MIME : Méthode d'Ingénierie de Méthodes par Evolution

Mohamed Ben Ayed

THESE DE DOCTORAT DE L'UNIVERSITE PARIS I – SORBONNE

Spécialité : **INFORMATIQUE**

Mohamed Ben Ayed

Pour l'obtention du titre de :

DOCTEUR DE L'UNIVERSITE PARIS I - SORBONNE

MIME : Méthode d'Ingénierie de Méthodes par Evolution

JURY

Mme Colette ROLLAND	Directeur de thèse
M. Michel LEONARD	Rapporteur
M. Naveen PRAKASH	Rapporteur
Mme Selmin NURCAN	

Table des Matières

CHAPITRE 1 :

INTRODUCTION

1. Domaine de la Thèse

Cette thèse s'inscrit dans le domaine de l'ingénierie des méthodes de développement des systèmes d'information et plus particulièrement dans une optique d'évolution de ces méthodes.

Cette introduction est consacrée à la présentation du domaine de la thèse et à la définition des concepts-clés utilisés dans ce travail.

- *Système d'Information*

Un *Système d'Information (SI)* est la représentation d'un système du monde réel. Parmi les différentes définitions d'un système d'information figurant dans la littérature, nous retenons la suivante :

"Un système d'information est un artefact qui supporte un réseau de flux d'informations nécessaires pour organiser, mettre en œuvre, gérer et maintenir les activités d'une organisation. C'est un instrument de communication qui sert aux échanges informationnels entre les partenaires de l'organisation et qui accroît leur efficacité." [Rolland 88]

Cette définition souligne l'importance du système d'information pour une organisation. En effet, c'est ce qui contribue à son bon fonctionnement. Avoir des systèmes d'information fiables, flexibles, adaptés et qui permettent de traiter les problèmes informationnels d'une manière efficace est devenu une des préoccupations dominantes des organisations.

La complexité du développement des systèmes d'information nécessite l'utilisation de méthodes d'analyse, de conception et de réalisation.

- *Méthode*

"*Le mot 'méthode', d'origine grecque, signifie chemin: celui tracé à l'avance, qui conduit à un résultat. Une méthode répond d'abord à une question pratique: comment faire, quoi entreprendre, afin d'atteindre un but donné".* (Encyclopédie Universalis).

A.F. Harmsen nous suggère la définition suivante :

« une collection de procédures, de techniques, de descriptions de produits et d'outils pour le support effectif, efficace et consistant du processus d'ingénierie d'un SI ».

Plusieurs autres définitions ont été proposées dans [Kronlof 93], [Smolander 91], [WyneKoop 93], [Lyytinen 89], [Brinkkemper 90], [Seligmann 89], [Harmsen 96], [Prakash97], [Brinkkemper 96]. La plupart d'entre elles convergent vers l'idée qu'une méthode apporte les concepts pour décrire le produit et les règles de conduite méthodologique pour produire un produit de qualité avec une efficacité raisonnable. Cette acception est synthétisée dans la définition de G. Booch [Booch 91] :

« ...un processus rigoureux permettant de générer un ensemble de modèles qui décrivent divers aspects d'un logiciel en cours de construction en utilisant des notations bien définies».

En d'autres termes, une méthode adresse des deux aspects: le produit et le processus et apporte deux éléments : un ou plusieurs *modèles de produit* et un ou plusieurs *modèles de processus*.

Une *Méta-Méthode* est une méthode pour construire des méthodes d'ingénierie de SI.

- *Modèle de produit*

Le *produit* est le résultat de l'application d'une méthode. Comme dit W. Olle [Olle 92], le produit est la cible désirée d'un développement de SI. Un schéma conceptuel de base de données est un exemple de produit, la base de données de l'application 'gestion des emprunts de la bibliothèque nationale' est un autre exemple. Un produit est exprimé dans les termes d'un *modèle de produit*.

Le *modèle de produit* prescrit ce que sont les caractéristiques attendues des produits fabriqués. Il est le moule de production des produits. Une méthode peut comporter plusieurs modèles de produits permettant de modéliser différentes facettes d'un SI, à différents niveaux de détail et à différents niveaux d'abstraction.

Un *méta-modèle de produit* est un ensemble de concepts capable de décrire un ensemble de modèles de produit de méthodes. Il permet une représentation homogène de modèles de produit appartenant soit à la même méthode soit à d'autres méthodes différentes.

- *Modèle de Processus*

Le processus est *« la route à suivre »* pour atteindre la cible que constituent les produits [Olle 92]. Il s'exprime le plus souvent comme « *un ensemble d'activités inter-reliées et menées dans le but de définir un produit »* [Franckson 91]. Il est exprimé dans les termes d'un *modèle de processus*. Le texte décrivant la séquence des activités qui ont conduit au schéma E/R de l'application de 'gestion des emprunts de la bibliothèque nationale' est un exemple de processus.

Un *modèle de processus* prescrit une manière de faire, une démarche méthodologique pour atteindre la cible souhaitée. Il décrit à un niveau abstrait et idéalisé la façon d'organiser la production du produit: les étapes, les activités qu'elles comprennent, leur ordonnancement, et parfois, les critères pour passer d'une étape à une autre. Il joue le rôle de moule des processus d'ingénierie.

D'une part, le modèle de processus n'a de sens que s'il est explicitement mis en relation avec le (ou les) modèle (s) de produits associé(s) mais d'autre part, la qualité du produit dépend fortement de celle du processus mis en œuvre pour l'obtenir.

Un *méta-modèle de processus* est un formalisme de description des modèles de processus. Il apporte un ensemble de concepts permettant la représentation homogène des modèles de processus.

- *Ingénierie des Méthodes*

L'*Ingénierie des méthodes* se préoccupe de la définition de nouvelles méthodes d'ingénierie des systèmes d'information. S. Brinkkemper la définit comme *"une discipline de conceptualisation, de construction et d'adaptation de méthodes, de techniques et d'outils pour le développement des systèmes d'information"* [Brinkkemper 96].

Plusieurs autres définitions restreignent la notion d'ingénierie des méthodes à la construction de nouvelles méthodes à partir de celles déjà existantes. A titre d'exemple, H.T. Punter définit l'ingénierie des méthodes comme: *"une approche de construction des méthodes combinant différentes parties de méthodes pour développer une solution optimale au regard du problème donné"* [Punter 96].

K. Kumar [Kumar 92] propose au contraire une définition plus générale qui n'impose pas l'utilisation de méthodes existantes comme point de départ de l'ingénierie des méthodes. Il définit cette dernière plutôt comme :*"une proposition pour la conception et le développement d'une méta-méthodologie destinée à la conception des méthodes de développement des systèmes d'information"*.

Une variante de cette discipline à été développée, il s'agit de l'*Ingénierie des Méthodes Situationnelles* (*IMS*) qui est définie par R.J. Welke comme : "la discipline visant à construire et à

adapter une méthode de développement de SI et les outils associés à chacun des projets spécifiques auxquels elle est appliquée" [Welke 92a].

2. Problématique

La diversification des domaines dans lesquels on a recours à des SI et la croissance de leur complexité conduisent à un forte demande concernant les méthodes. Il est en effet impensable de nos jours de développer un SI sans le recours à une méthode adéquate. Cependant, l'analyse de la pratique des méthodes [Siau 98], [Stamper 03], [EMMSAD] met en évidence des failles et des limites des méthodes actuelles qu'il est nécessaire de combler. De nombreuses enquêtes [Wijers 90], [Aaen 92], [Yourdon 92], [Russo 95] ont montré que les méthodes ne sont pas bien adaptées aux besoins de leurs utilisateurs, les concepteurs de SI. Les observations que nous avons réalisées auprès d'entreprises en Europe ont révélé que la plupart d'entre elles fabriquent leur propre méthode d'ingénierie de SI convenant à leur structure et à leur expérience du domaine, à partir de fragments d'une ou plusieurs méthodes existantes. L'application de ces méthodes « sur mesure » dépend par la suite de la nature du projet en cours. En effet, selon la taille et les contraintes du projet les ingénieurs d'application peuvent choisir d'appliquer l'intégralité ou des parties de la méthode pour le développement du SI demandé.

Pour répondre à ce besoin d'ingénierie de méthodes, plusieurs approches d'Ingénierie de Méthodes Situationnelles ont été proposées dans la littérature [Philon 98], [Punter 96], [Song 97], [Brinkkemper 98], [Brinkkemper 99], [Ralyté 01a], [Ralyté 01b], [Saeki 03a]. L'objectif de ces approches est de répondre au besoin d'adapter les méthodes aux exigences de leurs utilisateurs et aux spécificités des projets auxquels elles sont appliquées. Ces approches proposent diverses techniques pour assister l'ingénieur de méthodes dans la construction de méthodes spécifiques à la situation de chaque projet à partir de fragments/composants de méthodes existantes.

Cependant un autre problème, non résolu par les approches d'IMS existantes, fait surface dans la l'application des méthodes dans les projets de développement de SI il s'agit du problème d'évolution des méthodes. En effet, plusieurs facteurs peuvent rendre les méthodes ou les fragments de méthodes utilisés inadéquats, rendant leur évolution nécessaire. Parmi ces facteurs on peut citer :

- la constatation de problèmes lors de l'application de la méthode dans les différents projets de développement de SI dans lesquels elle a été utilisée,
- l'apparition de nouveaux besoins d'ingénierie suite aux évolutions continuelles de l'environnement du SI ou à celles des objectifs des organisations,
- les évolutions technologiques et l'apparition de nouveaux paradigmes de programmation qui doivent se répercuter sur les méthodes d'ingénierie de SI. On constate par exemple, que des méthodes ont évolué pour suivre l'apparition des paradigmes *Orienté Objet*, *Orienté*

Composant et dernièrement *Orienté Service* (*Merise* est un exemple de méthode ayant subie plusieurs évolutions : *Merise* en 1979, *Merise 2* en 1990 et *OOM* en 1992),

- un besoin exprimé par les éditeurs pour améliorer leur méthode afin d'augmenter sa productivité, ou d'élargir son champ d'application pour couvrir d'autres domaines dans un souci de diversité et de polyvalence,

Comme c'est le cas de la plupart des projets d'ingénierie de méthodes, l'évolution de méthodes est réalisée actuellement de manière ad-hoc sans réelle réflexion sur le processus d'évolution. Cette façon de faire cause des erreurs et génère des problèmes de cohérence au sein de la méthode. Les répercussions de tels problèmes dans un projet de développement de SI peuvent être catastrophiques en terme de coût pour l'entreprise.

Peu d'approches d'IM ont traité le problème de l'évolution de méthodes. On peut citer les travaux de J.-P. Tolvanen [Tolvanen 98] qui propose une approche incrémentale d'évolution de méthodes par affinement de modèles; Terrasse et al. [Terrasse 03] proposent une approche basée sur la méta-modélisation pour conduire l'évolution de modèles; Enfin, M. Boyd et P. McBrien ont défini dans [Boyd 04] une approche semi-automatique, basée sur un ensemble d'opérateurs, pour la transformation de modèles utilisant des formalismes différents. La limite de ces approches consiste à restreindre l'activité d'évolution à une partie de méthode ou à un seul type de modèle. Elle est également liée à l'absence d'un modèle de processus permettant de guider l'ingénieur de méthodes dans les différentes phases du projet d'évolution.

Le problème de l'évolution des méthodes reste largement non résolu et nous l'adaptons comme problématique de ce travail de recherche. Cette problématique peut s'énoncer comme suit : **Répondre au besoin d'une approche pour assister l'ingénieur de méthodes dans l'évolution d'une méthode existante vers une nouvelle méthode satisfaisant de nouveaux besoins d'ingénierie.**

L'évolution d'une méthode est une tâche difficile à mener dans laquelle il faut tenir compte de plusieurs paramètres. En effet, une méthode est un ensemble complexe composé d'un ou plusieurs modèles de produit et d'un ou plusieurs modèles de processus en relation les uns avec les autres. Toucher à l'intégrité de cet ensemble en modifiant un de ces éléments impose d'étudier la répercussion d'un changement local sur les autres éléments composant de façon à garantir la cohérence du tout.

Plusieurs autres difficultés accompagnent l'activité d'évolution d'une méthode et doivent être prises en compte dans la proposition d'une approche d'évolution. Ce sont :

- La diversité des méthodes : chaque méthode à ses propres spécificités liées notamment à la nature des modèles utilisés à son domaine d'application, à ses objectifs et à l'expérience du

domaine de ces concepteurs. → *Proposer une approche d'évolution de méthodes nécessite de définir des mécanismes permettant de tenir compte de cette diversité.*

- Une insuffisance de description de la méthode à faire évoluer : l'évolution d'une méthode consiste à faire évoluer ses modèles; une difficulté fréquemment rencontrée est l'absence même de modèles précis ou la non conformité de ceux-ci à la méthode effectivement appliquée par les utilisateurs → *La proposition d'une approche d'évolution passe par la rétro-ingénierie de la méthode de départ.*

- La nature complexe des méthodes : Comme nous l'avons énoncé ci-dessus une méthode est composée d'un ou plusieurs modèles de produit et d'un ou plusieurs modèles de processus contenant des concepts à différents niveaux de détail et traitant différentes facettes du SI. Cette complexité rend difficile la tâche d'évolution → *L'approche proposée doit utiliser un formalisme permettant d'offrir un guidage fin pour l'évolution d'une méthode tout en tenant compte de sa complexité*.

- La satisfaction des objectifs d'évolution : comme tout projet d'ingénierie, le projet d'évolution d'une méthode doit satisfaire les besoins et objectifs d'évolution. → *La proposition d'une approche d'évolution requiert de définir le moyen de capturer les besoins d'évolution et de garantir que la solution finale satisfait ces besoins* .

- Garantir la cohérence et l'exactitude de la méthode obtenue par évolution → *L'approche à proposer doit offrir des mécanismes pour la vérification et la correction de la méthode après l'évolution.*

Les propositions de cette thèse visent à satisfaire ces contraintes.

3. Objectifs de la Thèse :

Pour résoudre la problématique présentée ci-dessus, nous nous sommes fixé dans ce travail, les objectifs suivants :

1. Proposer un *cadre général* pour l'ingénierie de l'évolution de méthodes,
2. Proposer un *méta-modèle générique* de méthodes capable de décrire la plupart des méthodes d'ingénierie de SI. Ce méta-modèle permet de venir à bout du problème de la diversité des méthodes évoqué ci-dessus. Notre approche d'évolution sera entièrement basée sur ce méta-modèle.
3. Définir une *démarche de rétro-ingénierie* de méthodes permettant de formaliser les modèles de produit et de processus d'une méthode informelle ou mal définie.

4. Définir une *démarche pour l'ingénierie de méthodes par évolution.* L'objectif de cette démarche est d'assister l'ingénieur de méthodes dans toutes les étapes du projet d'évolution d'une méthode.
5. *Positionner l'approche proposée* dans le contexte général de l'ingénierie des méthodes.
6. *Valider l'approche proposée* sur un cas d'étude. Ce cas est celui de la méthode industrielle Lyee [Negoro 01a], [Negoro 01b].

4. Éléments de la solution proposée

Pour atteindre ces objectifs nous proposons une nouvelle *Méthode d'Ingénierie de Méthodes par Evolution* appelée *MIME*. Cette méthode vise à guider l'ingénieur de méthodes dans les différentes étapes de l'évolution d'une méthode existante vers une nouvelle méthode satisfaisant un ensemble d'objectifs d'évolution.

La Figure 1 présente une vue d'ensemble de la méthode *MIME*. Comme toute méthode, MIME est constituée d'un *modèle de produit* et d'un *modèle de processus*. Le produit de l'application de la méthode MIME est une méthode de développement de SI adaptée aux exigences d'évolution ; le modèle de produit de MIME consiste alors en un *méta-modèle de méthode*. Par ailleurs, la méthode MIME comporte deux modèles de processus : un *modèle de processus de rétro-ingénierie* et un *modèle de processus d'évolution.* Ces deux modèles s'exécutent de manière séquentielle.

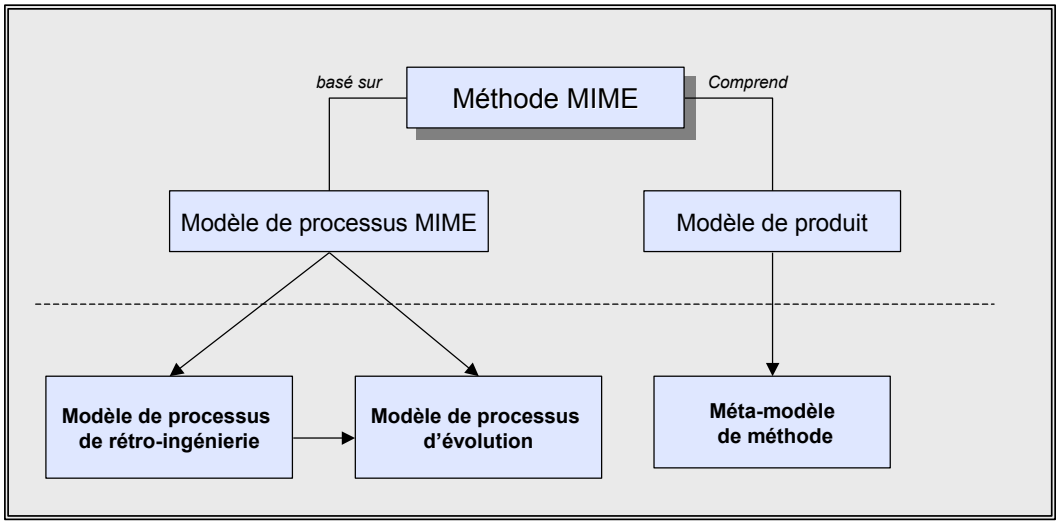


Figure 1. Vue d'ensemble de la méthode MIME

Dans ce qui suit, on présente successivement les composants de la méthode MIME :

- Le *méta-modèle de méthodes* que nous proposons est le méta-modèle de référence dans ce travail. Ce méta-modèle recense l'ensemble des concepts génériques qui représentent les caractéristiques communes à toute méthode d'ingénierie de SI. Le problème de la diversité de méthodes est ainsi

résolu par abstraction. La définition de la méthode *MIME* est entièrement basée sur ce méta-modèle. Toute méthode à faire évoluer par *MIME* est une instance de ce méta-modèle; de même toute méthode obtenue par l'application de *MIME* est instance de ce méta-modèle. Enfin, *MIME* est elle-même une instance de ce méta-modèle.

- *Le modèle de processus de rétro-ingénierie de méthodes* : permet de formaliser les modèles mal ou non-définis de la méthode à faire évoluer. La rétro-ingénierie est une phase optionnelle et préliminaire à l'application du modèle de processus d'évolution.

- *Le modèle de processus d'évolution de méthodes* : offre un guidage à l'ingénieur de méthodes Il l'assiste dans les différentes étapes d'évolution d'une méthode. Ce modèle propose plusieurs démarches alternatives selon les spécificités de chaque méthode et les cas d'évolution. Il offre à l'ingénieur de méthodes un ensemble de *directives* permettant de le guider dans le choix de la démarche adéquate à la situation de son projet d'évolution et dans l'application de la démarche sélectionnée en lui décrivant les actions à exécuter dans chacune des situations rencontrées durant le projet d'évolution.

La Figure 1 illustre l'application de la méthode *MIME* pour l'évolution d'une méthode. Le processus guidée par *MIME* consiste en deux étapes : (i) Une *étape de rétro-ingénierie* dans laquelle l'ingénieur de méthodes est guidé par le modèle de processus de rétro-ingénierie pour formaliser une méthode. Il part d'une description initiale informelle de la méthode et aboutit à l'ensemble des modèles de produit et de processus qui la formalisent. (ii) Une *étape d'évolution* au cours de laquelle l'ingénieur de méthodes est guidée par le modèle de processus d'évolution pour faire évoluer une méthode existante (*la méthode As-Is*) vers une nouvelle méthode (*la méthode To-Be*) satisfaisant les objectifs d'évolution. Il est à noter que l'étape de rétro-ingénierie est optionnelle dans la mesure où une méthode correctement formalisée peut être évoluée directement sans formalisation préliminaire.

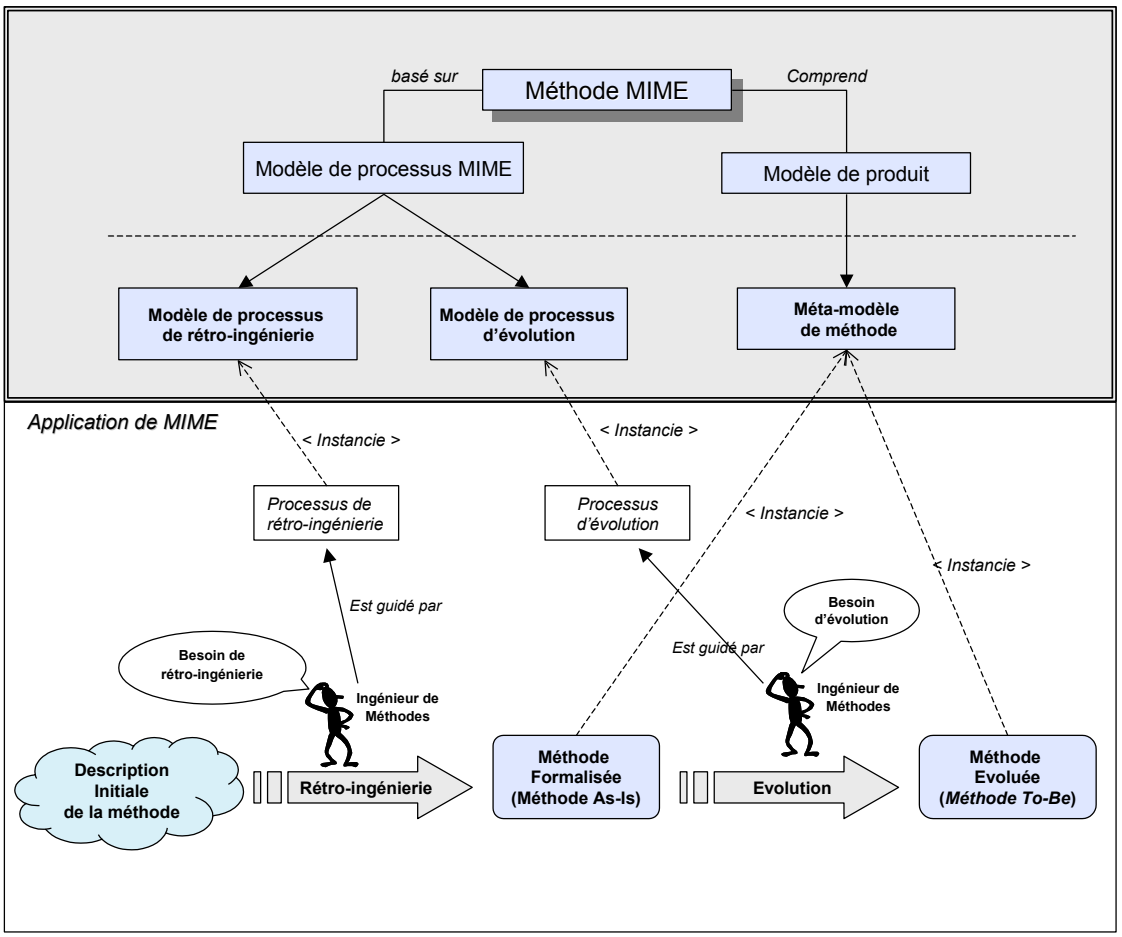


Figure 2. Application de la méthode MIME

Pour positionner la méthode *MIME* dans le contexte général de l'ingénierie de méthodes, nous proposons un *modèle de processus générique d'ingénierie de méthodes* défini comme point de départ de tout projet d'ingénierie de méthodes. Ce modèle vise à aider l'ingénieur de méthodes dans la définition de son objectif d'ingénierie et dans la sélection de l'approche de construction de méthodes la plus appropriée au cas spécifique de son projet. Notre approche d'évolution est une alternative parmi d'autres dans ce modèle.

Pour la construction des modèles de processus présentés ci-dessus, nous utilisons un méta-modèle de processus appelé le modèle de la *carte* [Rolland 99], [Benjamen 99]. Une *carte* est un modèle de processus *multi-démarches* qui permet de représenter un ensemble très riche de processus puisqu'il propose plusieurs démarches alternatives en fonction des objectifs à atteindre et de la situation d'ingénierie. Le modèle de la Carte a par ailleurs l'avantage de garantir l'*extensibilité* des modèles de processus proposés.

La méthode *MIME* pour l'ingénierie de méthodes par évolution proposée dans ce travail de recherche présente un avantage par rapport aux autres tentatives d'approches, dans la mesure où elle caractérise les différentes manières d'effectuer l'évolution d'une méthode (*par adaptation de modèles*, *par*

évolution de paradigmes, *par instanciation de méta-modèles* et enfin, *par abstractions de modèles*) et propose une démarche permettant de conduire chaque type d'évolution tout en tenant compte du contexte dans lequel elle est appliquée.

5. Contexte de la Thèse

Le travail proposé dans cette thèse et effectué au *Centre de Recherche en Informatique*, a été réalisé en grande partie, dans le cadre du projet « *Lyee International Scientific Research Project*»[1] [LyeeProject]. Il s'agit d'un projet de recherche international impliquant trente trois chercheurs et différents partenaires industriels de douze pays. L'objet de ce projet est d'effectuer des recherches autour de la méthode *Lyee*. *Lyee « GovernementaL MethodologY for softwarE providencE »* est une méthode de développement de logiciels inventée par F. Negoro [Negoro 01a], [Negoro 01b]. Cette méthode réduit significativement le cycle de développement de logiciels en éliminant la phase de programmation et en permettant la génération automatique de code à partir des spécifications déclaratives des besoins logiciels.

Les objectifs de ce projet sont multiples : il vise en premier lieu l'étude, l'évaluation de la méthode Lyee et sa comparaison avec les méthodes d'ingénierie de SI existantes. En deuxième lieu, il vise à apporter des améliorations à cette méthode pour pallier ses limites, élargir son champ d'application et la promouvoir dans le milieu industriel.

Notre participation dans ce projet a un double objectif : Le premier concerne la *ré-ingénierie* de la méthode *Lyee* par la formalisation de ses modèles de produit et de processus. Le deuxième concerne *l'évolution* de cette méthode pour proposer une nouvelle méthode (baptisée "*Lyee User Driven"*) supportant le développement de logiciels en deux étapes : l'ingénierie des besoins et la génération automatique de code. La première étape est notre contribution ; la deuxième étape est assurée par l'environnement *CASE* appelé *LyeeALL*.

