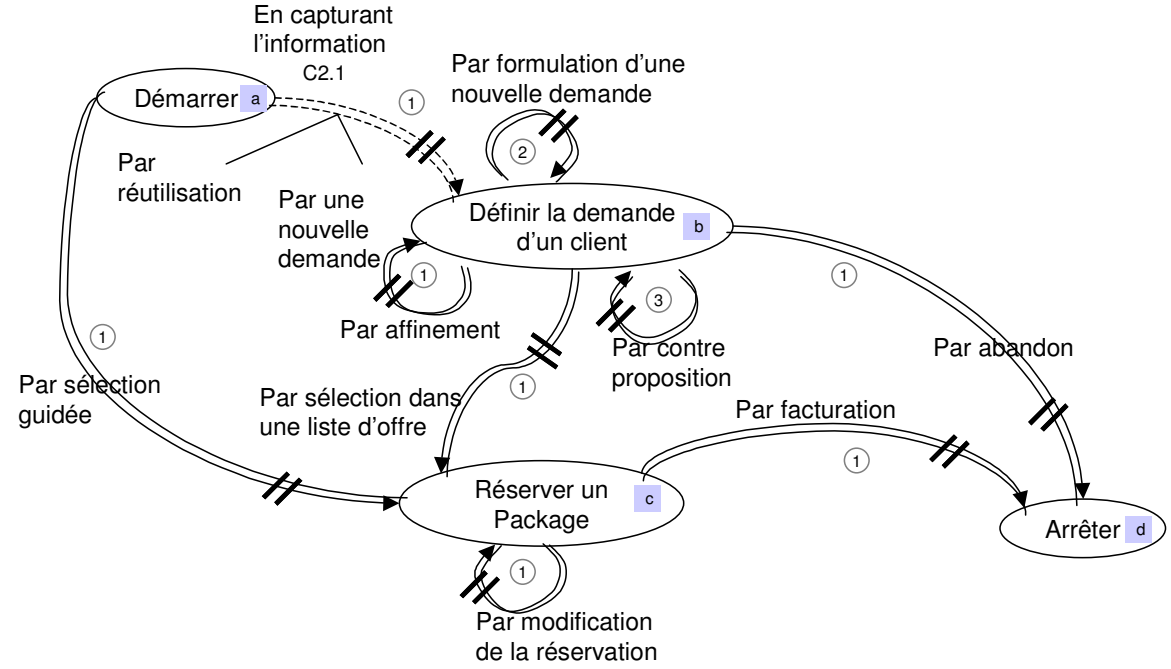


Figure 192 : Carte pivot To-Be affinant la section bc1

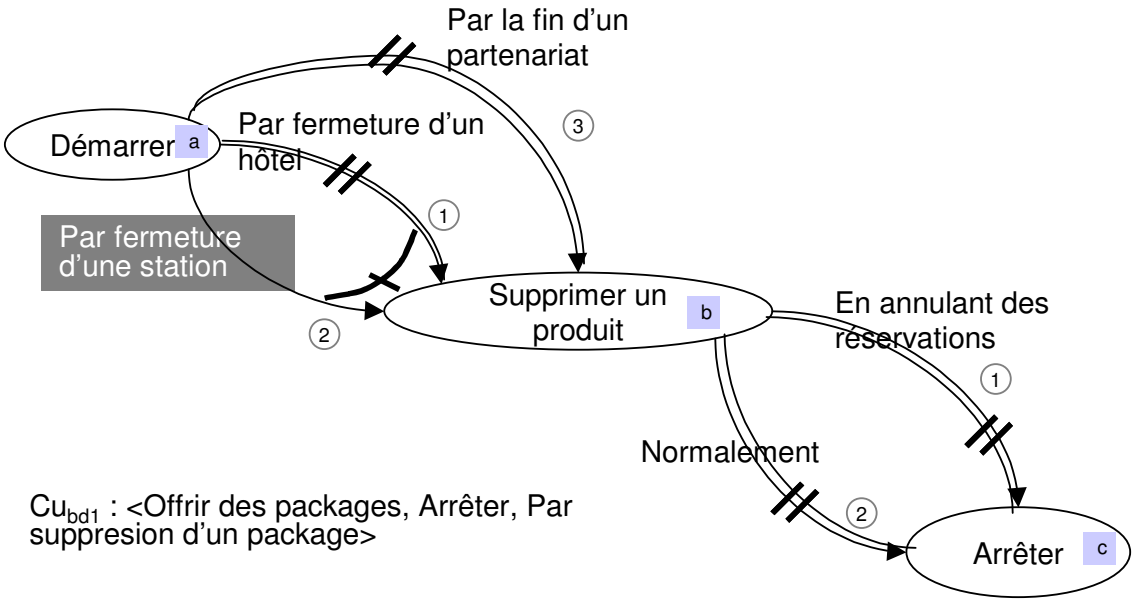


Figure 193 : Carte pivot To-Be affinant la section bd1

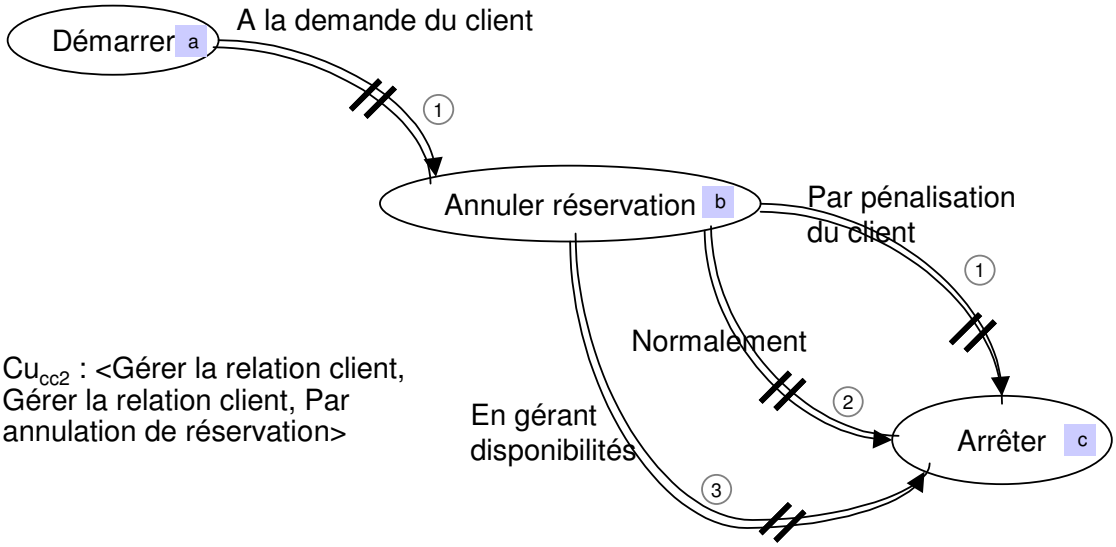


Figure 194 : Carte pivot To-Be affinant la section cc2