Après réflexion sur le travail réalisé dans le cadre de ce projet pour l'évolution de la méthode Lyee, nous concluons que le problème que nous avons résolu est récurrent dans plusieurs situations où une méthode a besoin d'être évoluée pour satisfaire de nouveaux besoins émergeants. Un effort d'abstraction a alors été réalisé pour placer ce problème dans le contexte général de l'ingénierie de méthodes et essayer de le résoudre.

6. Plan de la Thèse

Le plan de ce mémoire est le suivant :

[1] Ce projet est sponsorisé par *Catena Corporation – Japan* et *the Institute of Computer Based Software Methodology and Technology*.

- Le chapitre 2 est consacré à un état de l'art en ingénierie des méthodes ; il définit les principes clés de cette discipline et présente les différentes approches de construction de méthodes proposées dans la littérature.
- Le chapitre 3 est consacré à la présentation du méta-modèle de méthodes proposé dans ce travail. Ce chapitre introduit : (1) la notion de méthode, (2) et celle de méta-modèle de méthode puis détaille, (3) le méta-modèle de produit et (4) le méta-modèle de processus. Il présente en détail la notion de *carte de processus méthodologique* et celles de *directive* et *patron* attachés à la carte.
- Le chapitre 4 présente le modèle de processus de *rétro-ingénierie de méthode*. Ce modèle de processus est formalisé par une *carte* et un ensemble de *directives*.
- Le chapitre 5 présente notre *Démarche d'Ingénierie de Méthodes par Evolution (DIME)*. Il définit le modèle de processus de cette démarche également décrit par une *carte* et un ensemble de *directives*.
- Le chapitre 6 est consacré à l'application de notre démarche pour l'évolution de la méthode Lyee. Ce chapitre présente étape par étape le processus d'évolution de cette dernière.
- Le chapitre 7 est consacré à la conclusion.

CHAPITRE 2 :
ETAT DE L'ART

1. Introduction

Ce chapitre est consacré à la présentation d'un état de l'art sur les différents travaux réalisés en ingénierie de méthodes. Il présente un aperçu des approches existantes en les classant selon la technique de construction de méthodes qu'elles proposent.

Le plan de ce chapitre est organisé comme suit : la première partie est consacrée à la présentation des principes fondamentaux ou principes-clés en ingénierie de méthodes, la deuxième partie est consacrée à la présentation des différentes approches d'ingénierie de méthodes qui utilisent ces principes.

2. Principes de l'ingénierie de méthodes

L'ingénierie de méthodes met en œuvre quatre principes-clés que l'on présente dans cette section :
- le principe de *méta-modélisation,*
- le principe de *réutilisation,*
- le principe de *modularité,*
- et enfin, le principe de *flexibilité.*

2.1. Principe de méta-modélisation

La méta-modélisation consiste à identifier les caractéristiques communes et génériques des différents modèles et à les représenter ensuite par un système de concepts génériques. Une telle représentation, appelée méta-modèle, permet de générer tous les modèles partageant ces mêmes propriétés.

Le principe de méta-modélisation a été largement utilisé pour la proposition de plusieurs approches en ingénierie de méthodes [Brinkkemper 99], [Kelly 96], [Harmsen 95a], [SiSaid 96], [Karlsson 02],

[Odell 96], [Smollander 91], [Saeki 94], [Saeki 02], [Hofstede 93], [Grundy 96], [Rolland 95], [Rolland 99], [Jarke 99], [Prakash 99], [Tolvanen 03], [Terrase 03], [MetaModel.com].

La méta-modélisation est non seulement utile dans la construction des méthodes mais aussi dans la formalisation des méthodes mal définies [Tolvanen 93] [Rolland 02a], dans la comparaison des méthodes [Hong 93], [Rossi 96], dans l'adaptation de méthodes [Terrasse 03], dans la standardisation des méthodes [Booch 98] et dans la définition des liens entre les méthodes d'ingénierie et les langages de programmation

2.2. Principe de réutilisation

La *réutilisation* en ingénierie des méthodes est inspirée de la réutilisation dans le monde du logiciel où elle se définit comme une approche de développement selon laquelle il est possible de construire un système à partir de composants existants, produits à l'occasion de développements antérieurs. Le principe de réutilisation logicielle est appliqué aujourd'hui à toutes les étapes du cycle de développement d'un logiciel. Initialement introduite pour améliorer la productivité de la programmation, la réutilisation intervient dans les activités d'analyse et de conception ainsi qu'en ingénierie des besoins.

L'ingénierie de méthodes applique le principe de réutilisation pour construire de nouvelles méthodes d'ingénierie des systèmes d'information en assemblant différents fragments de méthodes qui ont déjà fait leurs preuves [Harmsen 94], [Harmsen 95a], [Brinkkemper 98], [Brinkkemper 99], [Saeki 03a], [Punter 96], [Song 97], [Prakash 99], [Prakash 02], [Ralyté 01b], [Ralyté 01a]. Les fragments de méthode sont des descriptions réutilisables des parties des modèles de produit et des modèles de processus qui constituent une méthode, c'est-à-dire des fragments de leurs méta-modèles. Ces descriptions constituent les blocs de construction réutilisables qui permettent de définir des méthodes de manière modulaire. Les méthodes ainsi construites sont elles-mêmes modulaires et peuvent être modifiées et étendues facilement. La présentation de la technique de construction de méthodes par assemblage de composants fera l'objet de la section 3.3 de ce chapitre.

2.3. Principe de modularité

La mise en œuvre de la réutilisation en ingénierie de méthodes nécessite que l'on découpe une méthode en blocs réutilisables et donc d'introduire la *modularité* comme principe dans la description des méthodes. Selon ce principe une méthode est vue comme une collection de composants réutilisables. Par analogie avec la notion de module en génie logiciel, un composant de méthode doit avoir un certain nombre de qualités telles que la *cohésion, l'autonomie* et *l'inter opérabilité*. Il est *cohésif* à condition que son contenu constitue un tout cohérent et *autonome* s'il peut être utilisé seul pour résoudre un problème d'ingénierie de système d'information. Il doit être *inter opérable* afin de

permettre son assemblage avec d'autres composants dans le processus de construction d'une nouvelle méthode.

Il n'y a pas de standard de définition d'un composant de méthode mais selon Brinkkemper [Brinkkemper 98] trois dimensions sont à prendre en compte : la *perspective*, l'*abstraction* et la *granularité.* Un modèle peut se décliner à différents niveaux de *granularité* : modèle de produit tout entier (le modèle E/R) ou bien heuristique méthodologique fine (enlever la cardinalité zéro dans un schéma E/R) et à différents niveaux d'*abstraction* : niveau d'un modèle spécifique (modèle E/R), d'un modèle générique (le modèle de processus 'en fontaine') ou celui d'un méta-modèle (MOF). Enfin dans la mesure où une méthode est composée d'un modèle de produit et d'un modèle de processus, on conçoit que deux *perspectives* [Ralyté 04], [Saeki 03b] puissent être considérées : celle d'un composant de méthode entièrement de produit ou de processus [Brinkkemper 98], [Van Slooten 96], [Song 97], [Saeki 03a], [Firesmith 02], [Debenhan 03], [Henderson 98], [Henderson 00], [Henderson 04] et celle qui voit un composant comme un mixte des deux [Punter 96], [Rolland 96a], [Ralyté 01a], [Prakash 99], [Wistrand 04].

2.4. Principe de flexibilité

Harmsen, [Harmsen 94] propose un spectre des méthodes d'ingénierie (Figure 3) qui organise les approches d'ingénierie des méthodes selon le degré de flexibilité de la méthode au regard de la situation rencontrée. Cette figure illustre le fait que l'on peut regrouper les approches d'ingénierie de méthodes dans trois grands types différents : *les approches rigides*, *les approches semi-rigides* et *les approches situationnelles*.

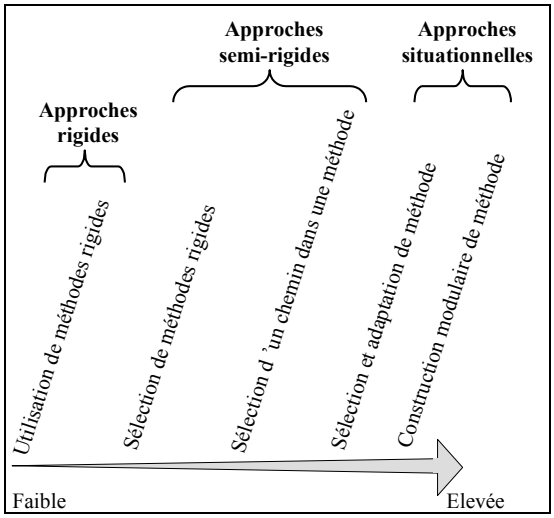


Figure 3. Spectre des approches d'ingénierie des méthodes

- *Les approches rigides* : ces approches utilisent des méthodes complètement pré-définies laissant peu de possibilités pour les adapter à la situation rencontrée. Elles adoptent certaines standardisations basées sur un type de projet et ne fournissent pas d'aide concrète pour

adapter ces méthodes à d'autres types de projets. Des méthodes comme SSADM [Longworth 92] ou Merise [Tardieu 83] appartiennent à cette catégorie.

- *Les approches semi-rigides* : Entre les approches *rigides* et les méthodes *situationnelles* existe un certain nombre d'approches que l'on peut appeler *semi-rigides* [Benjamen 99]. Ce sont des approches qui permettent la sélection dans un panel de méthodes ou chemins rigides (sélection basée sur la situation du projet). Ensuite, la méthode sera utilisée telle quelle sans adaptation. Il est en général peu vraisemblable que la méthode sélectionnée réponde à toutes les exigences du projet. Cette typologie comprend également les approches proposant plusieurs chemins à suivre selon le type du projet en cours. Dans ce type d'approche se trouvent par exemple *Toolkit* [Nenyon 87] et *Multiview* [Wood-Hgarper 85] qui se résument en l'inclusion de plusieurs méthodes, chacune concernant un aspect spécifique du système. Selon la situation, l'une de ces sous-méthodes sera employée dans le projet. Cependant, cette approche ne fournit pas de suggestions concrètes pour les facteurs qui déterminent la sélection de ces outils. De plus, la compatibilité des sous-méthodes incorporées est problématique [Zaal 92].
- *Les approches situationnelles* : ces approches utilisent et/ou modifient des méthodes pour les adapter à une situation donnée en tenant compte de ses spécificités. Les approches par sélection et adaptation d'une méthode permettent, pour chaque nouveau projet, de sélectionner des méthodes parmi plusieurs et de les accorder aux besoins du projet (modification, extension, suppression, etc.). Les approches par la construction de méthodes modulaires permettent de créer des méthodes facilement décomposables en modules, ce qui facilite leur réutilisation pour la construction d'autres méthodes.

L*e principe de flexibilité* est au cœur de l'IM dont la problématique centrale est celle de l'adaptation des méthodes : adaptation selon l'acception de Harmsen, c'est-à-dire aux contingences d'un projet, adaptation aux besoins spécifiques d'un groupe d'usagers, adaptation dynamique dans le contexte du processus d'IM lui-même. Le principe de flexibilité influence le produit de l'ingénierie, c'est-à-dire la nouvelle méthode mais aussi le processus d'IM. Le spectre d'Harmsen s'attache au premier aspect et conclue que les méthodes modulaires sont les plus flexibles ; le second aspect est implicitement dans le spectre de la Figure 3 et conduit à la conclusion que la construction d'une méthode '*à la volée'* est la plus flexible.

3. Un état de l'art des approches de construction

Un grand nombre d'approches d'ingénierie de méthodes a été proposé dans la littérature, cette section est consacrée à la présentation d'un aperçu de ces approches classées par rapport à la démarche de construction de méthodes qu'elles proposent.

3.1. Les approches Adhoc

La technique de construction de méthodes *ad-hoc* est une technique traditionnelle basée sur l'expérience acquise lors des développements de différents systèmes d'un domaine spécifique. Tant que cette expérience n'est pas formalisée et ne constitue pas une connaissance de base disponible pour les différents ingénieurs d'application, on peut dire que cette connaissance est le résultat d'une technique de construction ad-hoc. Ceci a deux conséquences majeures : la méconnaissance de la manière dont la méthode a été générée et sa dépendance au domaine d'expertise. Si la méthode doit être indépendante du domaine d'expertise, facile à appliquer et à modifier, il est alors nécessaire de sortir du cadre des techniques de construction basées sur l'expérience. Les techniques comme l'instanciation et l'assemblage favorisent la flexibilité et la modularisation des méthodes et facilitent la capitalisation des bonnes pratiques et l'amélioration des modèles existants.

3.2. Construction de méthodes par instanciation de méta-modèle

Une approche de construction des méthodes très répandue en ingénierie de méthodes est celle de l'utilisation de la technique d'instanciation de méta-modèle.

En appliquant cette approche, le modèle de processus de la méthode est construit par instanciation du méta-modèle de processus, de la même manière le modèle de produit est construit par instanciation du méta-modèle de produit. La Figure 4 résume les niveaux d'abstraction définis pour cette approche.

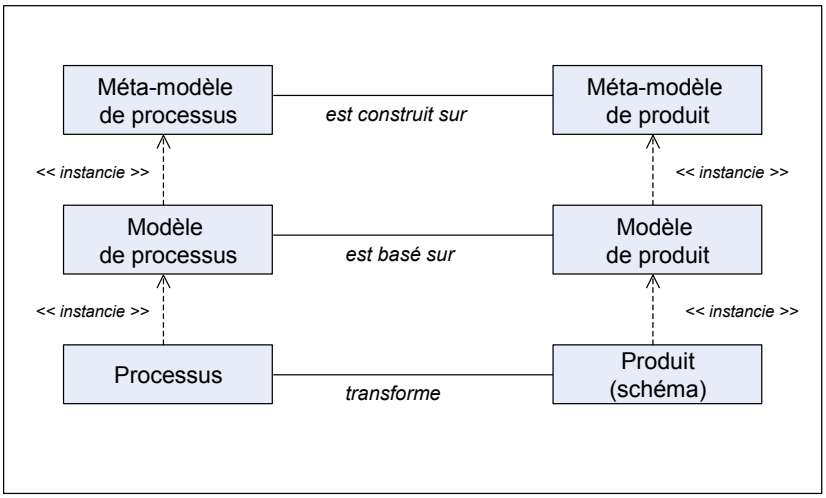


Figure 4. Les niveaux d'abstraction

Au niveau le plus haut de cette figure, se trouvent le *méta-modèle de processus* et le *méta-modèle de produit*. Ces deux méta-modèles contiennent des concepts génériques qui permettent respectivement, de décrire le *modèle de produit* et le *modèle de processus* d'une méthode spécifique.

Tout élément d'un modèle de produit est instance d'un concept du *méta-modèle de produit*. Le *méta-modèle de produit* définit les concepts nécessaires pour caractériser une situation du produit. De même, tout élément du *modèle de processus* est instance d'un concept du *méta-modèle de processus*. Les concepts du modèle de processus font référence aux concepts du modèle de produit.

Au niveau d'une application réelle, l'ingénieur d'application suit un *processus*. L'exécution de ce processus est contrôlée par une instance du *modèle de processus* qui prescrit la démarche à suivre. Le *processus* exécuté transforme un *produit* dont les éléments sont des *instances* du *modèle de produit*.

Cette approche de construction de méthodes propose deux étapes fondamentales : (1) l'identification d'un méta-modèle de méthodes adéquat et (2) l'instanciation du méta-modèle sélectionné. Plusieurs méta-modèles de méthodes ont été proposées dans la littérature. La plupart d'entre eux prennent en compte les deux perspectives de modélisation : le produit et le processus [Hofstede 93], [Grundy 96], [Saeki 93], [Saieki 94], [Plihon 96], [Prakash 99], [Prakash 02], [Ralyté 01b], [Ralyté 04].

3.3. Construction de méthodes par assemblage

Les approches de construction de méthodes par assemblage font partie des approches situationnelles dans le spectre des approches d'ingénierie de méthodes présenté à la Figure 3. La plupart des travaux réalisés dans cette discipline durant cette décennie sont inscrits dans cette direction.

Les approches de construction de méthodes par assemblage se basent sur les principes de *réutilisation* et de *modularité* présentés à la section 2. La ligne directrice de ces approches est la construction de méthodes adaptées aux caractéristiques et spécificités de chaque projet par assemblage de *composants de méthode* (aussi appelés *fragments*, *modules* ou *blocs*).

Parmi les principaux travaux réalisés dans cette direction on peut citer :

- Harmsen, Brinkkemper et Saeki proposent dans [Harmsen 94], [Harmsen 95], [Brinkkemper 98] une approche d'assemblage à base des fragments de méthodes de différents niveaux de granularité. Afin de la distinguer des autres approches, nous attribuons le nom *Fragment de méthodes* à cette approche.
- Rolland et Prakash [Rolland 96] proposent une structure pour représenter et stocker des composants de méthodes de différents niveaux d'abstraction et de différents niveaux de granularité. Nous appelons cette approche par *Composant contextuel*.

- Prakash propose un méta-modèle pour représenter des méthodes sous forme modulaire [Prakash 97], [Prakash 99]. Les modules sont appelés des blocs de méthodes d'où le nom de l'approche : *Bloc de méthodes*.
- Song propose une approche d'intégration des méthodes de conception qui est basée sur l'adaptation des méthodes existantes [Song 97]. Elle permet de compléter une méthode par une fonctionnalité complémentaire prise dans une autre méthode et/ou d'améliorer sa qualité en lui associant des propriétés issues d'autres méthodes.
- Punter et Lemmen proposent une approche appelée MEMA [Punter 96] comportant un méta-modèle pour représenter les fragments de méthodes ainsi qu'un processus de sélection et d'assemblage de fragments afin de construire une méthode pour un projet concret.
- L'approche proposée par Van Slooten se base surtout sur la caractérisation des projets en utilisant des facteurs de contingence [Van Slooten 93], [Van Slooten 96].
- J. Ralyté [Ralyté 01b] propose une approche d'ingénierie de méthodes à base de composants, proposant un méta-modèle de méthodes modulaires, un modèle de processus de ré-ingénierie de méthodes sous forme de composants et un modèle de processus d'assemblage de ces composants. Ce dernier utilise un ensemble d'*opérateurs d'assemblage*, de *mesures de similarité* et de *règles de validation*.
- M. Saeki [Saeki 02] propose une approche d'assemblage de composants se basant sur la transformation de modèles. Il fournit pour cela des règles pour l'assemblage de composants issus de méthodes basées sur les diagrammes et de méthodes formelles.

Trois étapes phares interviennent dans les approches de construction de méthodes par assemblage. Ces étapes sont présentées à la Figure 5 qui illustre notre vision du cycle de l'ingénierie de méthodes situationnelles.

- la ré-ingénierie de méthodes existantes sous forme de composants réutilisables indépendamment,
- le stockage de ces composants dans une base de méthodes et
- le guidage de l'ingénieur de méthodes dans la sélection et l'assemblage de composants.

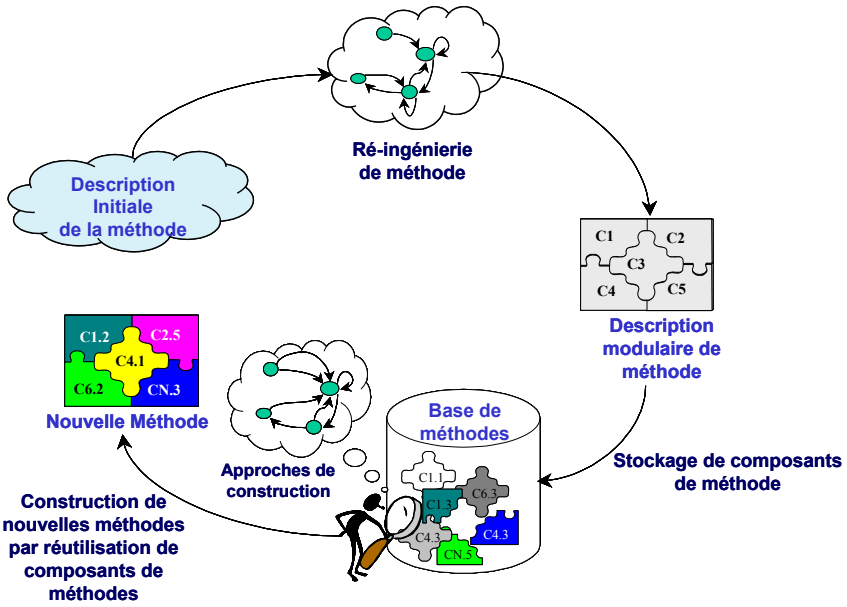


Figure 5. Le cycle de l'Ingénierie des Méthodes Situationnelles

Les travaux cités ci-dessus couvrent partiellement ou totalement ces étapes, dans ce qui suit nous allons présenter chacune de ces étapes et les différentes solutions proposées pour les supporter.

3.3.1. Ré-ingénierie de méthodes

Comme l'illustre la Figure 5, la ré-ingénierie de méthodes est la première tâche à réaliser dans le cycle de l'ingénierie des méthodes situationnelles, elle vise la redéfinition d'une méthode sous la forme de composants réutilisables développant ainsi le principe de *modularité* présenté à la section 2.

Dans la plupart des cas, la définition des composants est basée sur la technique de méta-modélisation. Ceci permet de résoudre le problème de représentation des méthodes sous forme de composants. La plupart des approches existantes ne prennent en compte que la structure des composants et ne se préoccuppent pas du processus de reconstruction des méthodes sous forme de composants.

J. Ralyté [Ralyté 01b] propose un modèle de processus multi-démarches pour assister l'ingénieur de méthodes dans la ré-ingénierie d'une méthode. Ce modèle se base sur un méta-modèle de méthodes modulaire et présente l'avantage de tenir compte du contexte et des spécificités de chaque méthode.

S'il y a peu de travaux sur le processus de ré ingénierie, de nombreux efforts ont été fournis pour définir des langages de spécification des composants réutilisables de méthodes. Brinkkemper [Brinkkemper 90] émet l'hypothèse que tout langage de modélisation conceptuelle est adéquat et de nombreux exemples tels que LOTOS [Saeki 91] semblent valider cette hypothèse. Cependant, d'autres auteurs ont une opinion contraire tels Venable et Grundy qui ont développé un langage spécifique COCOA [Venable 93] et qui a été utilisé par la suite dans l'environnement MVIEWS [Grundy 96]. L'étude des exigences faite par Marttiin dans [Marttiin 95] semble confirmer cette

dernière position et la justifier par le besoin de constructions sémantiques spécifiques de la modélisation des méthodes que n'offrent pas les langages standards tels que UML.

Harmsen et Saeki [Harmsen, 1996] distinguent quatre écoles. La première opte pour une approche orientée données qui met l'accent sur la représentation de l'aspect produit des méthodes. Elle inclut des langages tels que GOPRR [Smolander 92], [Kelly 96], PSM-LISA/D [Hofstede 93], les diagrammes de structure objet de NIAM [Brikkemper 90], [Wijers 91], les modèles sémantiques de données [Sowa 92] et ADSM [Heym 92]. Une deuxième école adopte l'approche orientée objet. Les langages appartenants à cette école sont Telos [Mylopoulos 90], Mataview [Sorenson 88] et ObjectZ [Saeki 94]. La troisième école met l'accent sur l'aspect processus des méthodes et propose des langages tels que les diagrammes de Structure de Tâches [Wijers 91], [Verhoef 95], HFSP [Katamaya 89], [Song 92], ALF [Benali 89] et Merlin [Emmerich 91]. La dernière école propose des langages hybrides, réellement définis pour l'IM comme MEL proposé par Harmsen et al. [Harmsen 95a], [Harmsen 95b] et Brinkkemper dans [Brinkkemper 95], [Brinkkemper 01].

Harmsen et Saeki [Harmsen 96] ont choisi quatre langages (un de chaque catégorie) : Object-Z, MEL, GOPRR et HFSP et les ont comparés. La conclusion de l'étude est qu'il n'y a pas de langage universel pour l'ingénierie de méthodes. Les différents langages ont des propriétés communes et des propriétés complémentaires. Le choix du langage dépend de l'objectif à atteindre dans l'IM. Comme solution, Harmsen et Saeki proposent d'appliquer la technique d'assemblage sur les langages d'ingénierie de méthodes eux-mêmes et de construire ainsi un langage adapté à la situation donnée à partir de divers fragments de langages existants.

3.3.2. Stockage des composants de méthodes

Au cœur du cycle présenté à la Figure 5 se trouve la *base des méthodes* qui stocke les composants réutilisables de méthodes. Elle apporte la *connaissance* qui est encapsulée dans les composants eux-mêmes. Un composant peut par exemple, apporter des heuristiques d'écriture d'un scénario d'interaction (partie processus) et comporter la description conceptuelle d'un tel type de scénario (partie produit). La connaissance qu'il fournit est réutilisable dans tout projet où la capture des besoins se fait par écriture de scénarios.

La plupart des approches d'assemblage utilisent la notion de base de méthodes pour stocker les composants. On trouve une base de méthodes dans les approches *Fragments de méthode*, *Composants contextuels*, MEMA, celle de van Slooten ainsi que celle de J. Ralyté. Toutefois, la plupart des approches ne précisent pas quelle est la structure de cette base ni comment les composants peuvent y être atteints.

[Song 97], au contraire, ne parle pas d'une base de méthodes et ne dit pas comment décrire ni comment stocker les composants. L'approche *Blocs de méthodes* propose un formalisme pour décrire les blocs de méthodes mais ne mentionne pas non plus comment les stocker.

L'approche *Composants contextuels* ainsi que celle de [Ralyté 01b] divisent la base de méthodes en deux niveaux : le premier niveau est appelé *connaissance méthode* et le deuxième *méta-connaissance*. Le niveau *connaissance méthode* comporte la connaissance qui est effectivement réutilisable, c'est-à-dire les composants eux-mêmes, tandis que le niveau de *méta-connaissance* comporte l'information nécessaire à la sélection et la réutilisation des composants. Dans le domaine de la réutilisation ces deux niveaux sont appelés la *connaissance réutilisable* et la *connaissance pour la réutilisation*.

La méta-connaissance peut prendre des formes variées : *descripteur* [Rolland 96b], *méta-classe* [Saeki 93], *liens hypertexte* [Brinkkemper 00], [Ralyté 99] et vise à satisfaire deux objectifs : (a) comprendre la nature de la connaissance apportée par le composant et (b) caractériser les situations de sa réutilisation. Slooten [Van Slooten 96] par exemple, associe les fragments de méthodes (les composants) à des valeurs d'un ensemble prédéfini de facteurs de contingence caractéristiques des situations de projets.

Pour représenter la méta-connaissance l'approche *Composants contextuels* et celle de J. Ralyté utilisent la notion de *descripteur* qui permet de personnaliser les composants. Ceci facilite l'accès aux composants ainsi que leur sélection dans la base de méthodes. Chaque composant de méthode a un descripteur qui décrit son contexte de réutilisation. Un descripteur relie la situation dans laquelle le composant est pertinent à l'intention qu'il permet de satisfaire dans le processus d'ingénierie de systèmes. La situation dans l'approche *Composants contextuels* détermine des caractéristiques qu'un projet doit avoir pour que le composant soit applicable dans son développement. La situation dans l'approche de J.Ralyté fait référence aux domaines dans lesquels le composant est applicable et les activités de conception dans lesquelles le composant peut participer. L'intention dans les deux approches représente une intention d'ingénierie de systèmes qui peut être réalisée en appliquant le composant.

3.3.3. Guidage dans la sélection et l'assemblage de composants

La dernière étape dans le cycle de l'ingénierie des méthodes situationnelles (Figure 3) est la construction de nouvelle méthode par sélection et assemblage de composants stockés dans la base de méthodes. Cette étape développe le principe de *réutilisation* présenté ci-dessus. On en trouve différentes variantes de techniques d'assemblage de composants dans [Harmsen 94], [Harmsen 95a], [Brinkkemper 98], [Brinkkemper 99], [Saeki 03a], [Punter 96], [Song 97], [Ralyté 01b], [Ralyté 01a].

Dans les approches *Fragments de méthodes,* MEMA ou celle de Van Slooten la première étape dans la construction de méthodes situationnelles concerne la caractérisation du projet en cours. Ceci est basé sur la définition des facteurs de contingence qui peuvent être identifiés à l'aide des interviews, des questionnaires et d'autres techniques d'acquisition de la connaissance. MEMA, par exemple, utilise un cadre de référence défini pour ce propos. [Van Slooten 96] définit une liste de facteurs de contingence comme l'importance du projet, son impact sur l'organisation existante, la pression du temps, l'expérience de l'équipe de développement, la taille, les relations avec d'autres systèmes, la dépendance des autres projets, la clarté, la stabilité, la complexité, l'innovation etc.

Selon les auteurs de ces approches, les facteurs de contingence sont utilisés ensuite pour sélectionner les composants appropriés. L'approche MEMA par exemple, a un processus qui permet de déterminer la correspondance entre la caractérisation du projet à l'aide de leur cadre de référence et les descriptions des composants stockés dans une base de méthodes. Cependant, les autres auteurs restent assez flous dans leurs explications concernant l'évaluation de l'ensemble de facteurs de contingence et leur impact sur le choix d'un composant de méthode plutôt qu'un autre.

Le processus d'assemblage des composants sélectionnés reste également assez flou dans les approches MEMA et celle de Van Slooten. Par contre, l'approche *Fragments de méthodes* propose un processus assez détaillé guidant l'assemblage des fragments de méthodes sélectionnés au préalable. Ce processus consiste à assembler les fragments de produits et les fragments de processus correspondants. L'assemblage de deux fragments de produit consiste à identifier un ou plusieurs liens entre différents concepts des deux fragments. Des nouveaux concepts peuvent être créés pour permettre la connexion entre les fragments de produit. En ce qui concerne l'assemblage des fragments de processus, des liens de précédence entre les différentes activités des composants doivent être établis. Des contraintes d'assemblage sous forme des règles formalisées sont proposées pour assurer la cohérence et la complétude de l'assemblage obtenu. Cependant, cette approche se limite à l'assemblage des fragments complémentaires qui n'ont pas d'éléments communs.

Ralyté et Rolland [Ralyté 01a], [Ralyté 01b] proposent un processus d'assemblage qui prend en compte deux cas d'assemblages : *Assemblage par association et Assemblage par intégration*. Dans *l'assemblage par association,* les composants issus de méthodes différentes (M1 et M2) sont disjoints. Ils sont en général, complémentaires et l'association consiste à établir des liens entre M1 et M2. Dans *l'assemblage par intégration,* les composants se recouvrent et la construction de la nouvelle méthode requiert un travail d'assemblage plus complexe qui consiste à intégrer les concepts de M1 à ceux de M2 au moyen d'opérateurs appropriés. L'intégration de tels composants permet d'obtenir un nouveau composant encore plus riche que les deux initiaux. La Figure 6 illustre ces deux cas d'assemblage

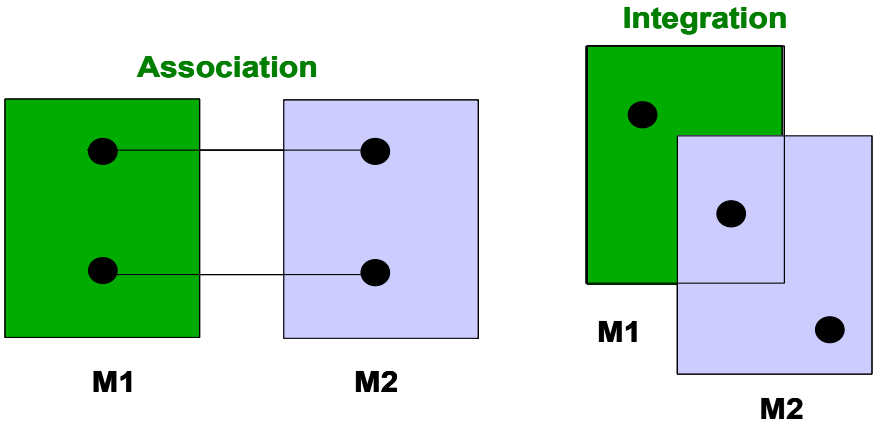


Figure 6. Les Cas d'assemblage

Song dans son approche parle d'adaptation et d'amélioration des méthodes existantes plutôt que de la construction d'une nouvelle méthode. Il propose deux types d'assemblage qu'il appelle *basé fonction* et *basé qualité*. Le premier type permet d'ajouter des nouvelles fonctionnalités dans une méthode tandis que le deuxième permet d'améliorer la qualité de la méthode, d'augmenter son efficacité en lui ajoutant de nouvelles propriétés. Les deux types d'assemblage concernent les deux niveaux de composants. Les processus d'assemblage restent cependant informels, proposant uniquement des raisons, des heuristiques, des exemples mais pas de guidage systématique.

3.4. Construction de méthode par extension

Les approches *'Par Extension'* proposent différents types d'extension qui peuvent être réalisés sur une méthode existante. Leur objectif est d'enrichir une méthode par de nouveaux concepts et propriétés [Deneckere 01], [Deneckere 02], [Etien 03]. Par exemple, une méthode statique telle que celle pour la construction de schémas E/R peut être étendue pour représenter le temps de manière systématique par l'introduction d'un calendrier de points temporels, d'intervalles etc. ainsi que celle des aspects temporels tels que l'historisation des entités.

R. Deneckere [Deneckere 01] propose une approche d'extension de méthodes existantes en utilisant des patrons génériques. De plus, cette approche comporte une technique d'organisation de ces patrons avec un guidage pour leur réutilisation ainsi qu'une technique de conception de nouveaux patrons d'extension à l'aide de méta-patrons. Cette approche offre :

- Une approche de représentation de la connaissance concernant les extensions de méthodes sous forme de *patrons* décrits par différents langages de description et de manipulation ;
- Une technique d'organisation de ces patrons, pour faciliter le guidage lors de leur réutilisation, par la spécification d'un modèle de processus décisionnel exécutable appelé *carte d'extension* ;
- Une approche de conception des patrons d'extension à l'aide de *méta-patrons* génériques.

Grâce à son pouvoir de réutilisation, le concept de patron a été introduit dernièrement en ingénierie de méthodes. En effet, le travail proposé dans [Rolland 96a]et [Rolland 96b] introduit la notion de patron comme un moyen de modéliser le comportement commun dans la construction des méthodes. [Rolland 96a] définit plusieurs patrons de construction référencés par les verbes d'intentions qu'ils permettent de satisfaire dans le processus de construction d'une méthode, comme par exemple *Identifier*, *Décrire*, *Construire*, *Définir*, *Valider* et *Affiner*. Ces patrons peuvent être appliqués à la construction de plusieurs méthodes. Ils sont génériques dans le sens où ils peuvent être utilisés par un ingénieur de méthodes dans le processus de construction de plusieurs méthodes.

3.5. Construction de méthode par évolution

Les approches de construction de méthodes par évolution se placent dans la classe des approches de *Sélection et adaptation de méthodes* du spectre de Harmsen (Figure 3). Elles offrent donc un degré de flexibilité inférieur à celui de la construction de méthodes par assemblage.

Comme nous l'avons précisé dans l'introduction, le principe de cette technique de construction est d'apporter un ensemble de transformations aux modèles d'une méthode existante afin de proposer une nouvelle méthode satisfaisant les objectifs d'ingénierie.

Rares sont les approches qui se sont intéressées au problème d'évolution de méthodes. Les solutions existantes ne s'attaquent pas au problème dans sa totalité et résolvent chacune une facette de celui-ci : Tolvanen [Tolvanen 98] a proposé une approche incrémentale d'évolution de méthodes par affinement de modèles dont la réalisation est effectuée par adaptation. Boyd [Boyd 4] quant à lui, propose une approche semi-automatique pour la transformation d'un modèle utilisant un formalisme (par exemple E/R) vers un modèle utilisant un autre formalisme (par exemple UML) en moyennant un ensemble d'opérateurs.

L'avantage de l'approche que nous proposons est de s'attaquer au problème d'évolution de méthodes sous tous ses angles et dans les différentes situations de projet. La proposition de cette thèse met l'accent sur le guidage et l'identification de différentes manières de procéder à l'évolution d'une méthode.

3.6. Utilisation d'un support logiciel

L'ingénierie de méthodes a fait émerger une nouvelle génération d'ateliers logiciels désignés par l'acronyme anglais CAME (Computer Aided Method Engineering). Par analogie avec les ateliers de génie logiciel dont le propos est d'apporter une aide à la conduite du développement des systèmes d'information, un *atelier méthode* vise à guider la construction des méthodes. Il doit donc offrir des fonctionnalités d'aide à l'accomplissement des activités clés en ingénierie de méthodes, soit :

- *la ré ingénierie de méthode existante,*

- *le stockage des composants de méthodes,*
- *l'extraction de composants répondant à certains critères,*
- *la construction de méthodes selon différentes stratégies : assemblage, adaptation, évolution, 'from scratch' etc.*
- *la vérification et la validation de la méthode construite,*
- *l'amélioration dynamique de la méthode obtenue.*

Harmsen [Harmsen 94] a été le premier à définir les exigences d'un environnement CAME dont certaines ont été implémentées dans le prototype Decamerone [Harmsen 95b]. Un certain nombre de prototypes ont vu le jour tels que MetaEdit+ [Kelly 96], Meet [Heym 93], Meru [Prakash 99], [Gupta 01] et Mentor [Si-said 96].

L'outil MEET propose un modèle de représentation de méthodes qui peut être utilisé pour standardiser, comparer et intégrer des différentes méthodes. MetaEdit+ inclut un nombre d'instanciations principalement sur les aspects produit d'une vingtaine de méthodes. L'accent, dans l'outil Mentor, a été mis sur l'unification des aspects de produit et de processus des méthodes ainsi que sur le guidage et le support qui peut être fourni pour l'ingénieur de méthodes et l'ingénieur d'applications. MERU consiste en deux parties : la première aide à formuler les exigences sur la méthode à construire, la deuxième génère la méthode. Les exigences sur la méthode sont exprimées en utilisant un méta-modèle spécifique appelé Method View. La génération de la méthode utilise la technique d'assemblage. La notion de composant de méthode est définie pour permettre ceci.

Parmi les outils cités ci-dessus, seul MetaEdit+ est devenu un produit commercial [Metacase.com].

4. Conclusion

Dans ce chapitre nous avons présenté un état de l'art des principaux travaux réalisés en ingénierie des méthodes. Dans cette présentation nous avons présenté en premier lieu, les principes-clés de l'ingénierie de méthodes puis dans un second temps, nous avons présenté les différentes approches de construction de méthodes.

Cet état de l'art a montré que peu de travaux se sont intéressés à l'ingénierie de l'évolution des méthodes. C'est à ce sujet que nous nous intéressons particulièrement.

.

CHAPITRE 3 : LE META-MODELE DE METHODES

1. Introduction

Le propos de ce chapitre est de définir la notion de méthode d'ingénierie de SI. Il présente en premier lieu une vue générale de la notion de méthode et, en deuxième lieu, le méta-modèle de méthode que nous proposons. Ce méta-modèle définit l'ensemble des fondements théoriques de la formalisation de la méthode MIME d'ingénierie de méthodes par évolution présenté dans ce mémoire.

Ce chapitre est organisé comme suit : la première section présente la notion de méthode d'ingénierie de SI ; la deuxième section est consacrée à la présentation du méta-modèle de méthodes développé dans ce travail ; la troisième présente le méta-modèle de produit et, la quatrième est dédiée à la présentation du méta-modèle de processus.

2. Notion de méthode d'ingénierie de SI.

Selon notre point de vue – partagé par plusieurs auteurs ([Kronlof 93], [Smolander 91], [Wynekoop 93], [Prakash 94], [Brinkkemper 90], [Harmsen 94], [Brinkkemper 96]) – une méthode comporte un ou plusieurs *modèles de produit* et un ou plusieurs *modèles de processus*.

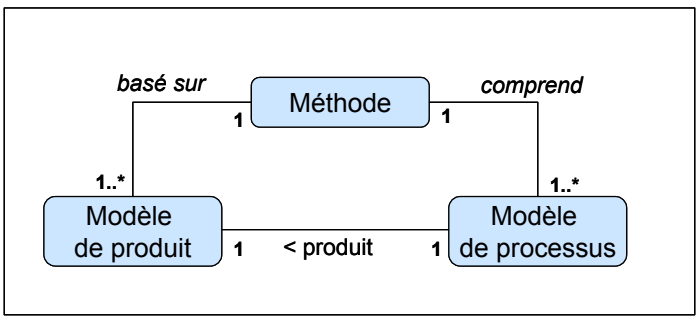


Figure 7. Vue Générale d'une Méthode

La Figure 7 illustre la vue générale d'une méthode, composée de deux parties : une partie produit et une partie processus. La partie produit est représentée par un ou plusieurs modèles de produit et la

partie processus par un ou plusieurs modèles de processus. Chaque modèle de produit est couplé à un modèle de processus. Le modèle de produit représente la structure d'une classe de produits obtenus en appliquant la méthode dans différentes applications, tandis que le modèle de processus représente la démarche à suivre pour obtenir le produit cible.

2.1. Le Modèle de produit

Comme nous l'avons défini dans l'introduction, le *produit* est le résultat d'application d'une méthode. Le *modèle de produit* prescrit ce que sont les caractéristiques attendues des produits fabriqués par la méthode, il est le moule de production de ces produits.

Comme l'illustre la Figure 7, une *méthode* peut comporter plusieurs *modèles de produit* permettant de modéliser différentes facettes d'un SI (*statique*, *dynamique*, *fonctionnel*) [Olle 92], à différents niveaux de détail (*classe*, *paquetage*, *sous-système*) [Muller 00], [Fowler 97], [UML 03], [PUML] et à différents niveaux d'abstraction, (*objet*, *classe*, *méta-classe*) [Shlaer 92], [Graham 94], [Coad 91], [Booch 91], [MOF 02]. Des modèles pour représenter le contexte organisationnel du SI et les exigences à son égard ont introduit de nouveaux concepts tels que *but*, *acteur*, *scénario*, *rôle* [Rolland 98a], [Rolland 98c], [Lamsweerde 00], [Yu 94], [Potts 94]. Parallèlement, les modeleurs de besoins ont introduits la distinction entre les besoins *fonctionnels* et les besoins *non-fonctionnels* [Robertson 99], [Chung 00]. Ces extensions du champ de modélisation dans l'ingénierie d'un SI ont conduit à une nouvelle typologie des modèles permettant de classer les aspects du SI en fonctionnel, non fonctionnel et intentionnel [Rolland 98b].

La Figure 8 donne l'exemple du modèle de produit de la méthode d'analyse de Coad et Yourdon [Coad 91] présenté comme une liste de cinq éléments types : *liste d'objets et de classes, relations entre classes, groupes de classes, attributs et opérations*. Tout produit conforme à ce modèle se compose d'un ensemble d'éléments des cinq types identifiés par le modèle.

Modèle de Processus	Modèle de Produit
1. Identifier les objets	1. Liste des objets et de classes
2. Identifier les structures	2. Relation entre classes
3. Identifier les sujets	3. Groupe de classe
4. Définir les attributs	4. Attributs
5. Définir les services	5. Opération (service)

Figure 8. Exemple des modèles de la méthode Coad et Yourdon

2.2. Le Modèle de processus

Comme nous l'avons défini dans l'introduction le processus est la route à suivre pour atteindre la cible que constitue le produit [Olle 92]. Il décrit à un niveau abstrait et idéalisé la façon d'organiser la

production du produit: les étapes, les activités qu'elles comprennent, leur ordonnancement, et parfois, les critères pour passer d'une étape à une autre. Il joue le rôle de moule des processus d'ingénierie.

La Figure 8 présente le modèle de processus de la méthode Coad et Yourdon qui est le corollaire du modèle de produit introduit dans la même figure. Il se présente sous la forme d'une séquence de cinq activités-types : *Identifier les objets, Identifier les structures, Identifier les sujets, Définir les attributs et Définir les services*. Tout processus conforme à ce modèle est une séquence de cinq activités de chacun des cinq types identifiés par le modèle.

D'une part, le modèle de processus n'a de sens que s'il est explicitement mis en relation avec le (ou les) modèle (s) de produit associé(s) mais d'autre part, la qualité du produit dépend fortement de celle du processus mis en œuvre pour l'obtenir. Les deux aspects sont donc fortement liés.

Dowson [Dowson 88] propose une classification des modèles de processus en trois types : *orienté-activité, orienté-produit* et *orienté-décision*.

Les modèles *orientés-activité* se concentrent sur les activités exécutées pour produire un produit et sur leur ordonnancement. Les méta-modèles de processus comme *Cascade* [Royce 70], *Spirale* [Boehm 90], *Hiérarchique en spirale* [Iivari 90] ou *Fontaine* [Henderson-Sellers 90] et plus récemment le RUP [Jacobson 99] font partie de la catégorie. Ces méta-modèles permettent de modéliser le processus par un ensemble de phases de haut niveau organisées en un processus séquentiel. Ils s'adressent donc à des processus tactiques dont l'objectif est de planifier les étapes à suivre.

Les modèles *orientés-produit* couplent l'état du produit à l'activité qui génère cet état. Ils visualisent le processus comme un diagramme de transitions d'état. Le modèle d'Humphrey [Humphrey 89], celui des ViewPoints [Finkelstein 90] et le modèle de processus proposé par le projet ESF(European Software Factory) [Franckson 91] appartiennent à cette catégorie.

Les modèles *orientés-décision* perçoivent les transformations successives du produit comme résultats de décisions. Ces modèles mettent l'accent sur la décision et le contexte de délibération autour de la décision à apprendre (alternatives, arguments PROs et CONs). Les activités ne sont plus au centre du modèle mais apparaissent comme les conséquences des décisions[Jarke 92], [Potts 89].

Deux nouvelles classes de modèle de processus ont été ajoutées récemment : *orienté-contexte* et *orienté-stratégie*.

Les modèles *orientés-contexte* comme celui de la théorie NATURE [Rolland 95], [Plihon96], [Jarke 99], du projet F3 [Rolland 94] ou de l'environnement PRIME [Pohl 99] sont inspirés du paradigme de planification en Intelligence Artificielle utilisé dans GRAPPLE [Huff 89]. Ils définissent le processus à travers la combinaison de situations observables avec un certain nombre d'intentions (de

décision) spécifiques. Le travail à faire est décrit dans le processus comme étant dépendant à la fois de la situation et de l'intention ; en d'autres termes, il dépend du contexte dans lequel on se trouve au moment de le réaliser.

Le modèle *orienté-stratégie* [Rolland 99], [Benjamen 99] est une extension des modèles précédents qui vise à représenter plusieurs démarches dans le même modèle de processus. Il est donc multi-démarches et prévoie plusieurs chemins possibles pour élaborer le produit. Il est basé sur les deux notions d'*intention d'ingénierie* et de *stratégie* à suivre pour réaliser l'intention. La puissance d'expression de ce modèle et les nombreux avantages qu'il confère nous ont amené à l'utiliser comme modèle central de la définition de la méthode *MIME*.

2.3. Rôles des méthodes dans le développement des SI

Le rôle d'une méthode d'ingénierie des SI est essentiellement d'aider les ingénieurs d'application à comprendre le domaine du projet en cours (la structure de l'entreprise, ses métiers, ses modes de fonctionnement, ses objectives et stratégies d'évolution, etc.) et à spécifier les besoins que le nouveau SI doit satisfaire effectivement. Elle doit également aider à concevoir et à réaliser une solution informatique adéquate en proposant des modèles permettant de représenter différentes vues et perspectives du SI en construction. Une méthode d'ingénierie de SI est aussi un moyen de communication entre les différentes parties prenantes, les ingénieurs, les futurs utilisateurs et les clients du SI ainsi qu'entre les différentes équipes de développement au sein du projet (les concepteurs, les développeurs, etc.).

L'utilisation de méthodes dans les projets de développement de SI permet d'assurer la cohérence de la solution globale, d'explorer plusieurs solutions possibles, de détecter des erreurs ainsi que de discerner les situations et les facteurs à risque qui pourraient mener le projet à l'échec. Finalement, la méthode est un guide précieux qui doit être suivi tout au long du cycle de développement de SI.

Une multitude de méthodes d'ingénierie de SI, a été proposée ces dernières décennies. Chacune de ces méthodes se distingue des autres par la nature de ses modèles, ses objectifs et son domaine et contexte d'application. Nous avons tenté d'intégré dans un même méta-modèle les différentes propositions existante. La méthode MIME présentée dans ce travail repose entièrement sur ce méta-modèle.

3. Méta-Modèle de Méthodes

La Figure 9 présente le *méta-modèle de méthodes* que nous proposons dans ce travail en utilisant la notation UML [UML 03]. Ce méta-modèle est une extension de la vue générale d'une méthode

présentée à la Figure 7, il définit les concepts du modèle de produit et ceux du modèle de processus tout en précisant les relations qui les lient.

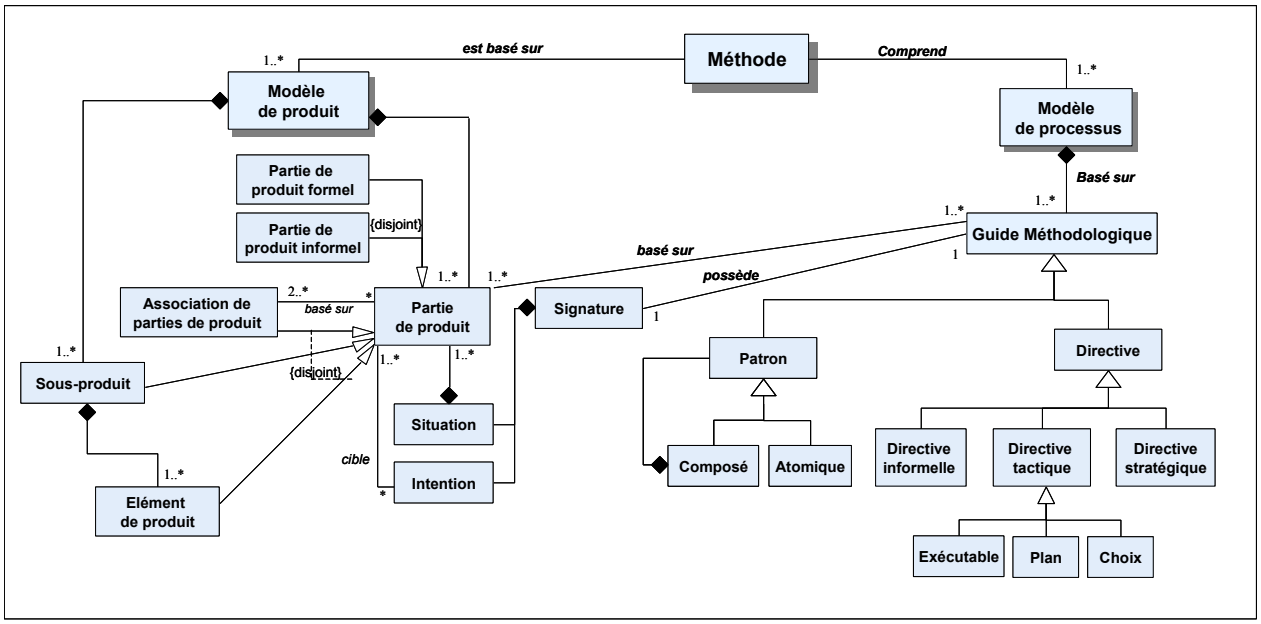


Figure 9. Méta-modèle de méthode

Comme le montre cette figure, le modèle de produit d'une méthode est composé d'une ou plusieurs *parties de produit*. Une partie de produit représente n'importe quel fragment de connaissance se rapportant au produit. Elle désigne soit la totalité du produit en cours de construction (le modèle de produit en entier), soit un *sous-produit* (un composant du modèle de produit), soit un *élément de produit* (un composant du sous-produit), soit une *association de parties de produit*.

Une *partie de produit* peut être *formelle ;* elle est dans ce cas structurée et descriptible en énumérant les différents concepts qui la composent. Elle peut également être *informelle,* c'est-à-dire, non structurée et énoncée en langage naturel.

Une méthode comprend un ou plusieurs modèles de processus. Selon la vision adoptée dans ce travail, tout modèle de processus est basé sur un ou plusieurs *guides méthodologiques*. Un guide méthodologique a pour rôle de guider l'ingénieur d'application dans les différentes activités lui permettant d'atteindre un but et aboutissant à la construction ou à la transformation d'une ou plusieurs parties de produit. Comme le montre la Figure 9, chaque guide méthodologique possède une *signature* qui représente le contexte de son application. Une signature est un couple <*situation*, *intention*>. Une *situation* est basée sur la ou les parties de produit qui sont nécessaires pour satisfaire l'intention du guide méthodologique. Une *intention* exprime un but que l'ingénieur d'application souhaite atteindre en appliquant ce guide, la *cible* de ce but est composée d'une ou plusieurs parties de produit. L'exécution d'un guide méthodologique permet alors d'assister l'ingénieur dans la

construction de la ou des parties du produit-cible de l'*intention* à partir de la ou des parties de produit décrivant la *situation*.

Un guide méthodologique peut être décrit par un *patron* ou une *directive.* Un *patron* offre une solution générique à un problème de conception fréquemment rencontré dans un contexte bien déterminé. Il peut être *composé* ou *atomique*. Une *directive* est définie comme « un ensemble d'indications qui donnent la manière de procéder pour atteindre un objectif ou réaliser une activité. »[2]. On distingue trois types de directives : les *directives informelles*, les *directives tactiques* et les *directives stratégiques*.

- Une *directive informelle* est une directive qui n'est pas structurée. Elle explique de manière textuelle comment procéder pour obtenir le produit cible.
- Une *directive tactique* est une directive complexe composée de sous-directives. Elle utilise une structure d'arbre composé de contexte qui peut être de type *plan*, de type *choix* ou de type *exécutable*. Ce type de directive est utilisé pour modéliser des *processus contextuels* permettant de présenter à la fois les activités composant un processus et les décisions et les contextes qui conduisent à les exécuter.
- Une *directive stratégique* a une structure de graphe, elle donne une vue stratégique de la démarche de développement du produit basée sur un ensemble d'*intentions* à satisfaire et un ensemble de *stratégies* pour satisfaire ces intentions. Une directive stratégique permet d'exprimer le processus de développement en proposant plusieurs chemins possibles pour satisfaire son intention. Ce type de directive est dédié à la modélisation des modèles de processus *orienté-stratégie* présentés ci-dessus ; elle est représentée sous forme d'une carte et d'un ensemble de directives associées [Rolland 99], [Benjamen 99].

Le méta-modèle présenté ci-dessus offre plusieurs possibilités à l'ingénieur de méthodes pour la définition du ou des modèles de processus d'une méthode selon les spécificités et le contexte de celle-ci. En effet, conformément à ce méta-modèle le modèle de processus d'une méthode peut être formalisé sous la forme d'un catalogue de patrons, sous la forme d'une directive stratégique (une carte et un ensemble de directives), sous la forme de directives informelles, sous la forme de directives tactiques ou encore avec une combinaison de ces formes. La section 5, consacrée à la présentation du méta-modèle de processus, aborde cet aspect en détail.

Les deux sections suivantes sont consacrées à la présentation détaillée du méta-modèle de produit et du méta-modèle de processus.

[2] Dictionnaire le Petit Robert 2000

4. Le Méta-Modèle de produit

Cette section présente le *méta-modèle de produit* développé dans ce travail. Ce méta-modèle est inspiré de celui proposé par V. Plihon [Plihon 96]. Tout modèle de produit de méthode utilisé dans ce travail est instance de ce méta-modèle.

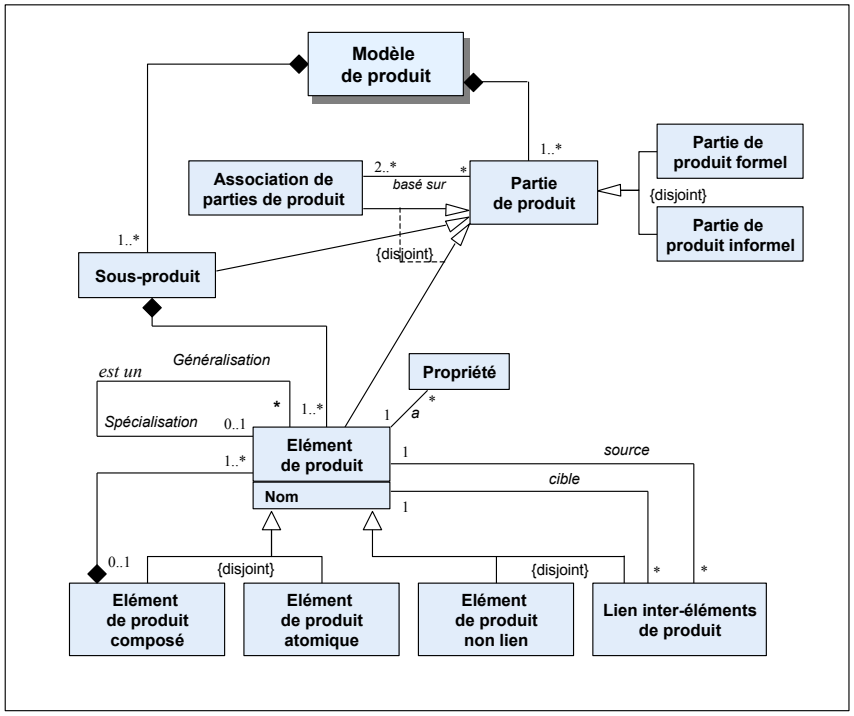


Figure 10. Méta-modèle de produit

Ce méta-modèle reprend la partie produit du méta-modèle de méthodes présenté à la section précédente et détaille le concept d'élément de produit. Comme le montre la Figure 10, on distingue les *éléments de produit* en fonction de deux caractéristiques qui sont le *type* et la *complexité* de l'élément. Ceci conduit à spécialiser la notion d'élément de produit en deux partitions, l'une permettant de faire la distinction entre les *éléments de produit non lien* et les *liens inter-éléments de produit*; et l'autre permettant de distinguer les *éléments de produit atomiques* des *éléments de produit composés*.

Les *éléments de produit non lien* représentent des concepts du produit qui ont une existence propre, le concept classe dans le modèle objet de la méthode OMT [Rumbaugh 95] est un exemple d'élément de produit non lien. Les *liens inter-éléments de produit* traduisent une relation existante entre deux éléments de produit, le concept association entre deux classes dans le modèle objet de la méthode OMT est un exemple de lien inter-élément de produit. Les liens inter-éléments de produit n'ont pas de sens si on les considère seuls sans leurs extrémités.

Les *éléments de produit atomiques* ne sont pas décomposables. Ils sont créés, modifiés et supprimés par des actions atomiques. Les *éléments de produit composés* se décomposent en éléments de produit,

pouvant être eux-mêmes atomiques ou composés. Un attribut appartenant à une classe est un exemple d'élément de produit atomique. Une classe, composée d'attributs et de méthodes, est un exemple d'élément de produit composé.

Chaque élément de produit a une ou plusieurs *propriétés*, le domaine d'un attribut dans le modèle objet de la méthode OMT est une instance de propriété.

Comme l'illustre la Figure 10, trois types de lien inter-élément de produit sont définis dans le méta-modèle de produit (i) le *lien d'héritage* modélise par le lien *Est-un* entre deux éléments de produit, (ii) le *lien d'agrégation* entre un élément de produit composé et un élément de produit et enfin (iii) le *lien d'association* entre deux éléments de produit.

Comme exemple d'instanciation de ce méta-modèle nous prenons une partie du modèle de produit de la méthode Lyee développée au chapitre 6 de ce travail. La Figure 11 présente le concept central de ce modèle, le *Process Route Diagram (PRD)*. Le *PRD* fournit le *cadre* pour structurer les programmes Lyee. C'est un graphe dirigé dans lequel les nœuds sont des instances d'une structure de base appelée *Scenario Function (SF)*. Un *SF* est une agrégation de trois sous-structures appelées chacune *Pallet (W04*, *W02* et *W03)*.

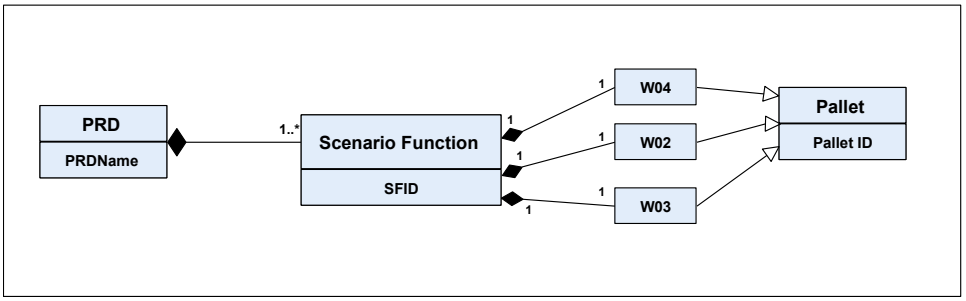


Figure 11. Partie du modèle de produit de la méthode Lyee

La Figure 12 illustre comment le modèle de produit présenté ci-dessus instancie le méta-modèle de produit proposé. Comme le montre cette Figure le concept *PRD* est une instance du méta-concept *élément de produit.* Il est considéré comme un *élément de produit composé* puisqu'il est composé d'un élément de produit plus fin qui est *Scenario Function*. Le lien d'héritage entre *Pallet* et *W04* est un instance du lien *Est-un* du méta-modèle. L'élément de produit *W04* est instance à la fois du concept *élément de produit non-lien* et du concept *élément de produit composé*.

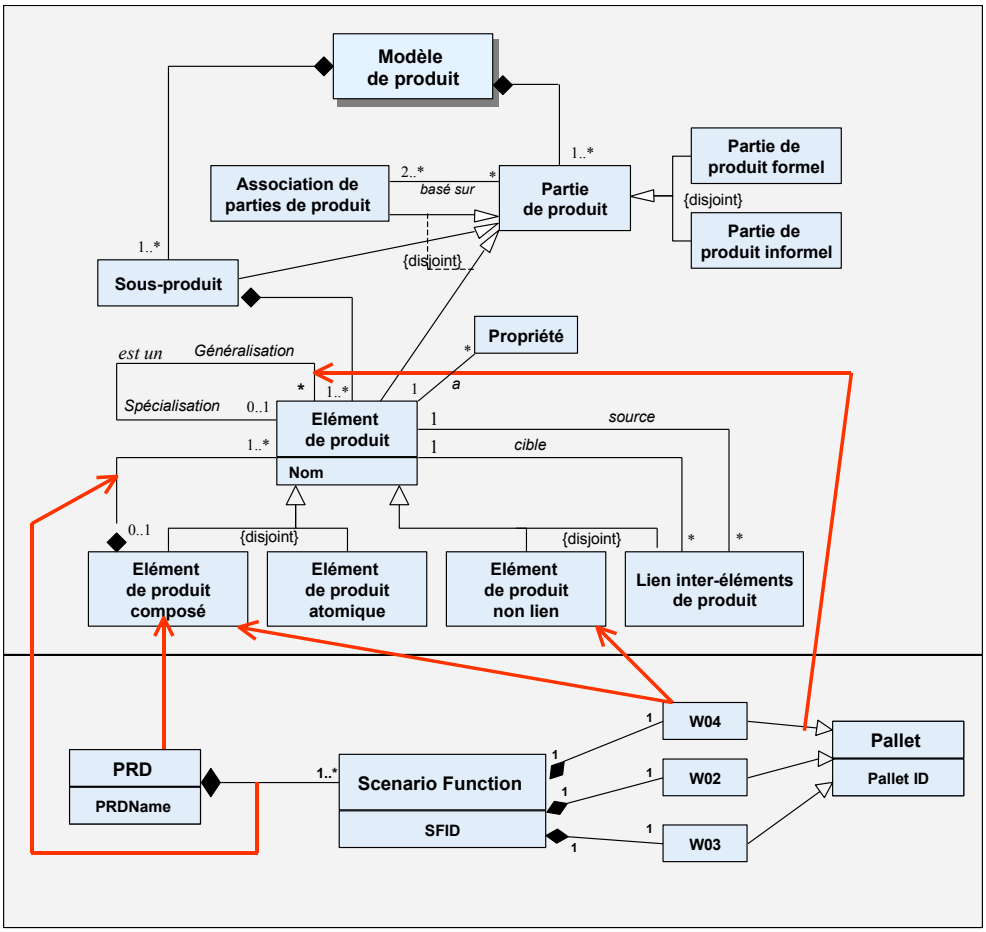


Figure 12. Exemple d'instanciation du méta-modèle de produit

5. Le Méta-Modèle de processus

Cette section présente le méta-modèle de processus qui a été développé sur la base d'un certain nombre de méta-modèles disponibles dans la littérature [Jarke 99], [Rolland 95], [Rolland 99]. La Figure 13 présente ce méta-modèle.

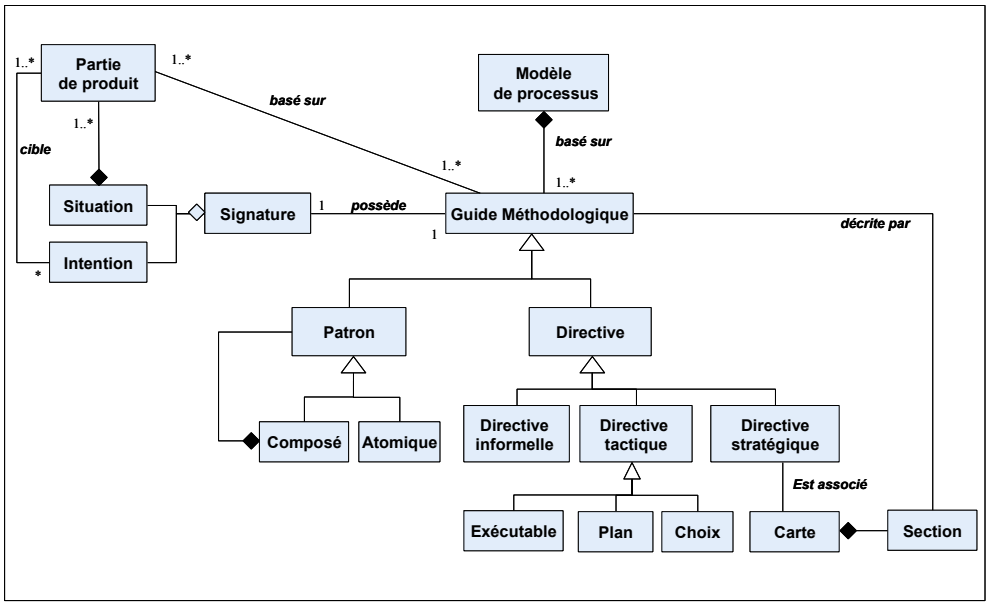


Figure 13. Méta-modèle de processus

Comme nous l'avons énoncé à la section 3, nous définissons tout modèle de processus comme une agrégation d'un ou plusieurs *guides méthodologiques*. Un guide méthodologique peut être décrit par un *patron* ou une *directive* de type *stratégique*, *tactique* ou *informelle*. Le choix parmi ces différentes formes de modélisation de processus dépend des objectifs définis pour la méthode, de la nature et de la complexité du produit souhaité par l'application de la méthode et de la situation d'ingénierie de celle-ci. La différence entre l'utilisation de *patron* ou de *directive* pour la formalisation du modèle de processus d'une méthode réside dans le fait qu'une directive est un moyen d'atteindre un but appartenant à la méthode dans un contexte défini tandis qu'un patron propose une solution générique à un problème de conception fréquemment rencontré lors de l'application de la méthode. Ces deux formalismes de représentation sont complémentaires pour la définition du modèle de processus d'une méthode.

Comme défini à la section 3, tout guide méthodologique a une signature qui est constitué un couple <*situation*, *intention*>. La *signature* représente la partie visible d'un guide méthodologique.

- *Situation*

La situation dans la signature d'un guide méthodologique identifie une partie de produit en cours de développement et nécessaire à la satisfaction de son intention. Chaque partie de produit référencée dans la situation est un élément du modèle de produit de la méthode. 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.

- *Intention*

Une intention est un but, un objectif qu'un ingénieur d'application a en tête à un moment donné du processus d'ingénierie. Par exemple, *"Construire un modèle des cas d'utilisation"* est une des intentions que l'on peut exprimer dans la démarche RUP.

Pour la représentation d'une intention nous utilisons la structure proposée par N. Prat dans [Prat 97], [Prat 99]. Suivant cette structure, nous représentons une intention par un verbe et un ou plusieurs paramètres. Chaque paramètre joue un rôle différent par rapport au verbe. La Figure 14 montre les différents paramètres d'une intention.

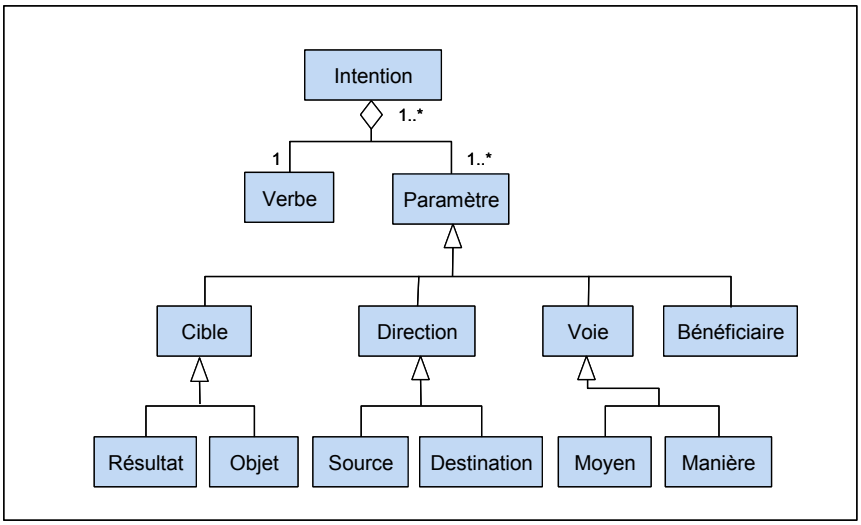


Figure 14. Structure d'une intention

Une intention doit être écrite sous la forme suivante :

Intention :: Verbe <Cible> [<Paramétres>*]

Les paramètres sont les suivants :

- Une *Cible* qui peut être un *Résultat* ou un *Objet* : la cible désigne la ou les partie(s) de produit affectée(s) par la réalisation de l'intention. L'objet représente la partie du produit déjà existante sur laquelle l'intention opère un changement, tandis qu'un résultat représente une partie de produit qui est créée par la satisfaction de l'intention ;
- Une *Direction* pouvant être une Source ou une Destination. La source montre l'emplacement initial de la partie produit sur laquelle l'intention va opérer. Alors que la destination indique l'emplacement de la partie produit après son utilisation ou sa création ;
- Une *Voie* qui peut être soit un *Moyen* soit une *Manière*. Le moyen est un instrument, un outil tandis que la manière est une approche ;
- Enfin un *Bénéficiaire* : c'est une entité qui représente une personne ou un système pour laquelle l'intention va être réalisée.

L'intention *"Construire un modèle des cas d'utilisation"* peut être ainsi décomposée comme suit : *(Construire)*$_{verbe}$ *(un modèle des cas d'utilisation)*$_{résultat}$.

5.1. Présentation de la notion de directive

Une *directive* représente une activité spécifique dans le développement des systèmes, réutilisable en tant qu'unité de processus indépendante et pouvant être intégrée dans différentes méthodes d'ingénierie des systèmes. Elle sert à guider l'ingénieur d'application pour réaliser une intention dans un contexte bien déterminé en lui suggérant un processus à suivre.

Dans ce qui suit on présente en détail les trois types de directives : *stratégique, tactique* et *informelle*

5.1.1. Directive Stratégique

Certaines méthodes proposent plusieurs manières possibles pour développer les produits cibles. Dans de tels cas, l'ingénieur d'application à la possibilité de choisir une démarche parmi plusieurs suggérées par la méthode. Nous proposons d'utiliser une directive stratégique pour formaliser l'ensemble des démarches au sein d'un même modèle de processus.

Une directive stratégique permet de représenter un processus de développement *intentionnel* et *multi-démarches*. Nous nous inspirons du formalisme MAP pour modéliser ce type de directive. *MAP* est un formalisme de représentation des processus d'ingénierie de SI qui a été proposé par C. Rolland et A. Benjamen dans [Benjamen 99], [Rolland 99]. Ce formalisme permet une *représentation intentionnelle* des processus en mettant l'accent sur les buts à atteindre et non sur les actions qui en sont les expressions opérationnelles. Il présente l'avantage d'offrir une représentation *multi-modèles* permettant la construction du processus d'ingénierie de manière dynamique.

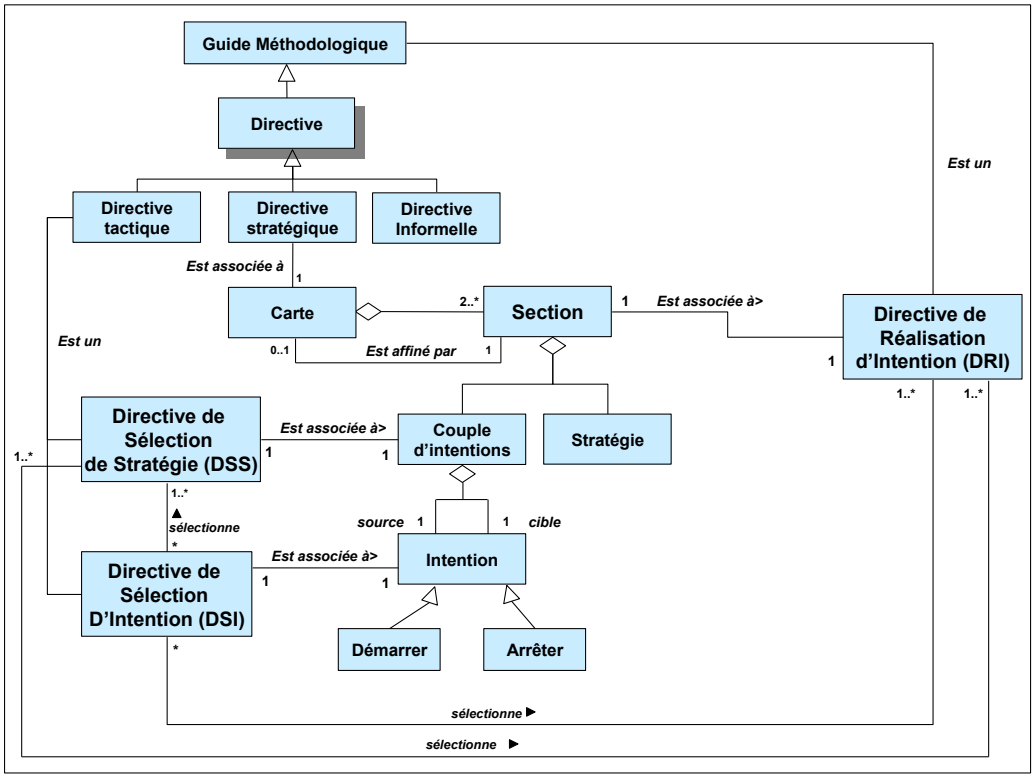


Figure 15. Structure d'une directive stratégique

Comme l'illustre la Figure 15, a chaque directive stratégique on associe une *carte*[3] qui est le paradigme sous-jacent au formalisme MAP. La carte se présente comme un graphe dirigé. Elle est constituée d'un ensemble de *sections*. Chaque *section* d'une carte est composée d'un nœud source, d'un autre nœud cible et d'un arc qui les relie. Les nœuds du graphe correspondent aux *intentions* que l'utilisateur veut réaliser. Les arcs du graphe décrivent les *stratégies* offertes pour progresser d'une intention à une autre. Une intention représente un but de processus à atteindre tandis qu'une stratégie est une façon d'atteindre le but qui en est la cible. Une section de la carte est définie par le triplet <*Intention source, Intention cible, Stratégie*>.

La Figure 16 illustre un exemple d'une carte contenant cinq intentions : *Démarrer*, I_i, I_j, I_k et *Arrêter*. Sur la base de ces intentions et des stratégies les reliant on détecte sept sections composant la carte: <*Démarrer*, I_i, S_i>, <*Démarrer*, I_k, S_k>, < I_i , I_j, S_{ij1}>, < I_i , I_j, S_{ij2}>, < I_j , I_k, S_{jk}>, < I_k , I_i, S_{ki}>, < I_k , *Arrêter*, S_T>.

[3] Dans la suite de ce travail nous faisons la différence entre *MAP* qui correspond au modèle de représentation et la *carte* qui constitue l'instance de ce modèle.

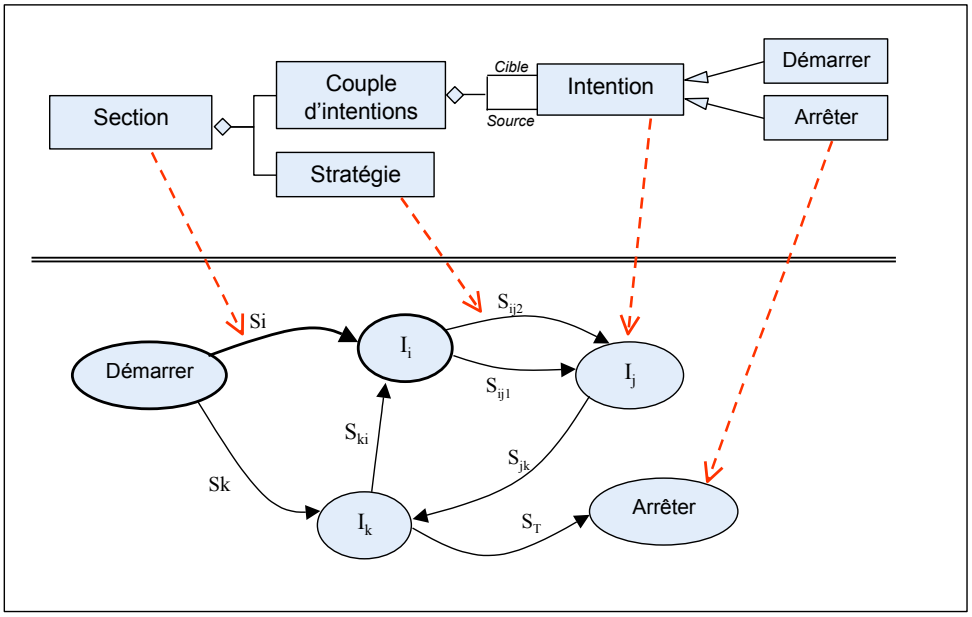


Figure 16. Exemple d'une carte

Trois types de relations peuvent exister entre les sections d'une carte.

- *ET/OU* : le lien ET/OU entre deux sections ayant la même intention source et la même intention cible, indique que celles-ci peuvent être exécutées soit ensemble (ET), ou que l'une ou plusieurs d'elles peuvent l'être (OU). Les deux sections $< I_i , I_j, S_{ij1}>$ et $< I_i , I_j, S_{ij2}>$ de la carte présentée à Figure 16 sont relié part un lien ET/OU.

- OU_{ex} : ce lien représente la notion de *cluster* dans une carte. Si deux sections, ayant une même intention source et une même intention cible sont reliées par ce lien, l'exécution de l'une rend l'exécution de l'autre interdite. Les deux sections sont alors mutuellement exclusives.

- *Enchaînement* : ce lien représente la relation d'enchaînement entre deux sections. Il décrit le cas ou l'intention cible de l'une représente l'intention source de l'autre. Ce lien permet de définir la notion de *chemin*. Un *chemin de navigation* dans une carte décrit les sections parcourues en partant de l'intention *Démarrer* pour arriver à l'intention *Arrêter*. La succession des sections $<$*Démarrer,* I_i, $S_i>$, $< I_i , I_j, S_{ij1}>$, $< I_j , I_k, S_{jk}>$ et $<I_k$,*Arrêter,* $S_T>$ constitue un exemple de chemin entre les intentions *Démarrer* et *Arrêter* de la carte présentée à la Figure 16.

Comme le montre la Figure 15, la section d'une carte peut être affinée par une carte d'un niveau d'abstraction inférieur. En fait, la relation *affiné par* permet de décrire une section à un niveau d'abstraction donné (i), par une carte complète à un niveau d'abstraction moins élevé (i+1). Cette opération s'appelle *affinement*. La Figure 17 illustre l'affinement d'une section de la carte.

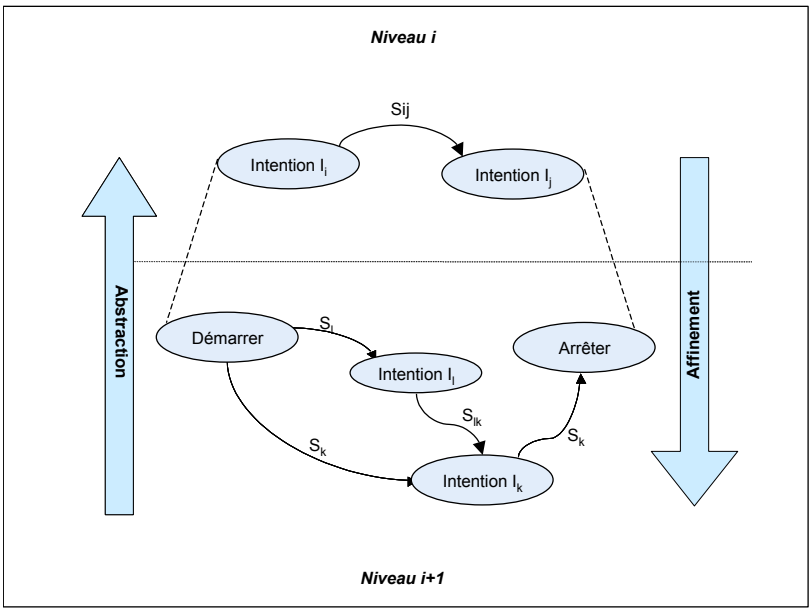


Figure 17. Exemple d'affinement d'une section

Comme le montre la Figure 15, à chaque section de la carte est associée une *Directive de Réalisation d'Intention* (*DRI*), ce type de directive offre un guidage à l'ingénieur pour la réalisation de l'intention cible de la section. Par ailleurs, deux types de directives permettent d'assister l'ingénieur pour la sélection d'un chemin de navigation dans la carte : les *Directives de Sélection d'Intention (DSI)* et les *Directives de Sélection de Stratégie (DSS)*. Dans ce qui suit, on présente en détail ces trois types de directives.

5.1.1.1. Directives de Réalisation d'Intention

Une *Directive de Réalisation d'Intention (DRI)* 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. Le rôle de la DRI est donc d'opérationaliser une intention pour aboutir à la transformation du produit.

Dans la carte, il existe une DRI pour chaque triplet $<I_i,I_j,S_{ij}>$. Elle aide l'ingénieur d'application dans l'accomplissement de l'intention I_j en suivant la stratégie S_{ij}.

Toute DRI est décrite par un guide méthodologique (Cf. Figure 15). Selon la complexité de l'intention à atteindre et la nature du produit cible, ce guide méthodologique peut prendre la forme d'un patron, d'une directive informelle, d'une directive tactique (de type plan ou choix) ou encore d'une directive stratégique (utilisation de la relation d'*affinement*).

Une DRI peut donc proposer plusieurs étapes pour atteindre le résultat final qui est d'exécuter des actions sur le produit en cours de développement. Une DRI peut, en conséquence, inclure la

décomposition de l'intention initiale en sous-intentions qui peuvent elles-mêmes être décomposées jusqu'à rencontrer des intentions exécutables sous forme d'actions sur le produit.

Comme toute directive, la signature d'une DRI associée à une section $<I_i, I_j, S_{ij1}>$ de la carte est un couple <situation, intention> construit comme suit :

- La situation comporte la partie de produit qui est la cible de l'intention *Ii* et dont l'état peut être précisé par une condition d'occurrence,
- L'intention est exprimée sous la forme *Ij Sij1*.

5.1.1.2. Directive de Sélection de Stratégie

Une *Directive de Sélection de Stratégie (notée DSS)* aide à choisir une stratégie. Elle s'applique lorsque l'intention source I_i et l'intention cible I_j sont déterminées et lorsqu'il y a plusieurs stratégies S_{ij1}, S_{ij2}, ..S_{ijn} possibles pour progresser de I_i à I_j. Le rôle de la DSS est de guider la sélection de la stratégie S_{ijk} la plus appropriée pour la situation donnée. La sélection d'une stratégie conduit à la sélection de la DRI pour le triplet sélectionné (I_i, I_j, S_{ijk}).

Autrement dit, pour chaque couple d'intentions (I_i, I_j) connectées par plus d'une stratégie de même direction, il existe une DSS (Cf. Figure 15). La signature de cette directive est exprimée de la manière suivante :

- La situation comporte la partie de produit qui est la cible de l'intention Ii ; une condition d'occurrence peut préciser l'état de cette partie de produit,
- L'intention est exprimée sous la forme : ***Progresser*** $_{\text{verbe}}$ (***vers*** *Ij*) $_{\text{cible}}$.

Le verbe *Progresser* est toujours utilisé pour exprimer les intentions des DSS d'une manière uniforme. De plus, le mot *vers* précise quelle est l'intention cible de la progression.

Une DSS est toujours une directive tactique de type choix. Elle est composée de directives exécutables, une par section à sélectionner. Chacune de ces directives exécutables contient une action de délégation qui sélectionne une DRI associée à la section.

5.1.1.3. Directive de Sélection d'Intention

Une *Directive de Sélection d'Intention (notée DSI)* aide à choisir l'intention à réaliser dans l'étape suivante du processus. Elle s'applique lorsqu'une intention I_i vient d'être réalisée et lorsque l'ingénieur d'application s'appuie sur la carte pour déterminer quelle sera l'intention à réaliser à la prochaine étape.

Etant donnée une intention I_i, une *DSI* identifie l'ensemble d'intentions $\{I_j\}$ qui peut être accompli dans la prochaine étape et sélectionne l'ensemble correspondant des *DRI* ou des *DSS*. Le premier

s'applique lorsqu'il n'y a qu'une section entre I_i et I_j alors que le second s'applique lorsqu'il y a plusieurs sections entre I_i et I_j.

Pour chaque intention I_i, il existe une DSI (Cf. Figure 15) . La signature de cette directive est exprimée comme suit :

- la situation comporte la partie de produit qui est la cible de l'intention I_i ; elle peut être précisée par une condition d'occurrence,
- l'intention est exprimée sous la forme : ***Progresser*** $_{verbe}$ ***(depuis** Ii)* $_{source}$

L'intention de la signature d'une DSI exprime le fait que la directive aide l'ingénieur d'applications à progresser dans la carte. Le verbe *"Progresser"* est utilisé à ce propos. Le paramètre *source* exprime l'intention qui est la source de progression.

Comme une DSS, la DSI est toujours une directive tactique de type choix, est composé d'un ensemble de directives exécutables comportant des actions de délégation.

5.1.1.4. Invariant et Règles de validité de la carte

La modélisation d'une *carte* associée à une directive stratégique doit respecter les règles de construction de la *carte*. En effet, pour qu'une *carte* soit correcte, elle doit respecter un certain nombre d'*invariants* et pour qu'elle soit valide elle doit vérifier des *règles de validité.*

- *Invariants de la carte :*

Un *invariant* de la carte est une propriété que la carte doit toujours vérifier [Rolland 04], [Banerjee 87]. I. Zoukar [zoukar 05] a identifié les trois invariants suivants :

I1. Toute carte a une et seulement une intention qui n'est la cible d'aucune stratégie ; c'est l'intention *Démarrer*.

I2. Toute carte a une et seulement une intention qui n'est la source d'aucune stratégie ; c'est l'intention *Arrêter*.

I3. Toute intention dans une carte doit pouvoir se réaliser au moins une fois, c'est-à-dire qu'il existe un chemin qui la relie à l'intention *Démarrer*.

A partir de ces trois invariants, on constate plusieurs corollaires :

C1. Les cartes sont des graphes connexes ; il n'y a aucune intention ou stratégie isolée.

C2. Chaque intention dans une carte est la source d'une stratégie exceptée l'intention *Arrêter*.

C3. Chaque intention dans une carte est la cible d'une stratégie exceptée l'intention *Démarrer*.

C4. Il y a toujours un chemin de *Démarrer* à *Arrêter*.

C5. Chaque section d'une carte appartient à un chemin entre *Démarrer* et *Arrêter*.

- *Règles de validité de la carte :*

Afin que la carte soit valide, elle doit vérifier les règles suivantes [Rolland 01] :

R1 : Les intentions de la carte doivent être de même niveau d'abstraction.

R2 : Aucune intention / stratégie de la carte ne doit pouvoir être considérée comme un sous-ensemble d'une autre.

R3 : Aucune intention ne doit apparaître dans une carte comme une manière d'en réaliser une autre.

R4 : Les intentions donnant le même produit doivent être agrégées.

R5 : Les sections représentant des manières mutuellement exclusives de produire un même résultat doivent être regroupées en cluster.

R6 : Les intentions considérées comme parties d'un processus « tout ou rien », c'est-à-dire partie d'une transaction, doivent être abstraites sous la forme d'une intention.

R7 : Les intentions qui se complémentent mutuellement et vont de pair doivent être agrégées sous la forme d'une intention unique qui les abstrait.

R8 : Une intention ne doit pas correspondre à une opération simple de manipulation ou de création d'une partie de produit. Elle est plus abstraite et exprime ce que sont les caractéristiques finales désirées de la partie de produit.

5.1.2. Directive Tactique

La structure de *directive tactique* proposée dans ce mémoire est inspirée de l'approche contextuelle de modélisation des processus NATURE [Rolland 95], [Grosz 97], [Jarke 99]. Cette approche représente le modèle de processus d'une méthode au moyen de la notion de *contexte* et d*'arbre de contextes* [Plihon 96], [Rolland 94c], [Plihon 94], [Plihon 95].

La Figure 18 détaille la structure d'une directive tactique en complément de la Figure 9. Cette Figure est commentée au fur et à mesure de la présentation des trois types de directives tactiques : *exécutable*, *choix* et *plan*.

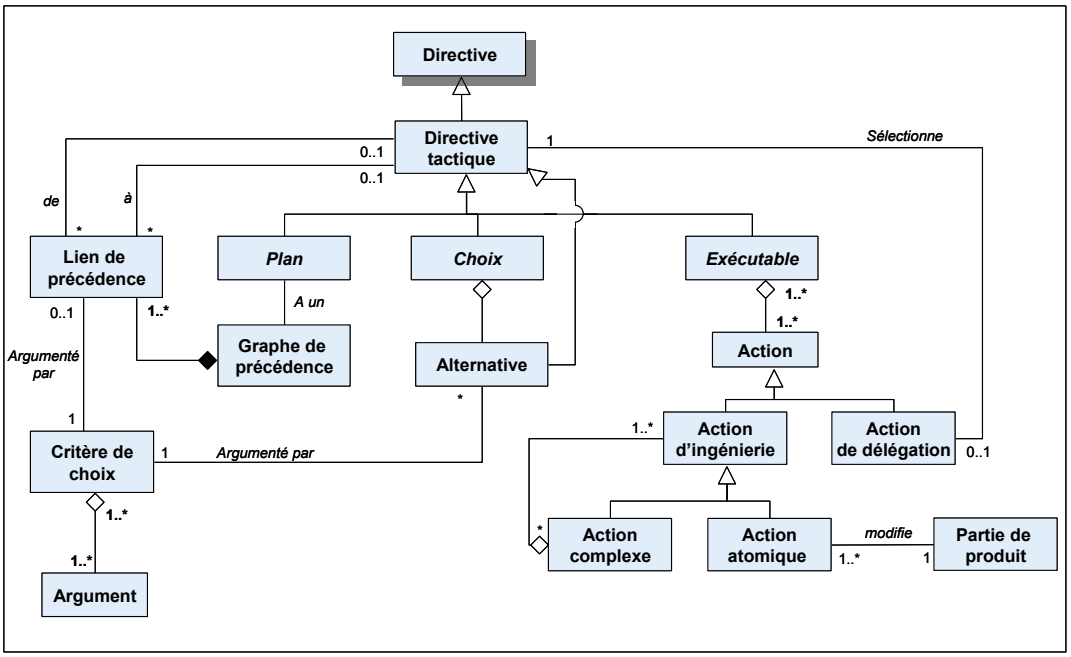


Figure 18. Structure d'une directive tactique

5.1.2.1. Directive tactique de type exécutable

Une *directive exécutable* correspond à une intention qui peut être concrétisée par une ou plusieurs *actions*. Une *action* peut être une action de transformation du produit ou une action de sélection d'une autre directive. Ces deux types d'action sont modélisés comme des types spécialisés du concept *action*, *Action d'ingénierie* et *Action de délégation* (Figure 18). L'action *d'ingénierie* consiste à modifier le produit en cours de développement tandis que l'action de *délégation* consiste à déléguer la réalisation d'une intention à une autre directive tactique.

Une action d'ingénierie peut être *atomique* ou *complexe* (composée d'une ou plusieurs actions d'ingénierie). Exécuter une action d'ingénierie modifie une ou plusieurs parties du produit associées à la directive et peut générer une nouvelle situation qui est elle-même sujette à de nouvelles intentions.

5.1.2.2. Directive tactique de type choix

Une directive choix correspond à une situation qui nécessite l'exploration de différentes possibilités alternatives. Dans ce cas, l'ingénieur d'application a différentes façons d'atteindre le but qu'il poursuit. Il doit faire un choix parmi un ensemble de possibilités permettant de résoudre le problème. Chaque solution alternative est décrite par une sous-directive qui peut être de type exécutable, plan ou choix.

La directive choix permet de décomposer une intention en sous-intentions alternatives en l'affinant en intentions plus fines. Les alternatives d'une directive choix affinent l'intention de cette dernière,

soit en ajoutant une information sur l'approche suivie pour mettre en œuvre l'intention, soit en décrivant les différentes transformations du produit qui peuvent être menées pour atteindre l'objectif de la directive choix.

Comme le montre la Figure 18, le choix parmi les différentes alternatives d'une directive choix est argumenté par des *critères de choix*. Un *critère de choix* est une combinaison d'*arguments* en faveur ou en défaveur du choix d'une alternative. Il aide l'ingénieur d'applications à choisir l'alternative la mieux appropriée aux caractéristiques de la situation qu'il traite. Les arguments sont basés sur des heuristiques ou sur les caractéristiques du produit en cours de développement. Ils sont atomiques et peuvent être réutilisés dans plusieurs critères de choix. Ils sont décrits en langage naturel.

La directive associée à l'intention *« Définir un lien entre deux classes du schéma statique »* est un exemple de directive choix. La Figure 19 illustre cette directive, elle offre à l'ingénieur d'application trois possibilités selon le type de la relation identifiée entre les deux classes :

- *Définir un lien Est-un entre deux classes*,
- *Définir un lien d'agrégation entre deux classes*, ou
- *Définir un lien d'association entre deux classes*.

Les arguments (a1), (a2) et (a3) présentés en bas de la figure permettent d'assister l'ingénieur dans le choix de la directive adéquate à sa situation.

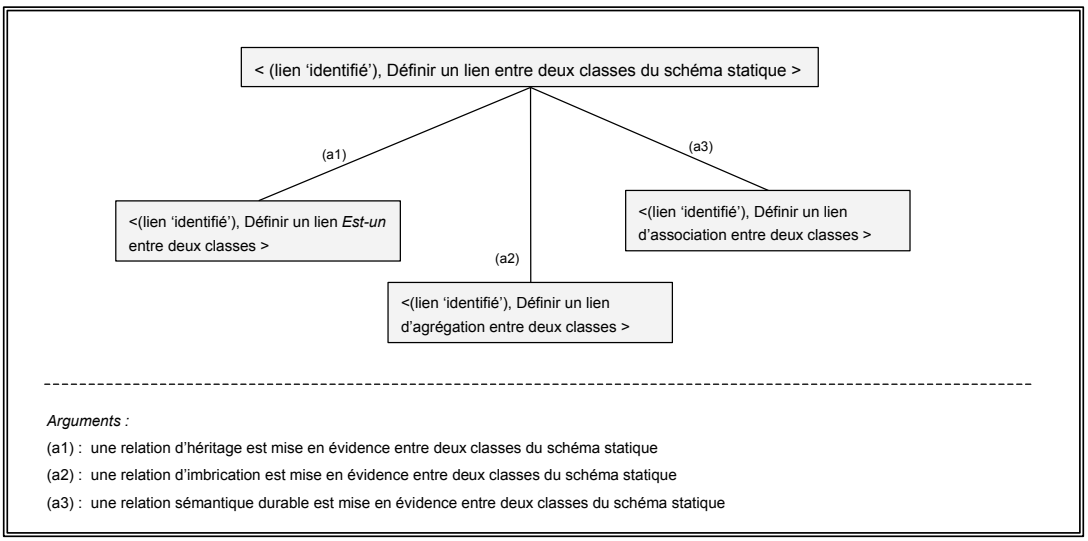


Figure 19. Exemple de directive choix

5.1.2.3. Directive tactique de type plan

Une directive *plan* correspond à un problème complexe qui, pour être résolu, nécessite d'être décomposé en un ensemble de sous-problèmes. L'ingénieur d'application connaît l'ensemble des décisions qui lui permettront d'atteindre le but qu'il poursuit. Ce type de directive propose un plan

composé d'un ensemble de sous-directives qui peuvent être des directives tactiques de type exécutable, plan ou choix.

L'ordre d'exécution des sous-directives est défini dans un *graphe de précédence* (Figure 18). Il y a un graphe par directive plan. Les nœuds du graphe sont des directives (les composants 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 de précédence permettent de prescrire les conditions d'occurrence d'une transition. Ils sont construits à partir d'*arguments* et sont de même nature que ceux attachés aux alternatives des contextes choix. Leur but est, ici, d'aider l'ingénieur d'applications à choisir le chemin à suivre pour l'exécution du plan.

Dans certains cas, le graphe de précédence peut être simple et ne proposer qu'un seul chemin pour exécuter le plan. Il est alors inutile d'associer des critères de choix aux liens. C'est le cas, par exemple, des plans dont les composants sont organisés en séquence. Mais un graphe peut aussi contenir plusieurs chemins d'exécution. Ceci permet d'introduire plus de flexibilité dans la démarche d'une directive.

Exécuter un plan revient à parcourir son graphe de précédence. Lors de l'exécution d'un contexte plan, les critères de choix des liens de précédence sont calculés grâce à l'évaluation de leurs arguments.

La directive associée à l'intention « *Définir une classe dans le schéma statique* » est un exemple de directive plan, ce plan présenté à la Figure 20 propose à l'ingénieur de (1) Créer la classe identifié, (2) Définir le nom de la classe et (3) Définir les attributs de la classe. Ces trois sous-directives s'exécutent d'une manière séquentielle ce qui explique l'absence d'un graphe de précédence dans cette directive.

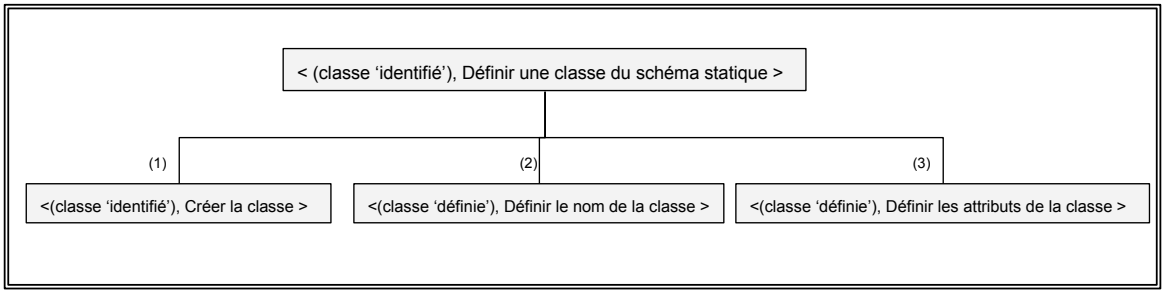


Figure 20. Exemple de directive tactique de type plan

5.1.3. Directive Informelle

Une *directive informelle* est une directive qui n'est pas structurée. Comme elle n'a pas de structure, une directive informelle est un élément atomique dans la démarche d'une méthode. Elle peut aussi

être tout simplement un élément d'une autre directive, plus importante. Une directive informelle est associée à une description textuelle.

La plupart des méthodes et des approches proposées dans la littérature ne sont pas définies d'une manière formelle. Elles utilisent rarement la méta-modélisation pour définir le produit et la démarche de la méthode. Même si leurs modèles de produit sont présentés avec un niveau de détail suffisant pour être formalisé, les modèles de processus sont souvent omis ou décrits d'une manière informelle sous forme d'hypothèses et de conseils à suivre.

Une directive informelle ne propose pas de démarche détaillée à suivre pour aboutir au résultat attendu. Elle définit seulement des hypothèses, des règles, des conditions à respecter et des contraintes à ne pas violer. Elle peut aussi définir quel type de résultat à obtenir sans préciser pour autant comment procéder formellement pour l'obtenir. Des conseils et des exemples peuvent être proposés à l'ingénieur d'applications pour le guider dans la satisfaction de l'intention.

5.2. Présentation de la notion de patron

Un modèle de processus est basé sur un ou plusieurs guides méthodologiques qui peuvent être de type directive ou *patron*. Le concept de *patron* a été introduit par Christopher Alexander [Alexander 77] qui a utilisé les patrons dans le domaine de l'architecture des bâtiments. Cet architecte a introduit l'idée d'un langage de patron permettant à des personnes non-spécialistes de construire leurs propres bâtiments. Ce langage est formé d'un ensemble de patrons où chacun donne la solution d'un problème particulier rencontré lors de la construction.

C. Alexander définit un patron comme « *un problème qui se produit souvent dans notre environnement et qui décrit alors le cœur de la solution à ce problème, d'une telle façon que l'on peut utiliser cette solution un million de fois, sans toutefois faire la même chose deux fois* ».

Plusieurs autres définitions du terme de *patron* ont été proposées :

- Pour Peter Coad [Coad 92], un patron est *une forme entièrement réalisée, originale ou un modèle accepté ou proposé pour une imitation ; quelque chose qui est vue comme un exemple normatif pouvant être copié, archétypé ou utilisé comme exemple*; un patron orienté objet est *une abstraction d'un doublet, un triplet ou autre petit groupe de classes qui peut être utile encore et encore dans tout développement orienté objet.*
- Dans [ACM 96] *un patron a pour but de décrire avec succès des solutions à des problèmes logiciels communs et d'aider les gens à réutiliser des pratiques vraies et résolues.*
- Pour C. Rolland [Rolland 98], un patron est *plus qu'une description d'un élément du monde réel, c'est aussi une règle décrivant quand et comment créer cet élément. Le patron doit à la*

fois inclure une description de cet élément et une description de la démarche qui permet de générer cet élément.

L'idée principale des patrons est alors d'associer une solution générique et réutilisable à un problème récurrent dans un contexte bien identifié.

Les patrons sont un sujet largement abordé par la communauté orientée objet. Ce sont des recueils de solutions qui ont fait leur preuve dans des cas pratiques et qui visent à résoudre des problèmes fréquemment rencontrés dans les projets de développement de logiciels. De nombreux patrons ont été développés pour couvrir les problèmes récurrents de développement. Depuis, l'utilisation des patrons s'est généralisée pour d'autres activités comme la conception système et logicielle [Coplien 95], [Coplien 00], [Gamma 93] et [Vlissides 96], la modélisation des données [Hay 96] et plus récemment, dans l'analyse des systèmes [Fowler 97a]. Tous ces efforts conduisent à réutiliser les meilleurs atouts d'une méthode dans une autre.

- ***La structure de patron***

La structure de patron retenue dans ce travail est présentée à la Figure 21.

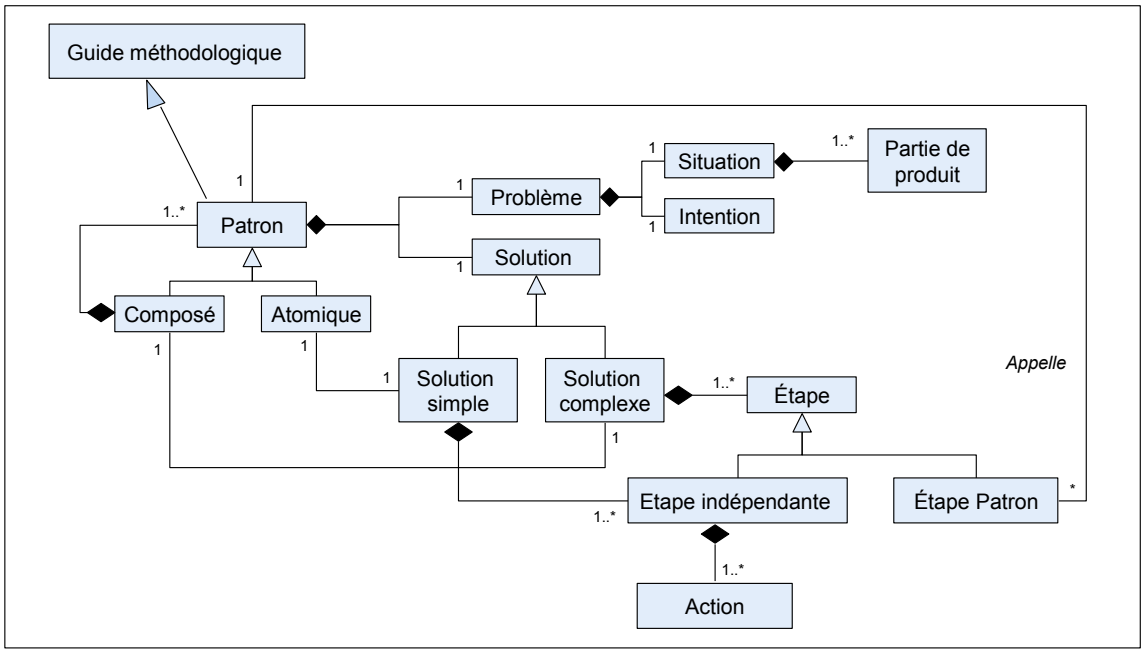


Figure 21. Structure de patron

Un patron est composé de deux parties le *problème* et sa *solution*. La partie *problème* est composée d'une *situation* et d'une *intention*. La *situation* est définie sur la base d'une ou plusieurs parties de produit, elle exprime une situation typique correspondant à un problème récurrent dans la construction du produit de la méthode. L'*intention* exprime un but qu'on souhaite atteindre par l'application du patron, la formalisation de cette intention respecte la structure (Verbe + Paramètres) définie ci-dessus. « *Résoudre les conflits entre le MCD et MCT* » est un exemple d'intention de la

méthode *Merise* pour laquelle on peut définir un patron permettant de guider le concepteur dans la réalisation de cette intention.

Un patron peut être *composé* ou *atomique*. Un patron *composé* correspond à un problème complexe qui nécessite d'être décomposé et peut être résolu par appel à d'autre patrons. Un patron *atomique* ne se décompose pas.

La *solution* d'un patron décrit la route à suivre pour obtenir la ou les parties de produit cible de l'intention à partir de la ou les partie de produit décrivant la situation. La définition de cette *solution* passe par la décomposition de l'intention du patron en un ensemble d'actions atomiques permettant la transformation du produit.

Comme le montre la Figure 21, un patron *atomique* possède une *solution simple* composée d'une ou plusieurs *étapes indépendantes*. Une étape indépendante est composée d'une ou plusieurs actions atomiques.

Un patron *composé* possède une *solution complexe* composée d'étapes qui peuvent être des *étapes indépendantes* et des *étapes patron*. Une étape patron contient une action de délégation permettant d'invoquer l'application d'un patron composant.

6. Conclusion

Nous avons consacré ce chapitre à la présentation du méta-modèle de méthode servant de base à ce travail. La méthode *MIME* que nous proposons est une instance de ce méta-modèle ; toute méthode devant être transformé par *MIME* doit aussi être une instance de ce méta-modèle.

Notre méta-modèle met en avant la notion de processus que la méthode propose par rapport à celle du produit qu'elle permet de construire. Il est essentiellement ciblé sur les notions de *directive* et de *patron*. La richesse de la structure de directive permet de représenter n'importe quel type de démarche en commençant par la plus simple (une instruction en langage naturel) jusqu'au modèle complexe de processus multi-démarches ayant une structure de graphe.

CHAPITRE 4 : RETRO-INGENIERIE DE METHODES

1. Introduction

Comme nous l'avons déjà indiqué dans l'introduction de ce mémoire, la méthode *MIME* comporte deux modèles de processus pour contrôler l'évolution d'une méthode : Un modèle de rétro-ingénierie et un modèle d'évolution. Le propos de ce chapitre est de présenter le modèle de rétro-ingénierie dont l'objectif est de permettre la formalisation de méthodes ne reposant pas sur de bases suffisamment précises. Le modèle définit une démarche de guidage de l'ingénieur de méthodes dans les différentes étapes de la rétro-ingénierie d'une méthode.

Ce chapitre est organisé comme suit : Une vue générale de la notion de rétro-ingénierie de méthodes est présentée en premier lieu. La deuxième partie est consacrée à la présentation du modèle de processus représenté par une *carte* et un ensemble de *directives*. La dernière partie est dédiée à la conclusion.

2. La notion de Rétro-ingénierie de méthodes

La *rétro-ingénierie* est une pratique courante dans le domaine de l'ingénierie du logiciel : elle vise à analyser des programmes existants dans le but d'en retrouver la structure et la spécification. Le résultat peut servir à documenter une application, à faciliter sa restructuration et son évolution. La rétro-ingénierie est aussi fréquemment utilisée dans le domaine des bases de données ; elle permet dans ce contexte de définir le schéma et la structure d'une base de données déjà existante.

Dans ce travail, nous avons recours à la technique de rétro-ingénierie dans le but de re-formuler une méthode. Il s'agit de la *rétro-ingénierie de méthodes*. Nous définissons la rétro-ingénierie de méthodes comme l'activité permettant, à partir d'une description initiale informelle d'une méthode, d'aboutir à une formalisation de celle-ci par la construction de ses modèles de produit et de

processus. Sur la base de nos observations, nous avons détecté deux situations pouvant déclencher un tel besoin d'ingénierie :

- *La description initiale de la méthode est informelle* : Dans ce cas de figure, la description initiale de la méthode s'arrête généralement à la présentation de la liste des concepts manipulés par la méthode et à la description succinte des étapes phares composant sa démarche. Cette situation pause plusieurs problèmes tels que : (i) la difficulté de compréhension de la méthode et de sa transmission à des novices et (ii) le risque d'erreurs encourues dans les projets de développement de SI en appliquant cette méthode. Ces erreurs sont dues généralement aux différentes interprétations de la méthode de la part de ses utilisateurs.
- *La description initiale de la méthode est mal définie :* Dans ce cas de figure la méthode est définie par un ensemble de modèles décrivant sa partie produit et/ou sa partie processus. Le problème réside dans le fait que ces modèles ne reflètent pas la méthode appliquée effectivement par les utilisateurs. Cet écart est généralement dû aux différentes transformations apportées à la méthode sans qu'elles aient été répercutées dans la description de la méthode.

Par ailleurs, le projet de rétro-ingénierie d'une méthode peut être entamé pour différents objectifs parmi lesquels on peut citer :

- La rétro-ingénierie d'une méthode sert à la formaliser et à la documenter. Ceci permet de faciliter la compréhension et l'apprentissage de cette méthode par ses utilisateurs.
- La rétro-ingénierie d'une méthode permet de la stocker dans une base de méthodes; la formalisation des modèles de la méthode doit alors respecter la structure de cette base.
- La rétro-ingénierie d'une méthode pour la formaliser et pouvoir y appliquer les techniques d'ingénierie de méthodes. Dans ce travail nous réalisons la rétro-ingénierie d'une méthode pour formaliser ses modèles et pouvoir les faire évoluer en utilisant notre démarche d'évolution de méthodes.

La démarche de rétro-ingénierie que nous proposons est décrite par un modèle de processus permettant de guider l'ingénieur de méthodes dans les différentes étapes de la rétro-ingénierie. Ce modèle de processus se base sur le méta-modèle de méthodes présenté au chapitre 4, les modèles de produit et de processus obtenus sont alors des instances de ce méta-modèle. La Figure 22 présente une vue générale de la solution que nous proposons pour la rétro-ingénierie de méthodes.

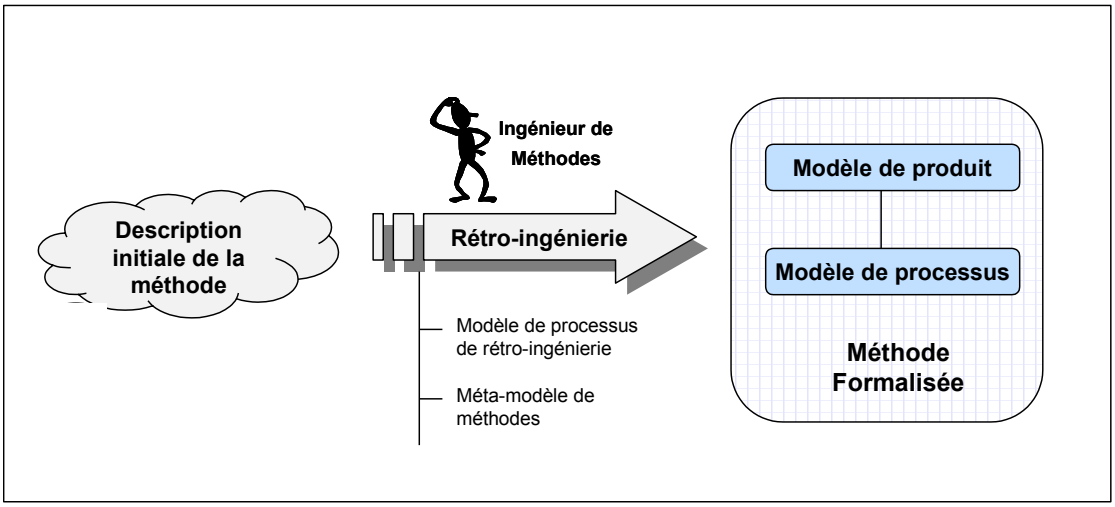


Figure 22. Vue générale de rétro-ingénierie de méthode

3. Le modèle de processus de Rétro-ingénierie

Le modèle de processus de rétro-ingénierie que nous proposons est formalisé par une directive stratégique, c'est-à-dire par une *carte* (la *carte de rétro-ingénierie*) et un ensemble de *directives*. La signature de cette carte est < (DIM[4]), *Faire la rétro-ingénierie d'une méthode* >. Les deux sections suivantes sont consacrées à la présentation la carte de rétro-ingénierie et à celle de ses directives.

3.1. La Carte Rétro-ingénierie

La rétro-ingénierie d'une méthode vise à formaliser ses modèles de produit et de processus. Un modèle de processus de rétro-ingénierie de méthodes doit alors assister l'ingénieur de méthodes dans l'identification et la modélisation des éléments composant ces modèles. De ce fait, la *carte de rétro-ingénierie* que nous proposons comporte trois intentions : *Identifier les éléments de la méthode*, *Formaliser le modèle de produit* et *Formaliser le modèle processus* et un ensemble de stratégies permettant d'atteindre ces intentions. Cette carte est présentée à la Figure 23.

[4] DIM : Description Initiale de la Méthode

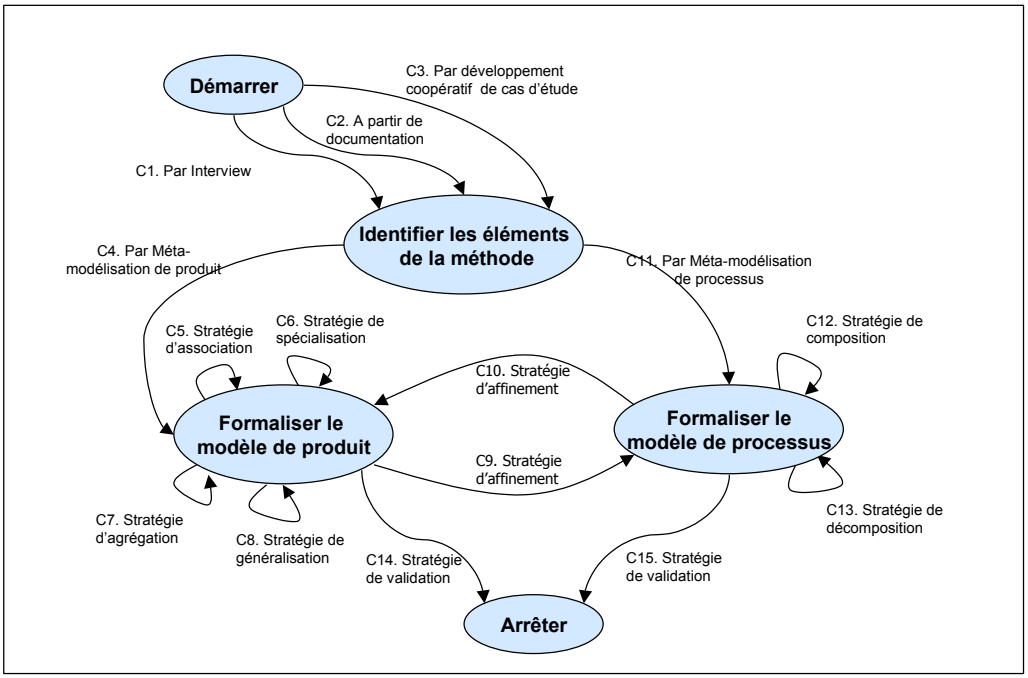


Figure 23. La carte de Rétro-ingénierie

La carte de *rétro-ingénierie* comporte quinze sections présentées au Tableau 1. Ces sections numérotées de C1 à C15 correspondent aux différentes stratégies permettant d'atteindre les différentes intentions de cette carte.

C_1 : < Démarrer, Identifier les éléments de la méthode, Par interview >
C_2 : < Démarrer, Identifier les éléments de la méthode, A partir de documentation >
C_3 : < Démarrer, Identifier les éléments de la méthode, Par développement coopératif de cas d'étude >
C_4 : < Identifier les éléments de la méthode, Formaliser le modèle de produit, Par méta-modélisation de produit >
C_5 : < Formaliser le modèle de produit, Formaliser le modèle de produit, Stratégie d'association >
C_6 : < Formaliser le modèle de produit, Formaliser le modèle de produit, Stratégie de spécialisation >
C_7 : < Formaliser le modèle de produit, Formaliser le modèle de produit, Stratégie de d'agrégation >
C_8 : < Formaliser le modèle de produit, Formaliser le modèle de produit, Stratégie de généralisation >
C_9 : < Formaliser le modèle de produit, Formaliser le modèle de processus, Stratégie d'affinement >
C_{10} : < Formaliser le modèle de processus, Formaliser le modèle de produit, Stratégie d'affinement >
C_{11} : < Identifier les éléments de la méthode, Formaliser le modèle de processus, Par méta-modélisation de processus >
C_{12} : < Formaliser le modèle de processus, Formaliser le modèle de processus, Stratégie de composition >
C_{13} : < Formaliser le modèle de processus, Formaliser le modèle de processus, Stratégie de décomposition >
C_{14} : < Formaliser le modèle de produit, Arrêter, Stratégie de validation >
C_{15} : < Formaliser le modèle de processus, Arrêter, Stratégie de validation >

Tableau 1 : Les sections de la carte de rétro-ingénierie

L'intention *Identifier les éléments de la méthode* est la première à atteindre dans le processus de rétro-ingénierie. La réalisation de cette intention permet d'identifier les éléments de la méthode à formaliser à partir de sa description initiale. Trois stratégies alternatives permettent d'atteindre cette intention *Par interview* (section C1), *Par développement coopératif de cas d'étude* (section C2) ou *A partir de documentation* (section C3). Ces stratégies correspondent à trois manières différentes et complémentaires pour identifier les éléments d'une méthode. Un élément d'une méthode peut être de type produit ou de type processus et peut être également atomique ou composé (d'autres éléments de la méthode).

L'intention *Formaliser le modèle de produit* succède à l'intention *Identifier les éléments de la méthode*. Une stratégie unique permet de relier ces deux intentions : *Par méta-modélisation de produit* (section C4). Cette stratégie utilise la technique de méta-modélisation pour formaliser le modèle de produit de la méthode. Les stratégies de *généralisation*, *spécialisation* et *agrégation* sont utilisées pour compléter et affiner le modèle de produit en cours de construction. La *Stratégie d'association* est utilisée pour relier les différents éléments de ce modèle.

La réalisation de l'intention *Formaliser le modèle de processus* permet de construire le modèle de processus de la méthode à formaliser. Quatre stratégies permettent d'atteindre cette intention *Par Méta-modélisation de processus* (section C11), par la *Stratégie de composition* (section C12), la *Stratégie de décomposition* (section C13) et enfin la *Stratégie d'affinement* (section C9).

L'application de la stratégie *Par méta-modélisation de processus* permet de modéliser le processus de la méthode en utilisant la technique de *méta-modélisation* et en se basant sur les éléments identifiés par la réalisation de l'intention *Identifier les éléments de la méthode*.

La *Stratégie de composition* est sélectionnée dans le cas où des éléments définis dans le modèle de processus nécessitent d'être agrégés dans un élément d'un niveau de granularité plus élevé dans le but d'améliorer la lisibilité du modèle.

La *Stratégie de décomposition* est sélectionnée dans le cas où des éléments définis dans le modèle de processus possèdent un niveau de granularité trop élevé et nécessitent d'être décomposés en éléments plus fins pour pouvoir guider l'ingénieur d'application dans la construction du produit souhaité par l'application de la méthode.

La *Stratégie d'affinement* (section C9) est sélectionnée dans le cas où un affinement de certains éléments du modèle de processus en cours de construction est nécessaire pour tenir compte de la structure du modèle de produit. D'une manière similaire, la confrontation du modèle de produit au modèle de processus en cours de construction peut suggérer d'appliquer la *Stratégie d'affinement* (section C10) pour affiner certaines parties de produit.

Deux stratégies permettent d'atteindre l'intention *Arrêter* pour terminer le processus de rétro-ingénierie: (i) La *Stratégie de validation* (section C14) partant de l'intention *Formaliser le modèle de produit* et (ii) la Stratégie de validation (section C15) partant de l'intention *Formaliser le modèle de processus*. Ces deux stratégies similaires permettent de valider la méthode obtenue par rétro-ingénierie en appliquant des règles de vérification des modèles.

Les deux intentions *Formaliser le modèle de produit* et *Formaliser le modèle de processus* sont parallèles dans la carte de rétro-ingénierie; ceci reflète le fait que la formalisation des modèles de produit et celle des modèles de processus se font conjointement.

Plusieurs chemins permettent de progresser de l'intention *Démarrer* à l'intention *Arrêter* dans la carte de rétro-ingénierie. Durant l'exécution de cette carte la sélection d'un chemin se fait de manière dynamique et contextuelle : à tout moment l'ingénieur de méthodes peut décider quelle intention réaliser et quelle stratégie utiliser selon la situation d'ingénierie rencontrée.

Un guidage est offert à l'ingénieur de méthodes pour cette prise de décision avec les directives de progression de la carte (Directives de Sélection d'Intention et Directives de Sélection de Stratégie). La présentation des DSI et DSS de la carte de rétro-ingénierie fait l'objet des deux prochaines sections de ce chapitre.

3.1.1. Directives de Sélection d'Intention associées à la carte de rétro-ingénierie

Afin de guider l'ingénieur de méthodes dans la progression dans la carte de rétro-ingénierie, les directives de sélection d'intention permettent d'aider à choisir, pour une intention déjà réalisée, les intentions qu'on peut sélectionner dans l'étape suivante du processus.

Trois Directives de Sélection d'Intention (DSI) sont associées à la carte de rétro-ingénierie. Le Tableau 2 présente ces DSI.

Intention	DSI
I_1 : Démarrer	N/A
I_2 : Identifier les éléments de la méthode	DSI_1 : < ({élément 'identifié'}), Progresser depuis Identifier les éléments de la méthode >
I_3 : Formaliser le modèle de produit	DSI_2 : < (MPdt 'en construction', MPrs 'en construction'), Progresser depuis Formaliser le modèle de produit >
I_4 : Formaliser le modèle de processus	DSI_3 : < (MPdt 'en construction', MPrs 'en construction'), Progresser depuis Formaliser le modèle de processus>
I_5 : Arrêter	N/A

Tableau 2. Les DSI de la carte de rétro-ingénierie

3.1.1.1. Progresser depuis « Identifier les éléments de la méthode » (DSI_1)

La Figure 24 présente la structure de la DSI_1. Cette directive est associée à l'intention *Identifier les éléments de la méthode*. Elle aide l'ingénieur de méthodes à progresser dans la carte après avoir réalisé cette intention. Comme le montre la Figure 24 l'ingénieur de méthodes a deux possibilités : atteindre l'intention *Formaliser le modèle de produit* en sélectionnant la DRI_4 ou atteindre l'intention *Formaliser le modèle de processus* en sélectionnant la DRI_{11}. Les arguments en faveur de chacun des choix sont présentés en bas de la figure.

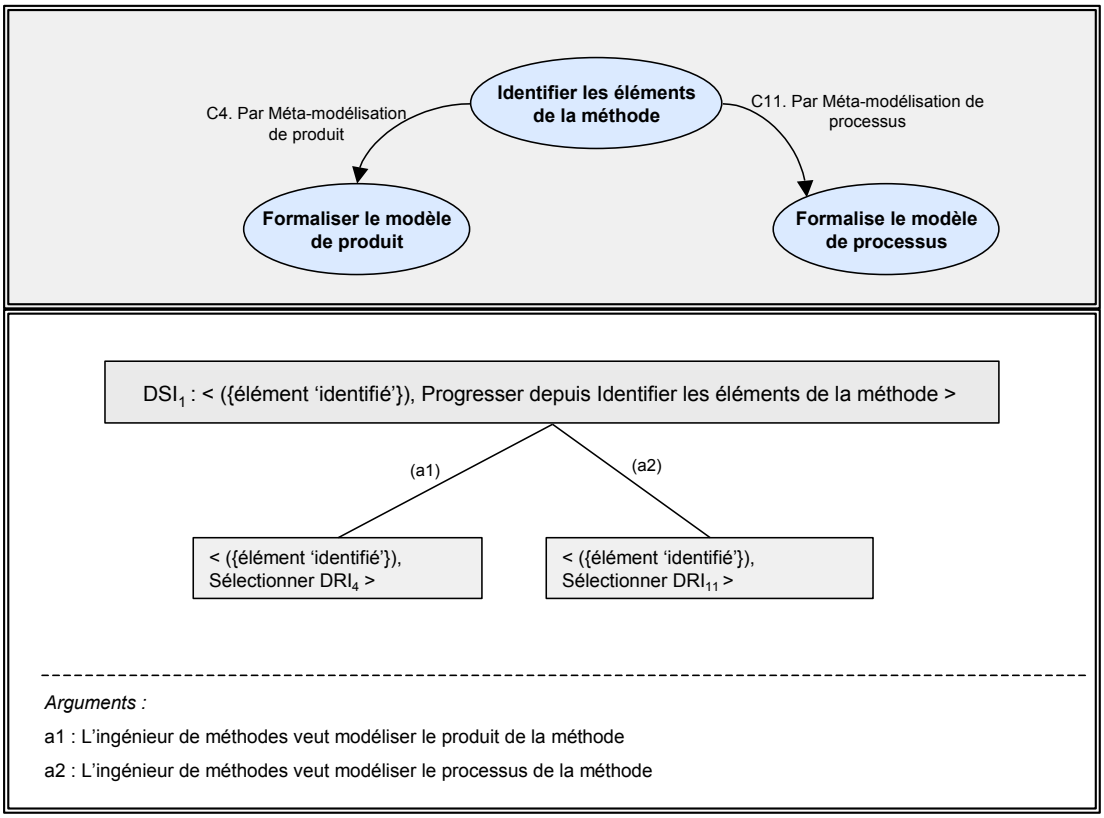


Figure 24. Structure de la DSI_1

3.1.1.2. Progresser depuis « Formaliser le modèle de produit » (DSI_2)

La DSI_2 est de type choix, elle permet de progresser à partir de l'intention *Formaliser le modèle de produit* en proposant d'atteindre une intention cible. Elle propose de réaliser soit l'intention *Formaliser le modèle de produit* (boucle), soit l'intention *Formaliser le modèle de processus*, soit l'intention *Arrêter*. Comme le montre la Figure 25, pour réaliser le premier choix, on sélectionne la DSS_2 qui propose quatre stratégies (*Stratégie d'association*, *Stratégie de spécialisation, Stratégie d'agrégation* et *Stratégie de généralisation*). Pour faire le deuxième choix, on sélectionne la DRI_9 qui correspond à la *Stratégie d'affinement* pour modéliser un élément de processus. Le troisième choix est réalisé, quant à lui, en sélectionnant la DRI_{14} correspondant à la *Stratégie de validation* pour arrêter le processus de rétro-ingénierie.

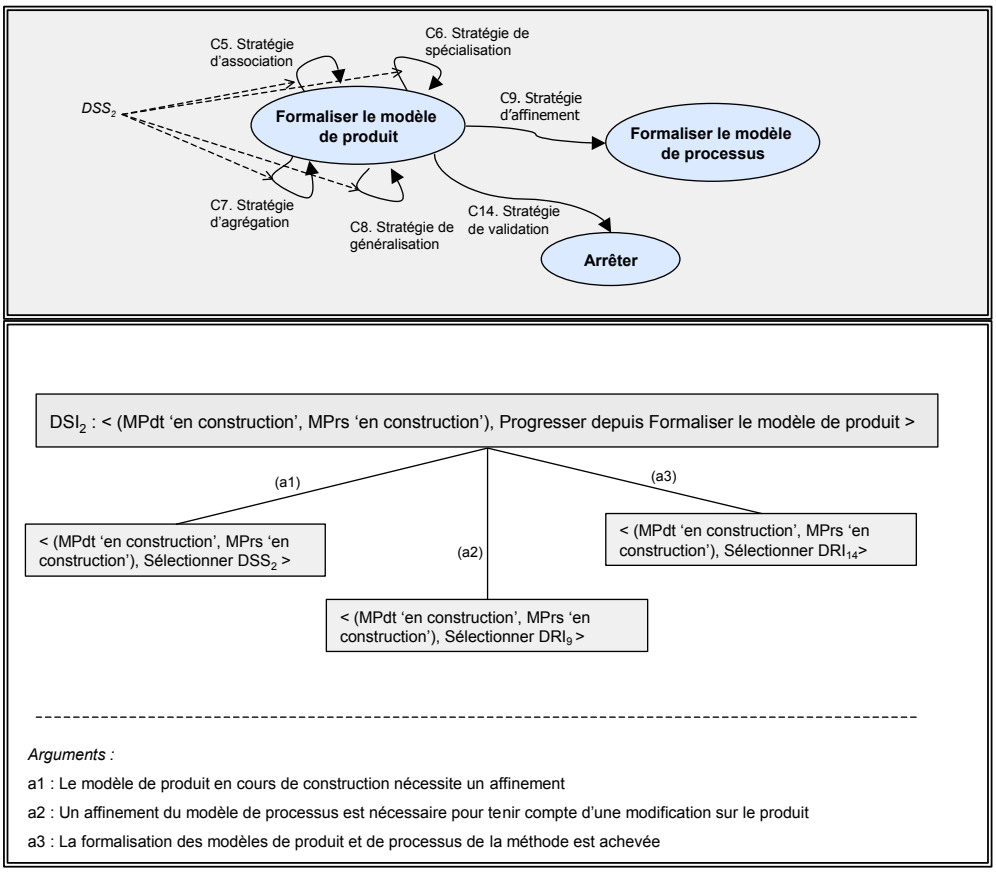


Figure 25. Structure de la DSI_2

3.1.1.3. Progresser depuis « Formaliser le modèle de processus » (DSI_3)

La Figure 26 présente la structure de la DSI_3 qui est une directive tactique de type choix. Elle guide l'ingénieur de méthodes à progresser dans la carte depuis l'intention *Formaliser le modèle de processus*. Cette directive offre trois possibilités à l'ingénieur de méthodes :

- Atteindre l'intention *Formaliser le modèle de processus* (boucle) en sélectionnant la DSS_3 qui propose deux stratégies (*Stratégie de composition* et *Stratégie de décomposition*),
- Atteindre l'intention *Formaliser le modèle de produit* en sélectionnant la DRI_{10},
- Atteindre l'intention *Arrêter* en sélectionnant la DRI_{15}.

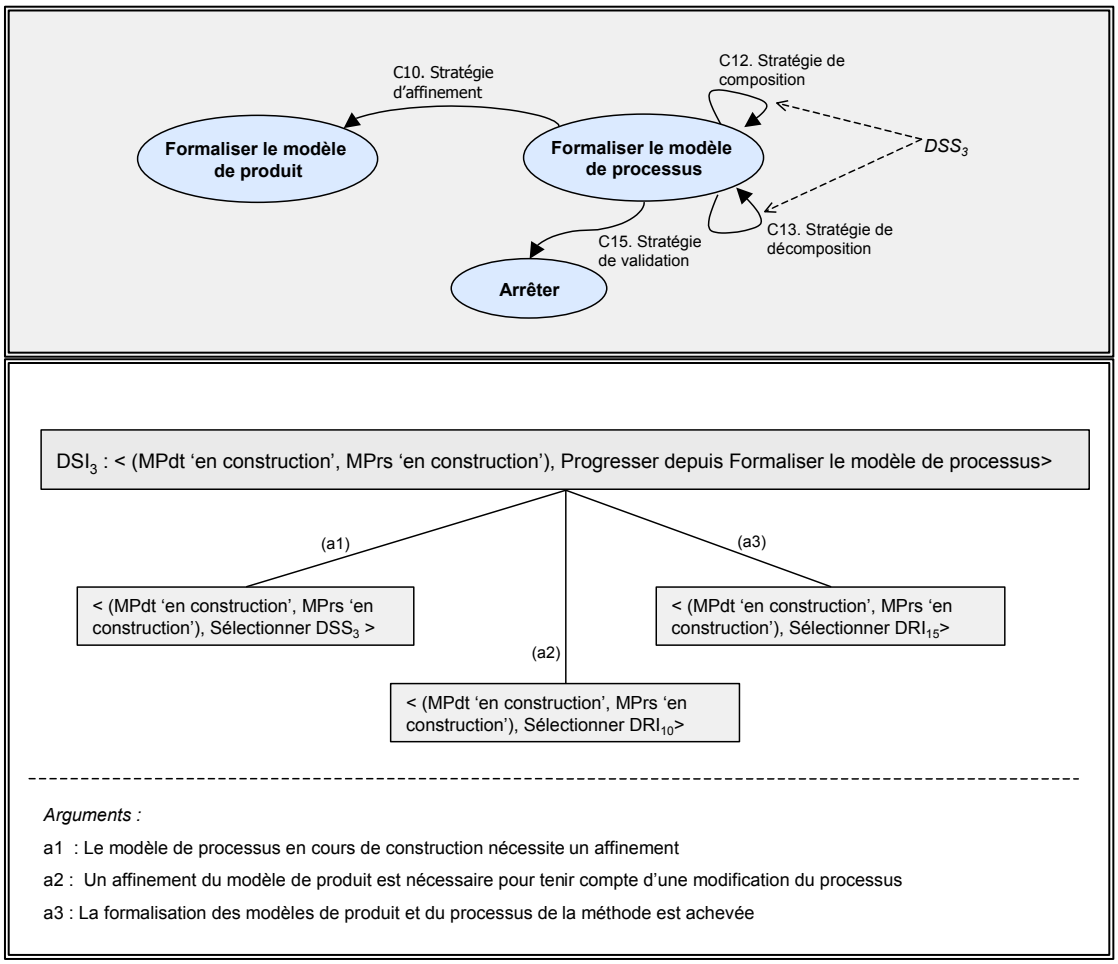


Figure 26. Structure de la DSI_3

3.1.2. Directives de Sélection de Stratégie associées à la carte de rétro-ingénierie

Comme nous l'avons défini au chapitre 3, nous associons une Directive de Sélection de Stratégie (DSS) pour tout ensemble de sections parallèles dans une carte. Trois DSS sont alors associées à la carte de rétro-ingénierie (Tableau 3).

Paire d'intentions	DSS
(I_1 - I_2) : (Démarrer, Identifier les éléments de la méthode)	DSS_1 : < (DIM), Progresser vers Identifier les éléments de la méthode>
(I_3 – I_3) : (Formaliser le modèle de produit, Formaliser le modèle de produit)	DSS_2 : < (MPdt 'en construction', MPrs 'en construction'), Progresser vers Formaliser le modèle de produit >
(I_4 – I_4) : (Formaliser le modèle de processus, Formaliser le modèle de processus)	DSS_3 : <(MPdt 'en construction', MPrs 'en construction'), Progresser vers Formaliser le modèle de processus>

Tableau 3. Les DSS de la carte de rétro-ingénierie

3.1.2.1. Progresser vers « Identifier les éléments de la méthode » (DSS_1)

La DSS_1 aide l'ingénieur de méthodes à démarrer le processus de rétro-ingénierie en lui proposant trois stratégies alternatives *Par interview, A partir de documentation* et *Par développement coopératif de cas d'étude*. La Figure 27 présente la DSS_1 qui est une directive tactique de type choix offrant trois possibilités :

(1) Sélectionner la directive DRI_1 pour appliquer la stratégie *Par interview.*

(2) Sélectionner la directive DRI_2 pour appliquer la stratégie *A partir de documentation.*

(3) Sélectionner la directive DRI_3 pour appliquer la stratégie *Par développement coopératif de cas d'étude*.

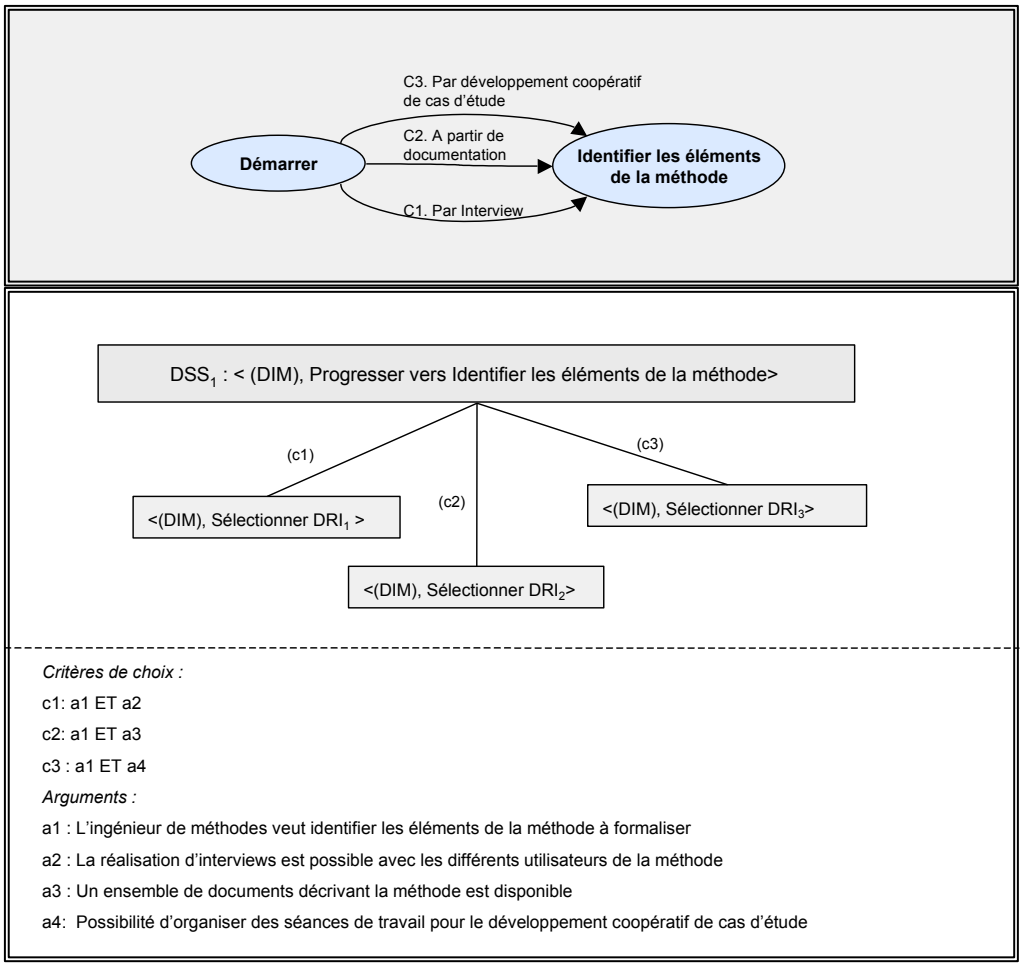


Figure 27. Structure de la DSS_1

3.1.2.2. Progresser vers « Formaliser le modèle de produit » (DSS_2)

La DSS_2 est de type choix, elle permet de progresser à partir de l'intention *Formaliser le modèle de produit* vers l'intention *Formaliser le modèle de produit*, c'est-à-dire qu'elle permet de boucler sur l'intention elle-même. Elle offre à l'ingénieur de méthodes quatre possibilités de progression dans la

carte : sélectionner la directive DRI_5 correspondante à la *Stratégie d'association*, sélectionner la directive DRI_6 associée à la *Stratégie de spécialisation*, sélectionner la DRI_7 associée à la *Stratégie d'agrégation* ou sélectionner la DRI_8 associée à la *Stratégie de généralisation*. La Figure 28 montre la structure de cette DSS ainsi que les arguments pour le choix d'une stratégie plutôt qu'une autre.

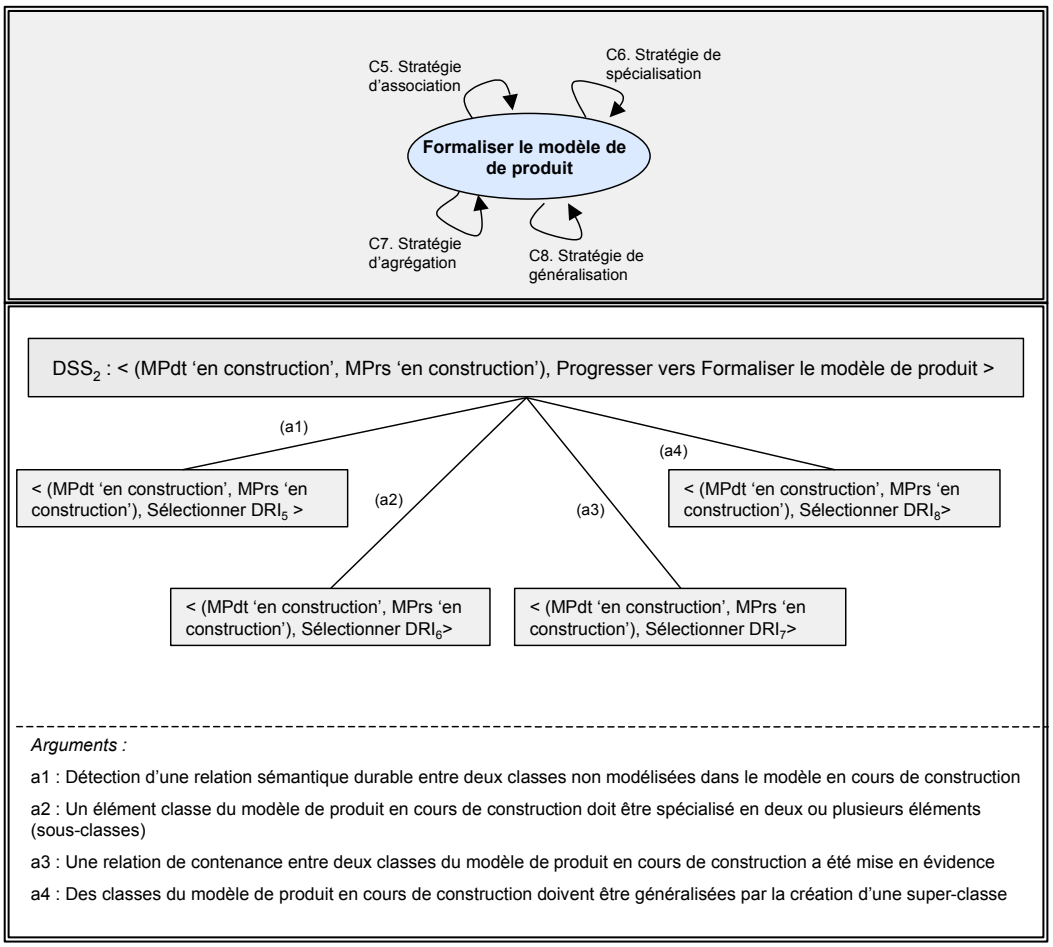


Figure 28. Structure de la DSS_2

3.1.2.3. Progresser vers « Formaliser le modèle de processus » (DSS_3)

La DSS_3 est de type choix, elle offre un guidage à l'ingénieur de méthodes pour progresser depuis l'intention *Formaliser le modèle de processus* vers l'intention *Formaliser le modèle de processus* (boucle). Cette directive offre à l'ingénieur de méthodes deux possibilités :

- Appliquer la *Stratégie de composition* en sélectionnant la DRI_{12}
- Appliquer la *Stratégie de décomposition* en sélectionnant la DRI_{13}

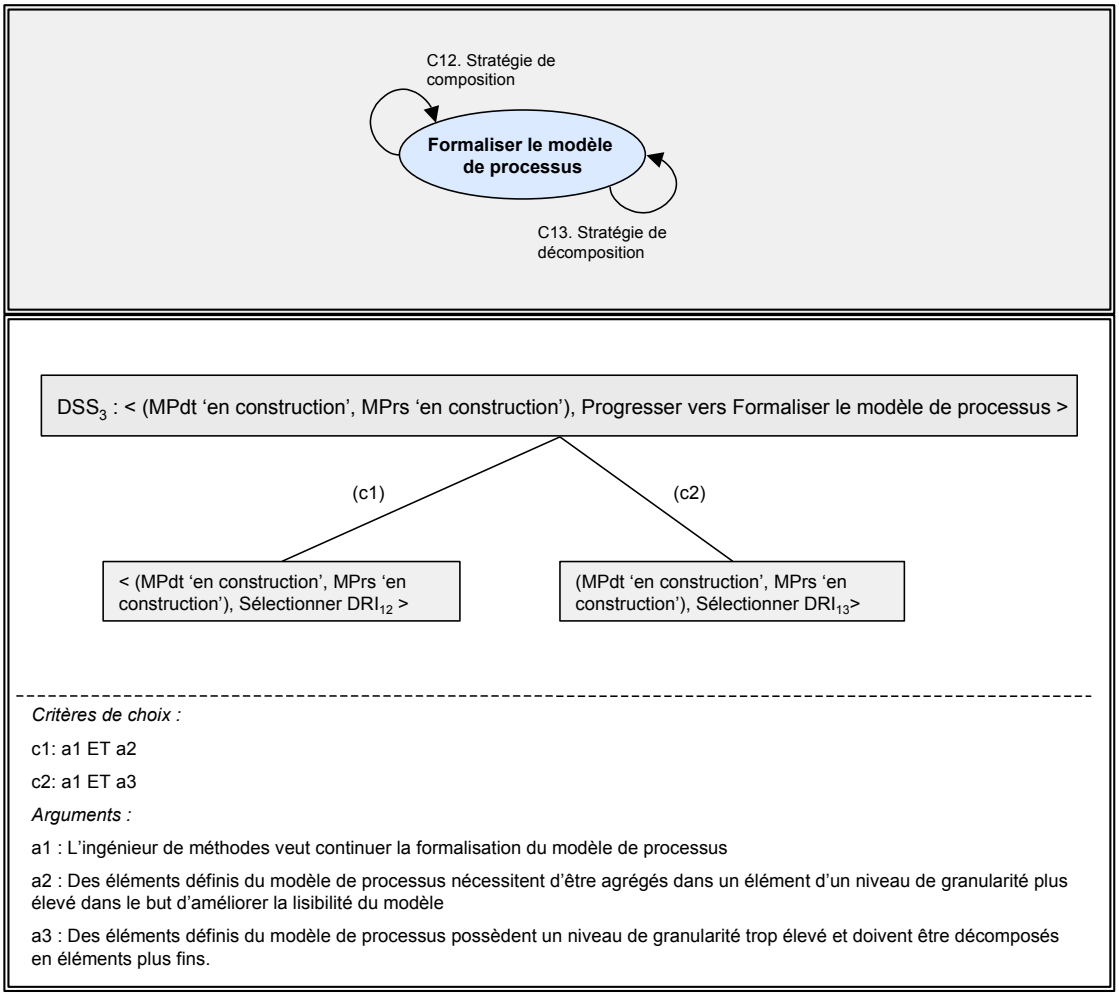


Figure 29. Structure de la DSS_3

3.1.3. Tableau des DRI associées à la carte de rétro-ingénierie.

La carte de rétro-ingénierie comporte 15 sections, à chacune d'entre elles on associe une Directive de Réalisation d'Intention (DRI), le Tableau 4 présente ces DRI.

Identifiant	Signature de DRI	Type
DRI_1	< (DIM), Identifier les éléments de la méthode Par Interview >	Tactique
DRI_2	< (DIM), Identifier les éléments de la méthode A Partir de documentation >	Tactique
DRI_3	< (DIM), Identifier les éléments de la méthode Par développement coopératif de cas d'étude >	Tactique
DRI_4	< ({élément 'identifié'), Formaliser le modèle de produit par méta-modélisation de produit >	Tactique
DRI_5	< (MPdt 'en construction', MPrs 'en construction'), Formaliser le modèle de produit avec la stratégie d'association >	Tactique
DRI_6	< (MPdt 'en construction', MPrs 'en construction'), Formaliser le modèle de produit avec la stratégie de spécialisation >	Tactique
DRI_7	< (MPdt 'en construction', MPrs 'en construction'), Formaliser le modèle de produit avec la stratégie d'agrégation >	Tactique

DRI_8	< (MPdt 'en construction', MPrs 'en construction'), Formaliser le modèle de produit avec la stratégie de généralisation >	Tactique
DRI_9	< (MPdt 'en construction', MPrs 'en construction'), Formaliser le modèle de processus avec la stratégie d'affinement>	Tactique
DRI_{10}	< (MPdt 'en construction', MPrs 'en construction'), Formaliser le modèle de produit avec la stratégie d'affinement>	Tactique
DRI_{11}	< ({élément 'identifié'}), Formaliser le modèle de processus par méta-modélisation de processus >	Tactique
DRI_{12}	< (MPdt 'en construction', MPrs 'en construction'), Formaliser le modèle de processus avec la stratégie de composition >	Tactique
DRI_{13}	< (MPdt 'en construction', MPrs 'en construction'), Formaliser le modèle de processus avec la stratégie de décomposition >	Tactique
DRI_{14}	< (MPdt 'défini', MPrs 'défini'), Arrêter avec la Stratégie de validation >	Tactique
DRI_{15}	< (MPdt 'défini', MPrs 'défini'), Arrêter avec la Stratégie de validation >	Tactique

Tableau 4. Les DRI de la carte de rétro-ingénierie

3.2. Sections de la carte de rétro-ingénierie

3.2.1. Réaliser « Identifier les éléments de la méthode Par Interview » (DRI_1)

La DRI_1 est associée à la section C1 de la carte de rétro-ingénierie. L'objectif de cette directive sert à identifier les éléments de la méthode à formaliser en utilisant la technique d'interview. La Figure 30 présente la structure de la DRI_1 qui est une directive tactique de type plan.

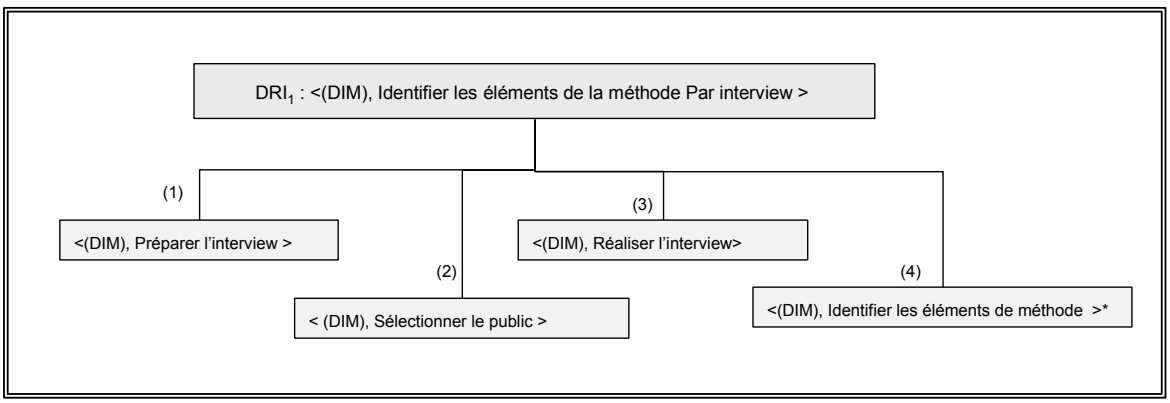


Figure 30. Structure de la DRI_1

Comme le montre cette figure, ce plan propose à l'ingénieur de méthodes de :

(1) *Préparer l'interview :* Cette directive suggère à l'ingénieur de méthodes de préparer une liste préliminaire de questions à poser durant l'interview. La définition de ces questions dépend de la situation d'ingénierie de la méthode; elles doivent porter sur les objectifs souhaités par son application, la nature du produit qu'elle construit et la démarche suivie pour construire ce produit.

(2) *Sélectionner le public :* Le but de cette directive est de sélectionner le public avec lequel on va réaliser l'interview, l'ingénieur de méthodes doit veiller à ce que les personnes sélectionnées aient une bonne connaissance de la méthode à formaliser.

(3) *Réaliser l'interview :* Cette directive suggère à l'ingénieur de méthodes de réaliser l'interview préparée avec le public sélectionné.

(4) *Identifier les éléments de méthode :* L'objectif de cette directive est de réaliser une synthèse des résultats de l'interview pour identifier les éléments de la méthode. Les éléments identifiés peuvent être de type produit ou processus.

3.2.2. Réaliser « Identifier les éléments de la méthode A partir de documentation » (DRI_2)

La DRI_2 est associée à la section C2 de la carte de rétro-ingénierie. L'objectif de cette directive est d'identifier les éléments de la méthode par lecture de la documentation disponible de cette méthode. La DRI_2 est une directive tactique de type plan. La Figure 31 présente cette directive.

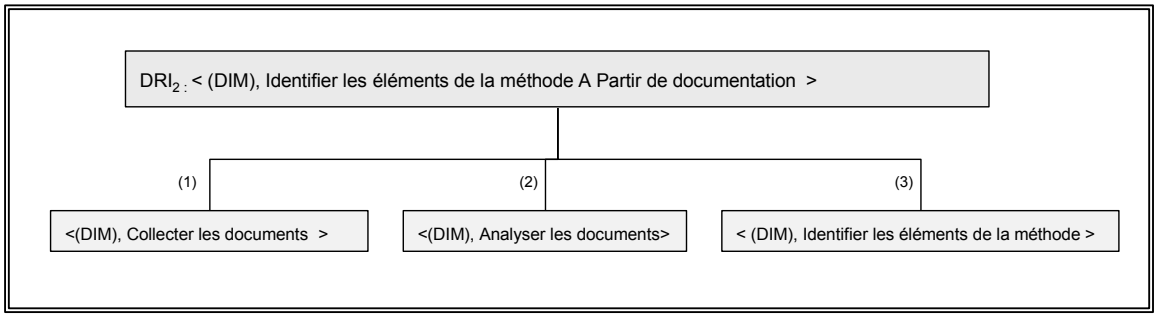


Figure 31. Structure de la DRI_2

(1) < *(DIM), Collecter les documents* > : Cette directive suggère à l'ingénieur de méthodes collecter tous les documents décrivant la méthode et pouvant servir à sa connaissance. Ces documents peuvent être (i) des documents dédiés à une présentation plus ou moins structurée de la méthode ou (ii) des comptes rendus d'utilisation de la méthode dans divers projets de développement de SI.

(2) < *(DIM), Analyser les documents*> : Cette directive suggère à l'ingénieur de méthodes d'effectuer une lecture des documents collectés par l'application de la directive précédente. Ceci permet d'avoir une première vision de la méthode à formaliser.

(3) < *(DIM), Identifier les éléments de la méthode* > : L'objectif de cette directive est d'identifier les éléments de la méthode à formaliser sur la base des lectures effectuées. Pour réaliser cette tâche l'ingénieur de méthodes est invité à différencier les éléments décrivant la méthode des instances utilisées pour présenter celle-ci.

3.2.3. Réaliser « Identifier les éléments de méthode Par développement coopératif de cas d'étude » (DRI_3)

La DRI_3 est associée à la section C3 de la carte de rétro-ingénierie. Comme les deux précédentes l'objectif de cette directive est d'identifier les éléments de la méthode à formaliser. La stratégie *Par développement coopératif de cas d'étude* est utilisée pour atteindre cet objectif.

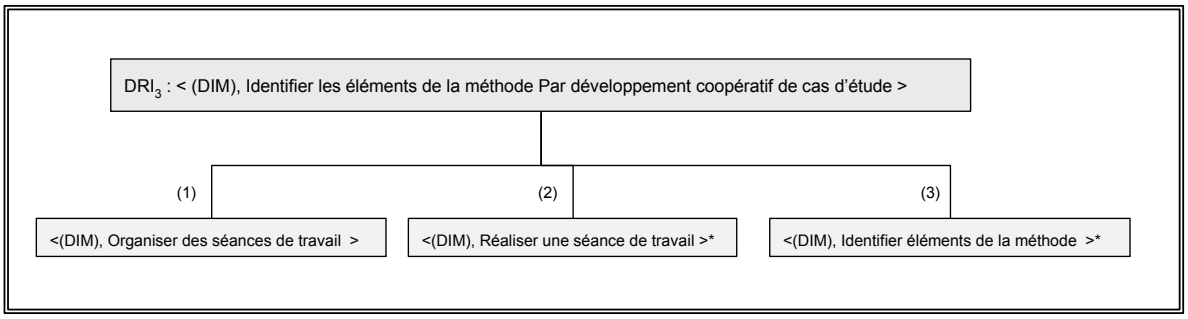


Figure 32. Structure de la DRI3

La Figure 32 présente la DRI_3 qui est une directive tactique de type plan, ce plan est décomposé comme suit :

(1) *<(DIM), Organiser des séances de travail >* : Des séances de travail sont organisées pour développer un cas d'étude en utilisant la méthode. Ce développement est réalisé conjointement entre l'équipe d'ingénierie de méthodes et les experts de la méthode. La Figure 33 présente cette directive de type plan, ce plan propose de (1.1) Rédiger différents cas d'étude avec une complexité croissante (1.2) Sélectionner les participants parmi l'équipe de l'ingénierie de méthode et les experts de la méthode et (1.3) convoquer les participants à l'endroit et l'horaire fixés.

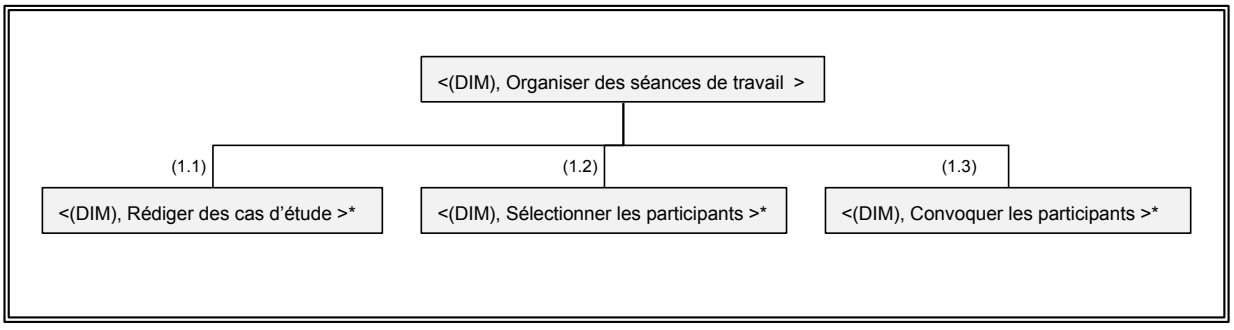


Figure 33. < (DIM), Organiser des séances de travail >

(2) *<(DIM), Réaliser une séance de travail >* : Cette directive guide l'ingénieur de méthodes durant le déroulement d'une séance de travail, elle lui suggère de documenter les différentes étapes de développement de cas d'étude et les différentes discussions et réflexions effectuées.

(3)<(DIM), Identifier éléments de la méthode> : Sur la base de la connaissance acquise de l'utilisation de la méthode durant le développement coopératif de cas d'étude, l'ingénieur de

méthodes est invité par cette directive à identifier les éléments de produit et les éléments de processus décrivant la méthode à formaliser.

3.2.4. Réaliser « Formaliser le modèle de produit Par méta-modélisation de produit» (DRI_4)

La DRI_4 est associée à la section C4 de la carte de rétro-ingénierie. L'objectif de cette directive est de formaliser le modèle de produit de la méthode en se basant sur les éléments identifiés en réalisant l'intention *Identifier les éléments de la méthode*. Cette directive utilise la technique de méta-modélisation en respectant le méta-modèle de produit défini au chapitre 3. Dans ce méta-modèle nous avons défini deux manières orthogonales pour classer un *élément de produit*. La première classification fait la distinction entre *élément de produit composé* et *élément de produit atomique*. La deuxième classification différencie les éléments *Lien* et *Non-Lien*. Dans cette directive nous désignons par *élément* de produit une *classe* du modèle de produit ou un *lien* (d'association, d'agrégation ou d'héritage) entre les classes du modèle.

La Figure 34 présente la DRI_4 qui est une directive tactique de type plan.

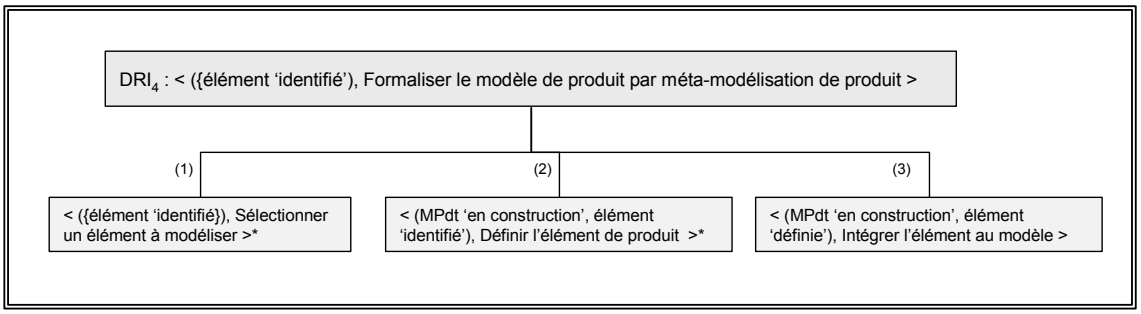


Figure 34. Structure de la DRI_4

(1) < *({élément 'identifié}), Sélectionner un élément à modéliser* > : Cette directive suggère à l'ingénieur de méthodes de sélectionner l'élément à modéliser parmi l'ensemble des éléments identifiés par la réalisation de l'intention *Identifier les éléments de la méthode*.

(2) < (*MPdt 'en construction', élément 'identifié'), Définir l'élément de produit*> : L'application de cette directive permet de définir l'élément de produit sélectionné par la directive précédente. Cette directive est de type plan, elle est présentée à la Figure 35. Ce plan propose de (2.1) créer une classe pour modéliser l'élément identifié, (2.2) définir le nom de la classe et (2.3) définir ses attributs.

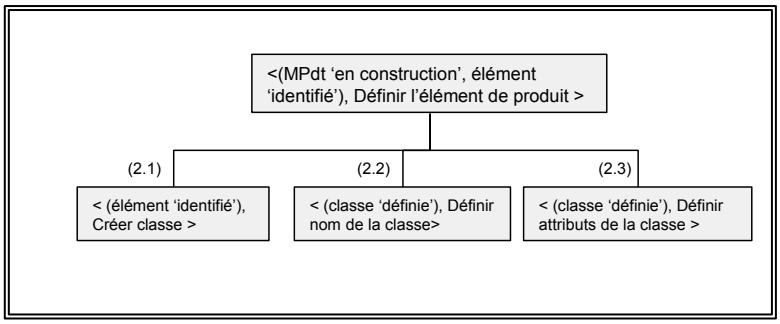


Figure 35. <(MPdt 'en construction', élément 'identifié'), Définir un élément de produit >

(3) < *(MPdt 'en construction', élément défini), Intégrer un élément au modèle* >: Une fois l'élément de produit défini, cette directive invite l'ingénieur de méthodes à l'intégrer dans le modèle de produit en cours de construction. Cette intégration passe par l'ajout d'éventuels liens d'association, d'agrégation ou d'héritage entre le nouvel élément et les éléments existants dans ce modèle.

3.2.5. Réaliser « Formaliser le modèle de produit avec la Stratégie d'association » (DRI_5)

La DRI_5 est associée à la *stratégie d'association* pour atteindre l'intention *Formaliser le modèle de produit*. L'objectif de cette directive est d'assister l'ingénieur de méthodes pour compléter le modèle de produit en cours de construction en ajoutant des liens d'association entre les éléments déjà définis dans le modèle. La Figure 36 présente la DRI_5 qui est une directive tactique de type plan. Ce plan se décrit comme suit :

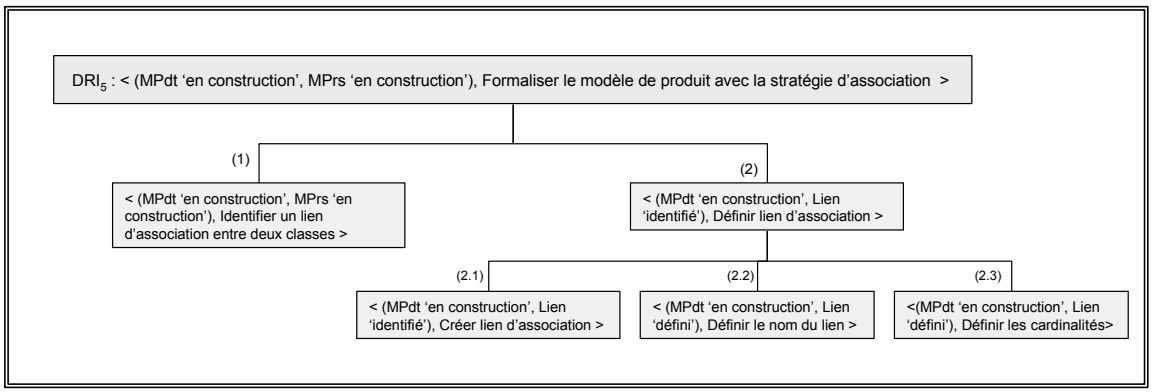


Figure 36. Structure de la DRI_5

(1) *Identifier un lien d'association entre deux classes* : Le but de cette directive est d'analyser le modèle de produit en cours de construction dans le but d'identifier d'éventuelles relations sémantiques durables entre deux classes non modélisées par un lien.

(2) *Définir un lien d'association* : L'application de cette directive permet de définir le lien d'association défini par l'application de la directive précédente. Cette directive est de type plan, et

propose de : (2.1) Créer le lien d'association, (2.2) Définir le nom du lien et (2.3) Définir les cardinalités associées au lien.

3.2.6. Réaliser « Formaliser le modèle de produit avec la Stratégie de Spécialisation » (DRI_6)

La DRI_6 est associée à la section C6 de la carte de rétro-ingénierie. L'objectif de cette directive est d'analyser le modèle de produit en cours de construction dans le but d'identifier un élément de ce modèle qui nécessite d'être spécialisé (i.e. une classe qui doit être spécialisée en deux ou plusieurs sous-classes).

La Figure 37 présente la DRI_6 qui est une directive tactique de type plan. Ce plan propose à l'ingénieur de méthodes de (1) identifier la super-classe qui nécessite d'être spécialisée, (2) définir les sous-classes et (3) créer le lien *Est-un* entre la super-classe et chaque sous-classe.

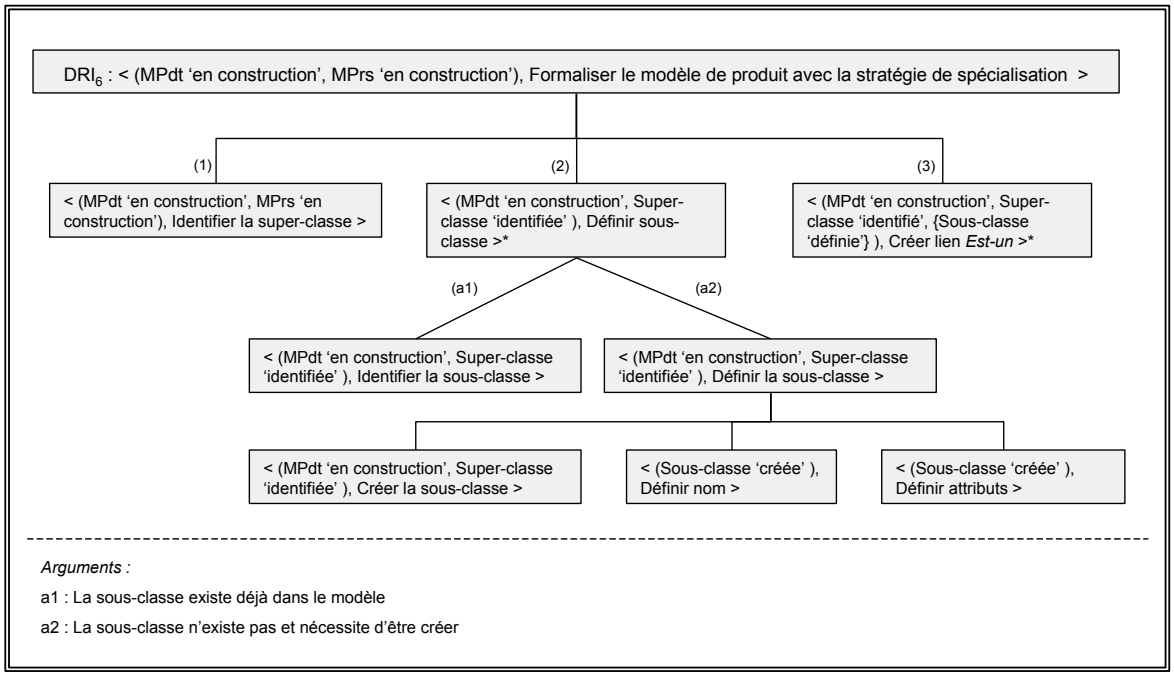


Figure 37. Structure de la DRI_6

Comme le montre cette figure, la DRI6 tient compte de deux cas de figure pour définir une sous-classe : (i) identifier une sous-classe existante dans le modèle, (ii) Créer une sous-classe spécialisant la super-classe.

3.2.7. Réaliser « Formaliser le modèle de produit avec la Stratégie d'agrégation » (DRI_7)

La DRI_7 est associée à la section C7 de la carte de rétro-ingénierie. Cette directive a pour objectif de compléter la construction du modèle de produit en ajoutant les liens d'agrégation entre les éléments définis dans ce modèle.

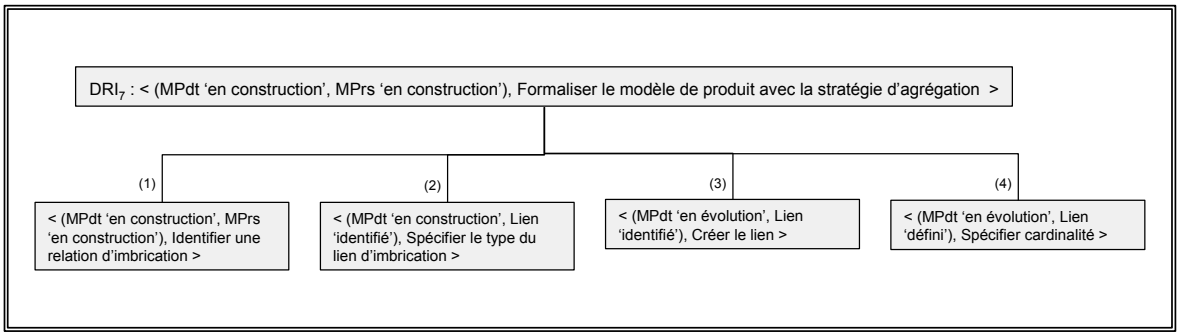


Figure 38. Structure de la DRI_7

La Figure 38 présente la DRI_7 qui est une directive tactique de type plan proposant de :

(1) *Identifier une relation d'imbrication* : Cette directive exécutable suggère à l'ingénieur de méthodes d'analyser le modèle de produit en cours de construction dans le but d'identifier d'éventuelles relations d'imbrication entre les éléments de ce modèle.

(2) *Spécifier le type du lien d'imbrication* : L'application de cette directive exécutable permet de spécifier le type du lien d'imbrication : *Composition* ou *Agrégation*. Une composition étant une agrégation plus forte impliquant que : (i) un élément ne peut appartenir qu'à un seul agrégat composite et (ii) la destruction de l'agrégat composite entraîne la destruction de tous ses éléments.

(3) *Créer le lien* : L'application de cette directive permet de créer une instance du lien d'agrégation ou de composition entre l'élément composite et l'élément composant.

(2) Spécifier cardinalité : Cette directive suggère à l'ingénieur de méthodes de spécifier les cardinalités associées au lien créé.

3.2.8. Réaliser « Formaliser le modèle de produit avec la Stratégie de généralisation » (DRI_8)

La DRI_8 est associée à la section C8 de la carte de rétro-ingénierie. L'objectif de cette directive est de compléter le modèle de produit en cours d'évolution en traitant les cas de généralisation (i.e. une ou

plusieurs classes du modèle ayant des caractéristiques communes et pouvant être représentés par une même entité générique)

La Figure 39 présente la DRI_8 qui est une directive tactique de type plan. Ce plan propose de (1) identifier les sous-classes (des classes représentant une même entité générique et ayant des attributs en commun), (2) définir la super-classe généralisant les sous-classes identifiées et (3) créer le lien d'héritage *Est-un* entre la super-classe et chacune des sous-classes.

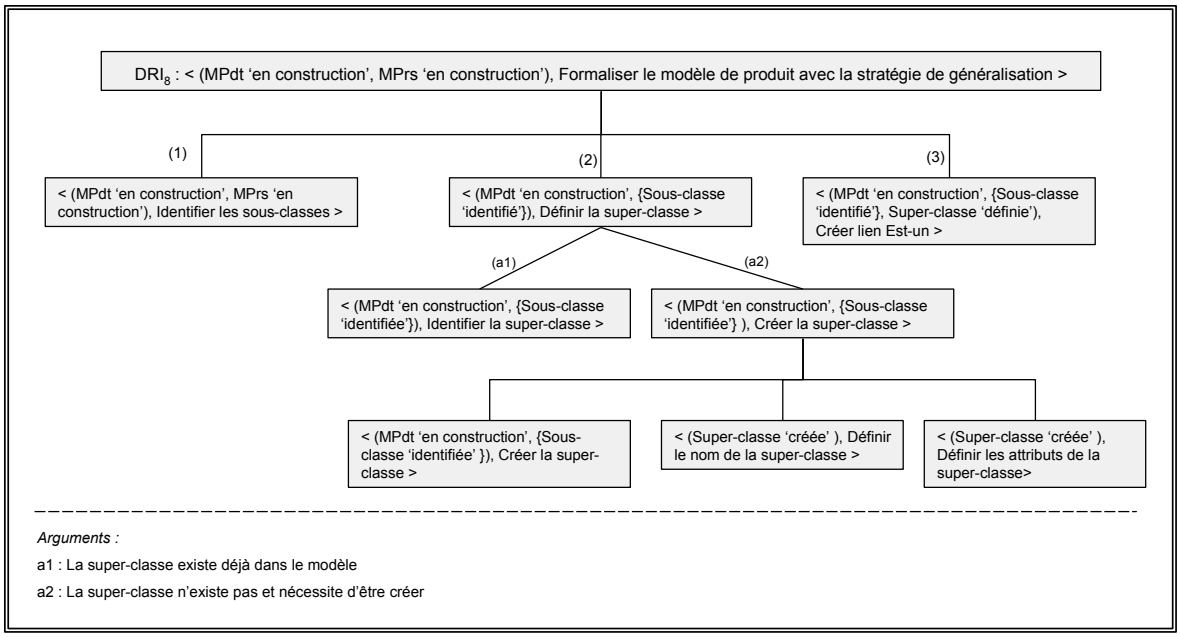


Figure 39. Structure de la DRI_8

3.2.9. Réaliser « Formaliser le modèle de processus avec la Stratégie d'affinement » (DRI_9)

La DRI_9 est associée à la section C9 de la carte de rétro-ingénierie, l'objectif de cette directive est d'affiner certains éléments du modèle de processus en cours de construction pour tenir compte de la structure de la partie de produit qu'ils permettent de transformer.

La DRI_9 est une directive tactique de type plan, elle est présentée à la Figure 40. Ce plan propose de (1) Identifier un élément de processus à transformer par la confrontation du modèle de processus et du modèle de produit, (2) Affiner l'élément identifié et (3) Vérifier la cohérence du modèle du modèle de processus après les transformations réalisées.

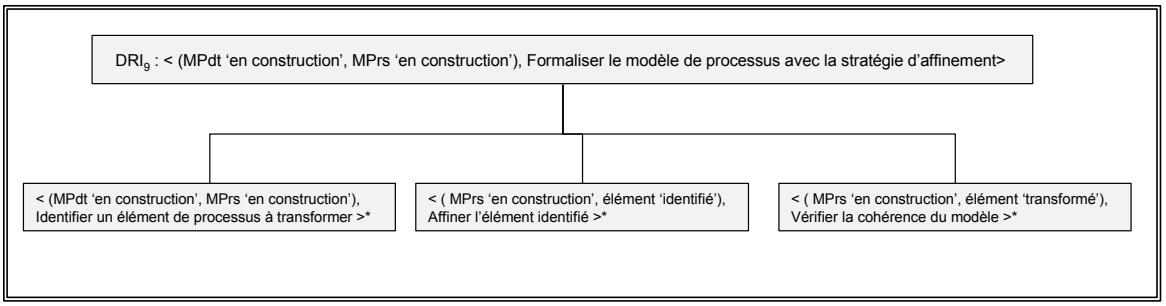


Figure 40. Structure de la DRI_9

3.2.10. Réaliser « Formaliser le modèle de produit avec la Stratégie d'affinement » (DRI_{10})

La DRI_{10} est associée à la section *<Formaliser le modèle de produit, Formaliser le modèle de processus, Par affinement>* de la carte de rétro-ingénierie. L'objectif de cette directive est de confronter le modèle de produit au modèle de processus en cours de construction dans le but d'identifier d'éventuels cas de discordance nécessitant d'être remédiés par affinement de certains éléments de produit.

La Figure 41 présente la DRI_{10} qui est une directive tactique de type plan. Cette directive est similaire à la précédente, elle propose de (1) Identifier un élément de produit à transformer par confrontation avec le processus, (2) Affiner l'élément identifié et (3) Vérifier la cohérence du modèle de produit.

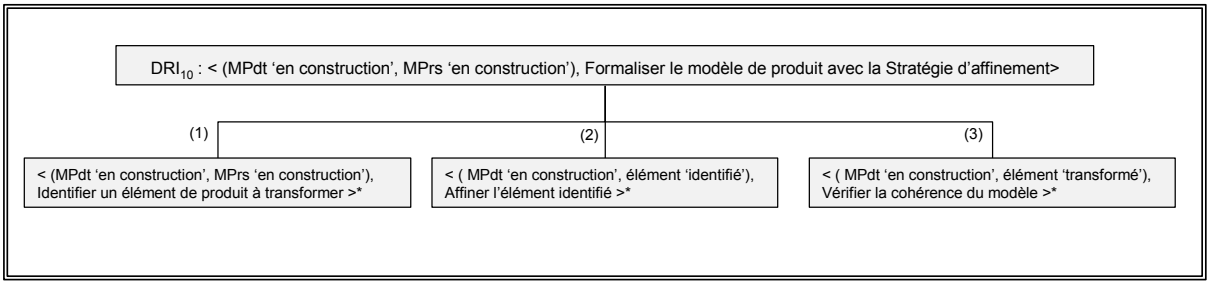


Figure 41. Structure de la DRI_{10}

3.2.11. Réaliser « Formaliser le modèle de processus Par méta-modélisation de processus » (DRI_{11})

La DRI_{11} est associée à la section C_{11} de la carte de rétro-ingénierie. L'objectif de cette directive est de guider l'ingénieur de méthodes dans la construction du modèle de processus de la méthode en modélisant les éléments de processus identifiés par la réalisation de l'intention *Identifier les éléments de la méthode*.

La DRI_{11} est une directive tactique de type plan, l'exécution de ce plan présenté à la Figure 42 permet la formalisation du modèle de processus en utilisant le paradigme *orienté-activité*, le choix de ce paradigme est motivé par sa simplicité et sa popularité.

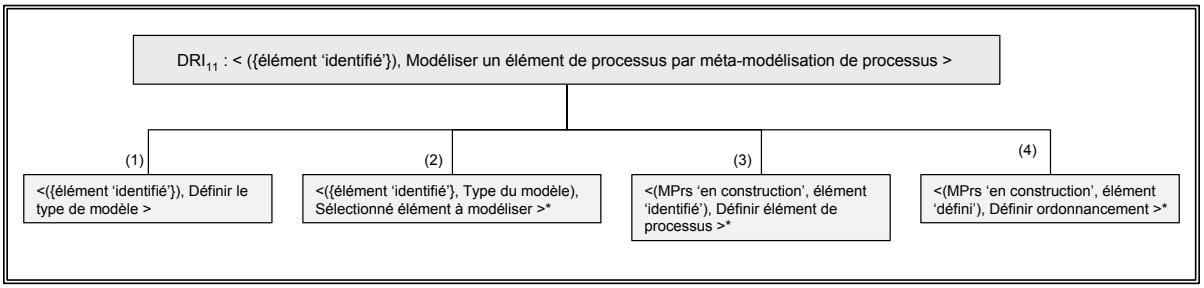


Figure 42. Structure de la DRI_{11}

(1) < *({élément 'identifié'}), Définir le type de modèle* > : Cette directive suggère à l'ingénieur de méthodes de sélectionner le type de modèle de processus adéquat à sa situation d'ingénierie. Plusieurs types de modèles proposant des manières différentes de définir l'ordonnancement des activités du modèle de processus sont disponibles parmi les quels on peut citer : le modèle en *Cascade* [Royce 70], le modèle en *Spirale* [Boehm 90], le modèle *Hiérarchique en spirale* [Iivari 90] ou le modèle en *Fontaine* [Henderson-Sellers 90].

(2) < *({élément 'identifié'}, Type du modèle), Sélectionné élément à modéliser* > : Cette directive suggère à l'ingénieur de méthodes de sélectionner un élément de processus à modéliser parmi l'ensemble des éléments identifiés par la réalisation de l'intention *Identifier les éléments de la méthode*.

(3) < *(MPrs 'en construction', élément 'identifié'), Définir un élément de processus* > : Cette directive invite l'ingénieur de méthodes à définir l'élément de processus identifié, cet élément peut être une action complexe ou une action atomique modifiant une partie de produit.

(4) < *(MPrs 'en construction', élément 'défini'), Définir ordonnancement* > : Cette directive suggère à l'ingénieur de méthodes d'intégrer l'élément défini dans le modèle de processus en cours de construction en modifiant l'ordonnancement de celui-ci en conséquence conformément au type de modèle choisi par la réalisation la première directive.

3.2.12. Réaliser « Formaliser le modèle de processus avec la Stratégie de composition » (DRI_{12})

La DRI_{12} est associée à la section C12 de la carte de rétro-ingénierie. Cette directive est sélectionnée dans le cas ou des éléments définis dans le modèle de processus nécessitent

d'être agrégés dans un élément d'un niveau de granularité plus élevé dans le but d'améliorer la visibilité du modèle.

La Figure 43 présente la DRI_{12} qui est une directive tactique de type plan.

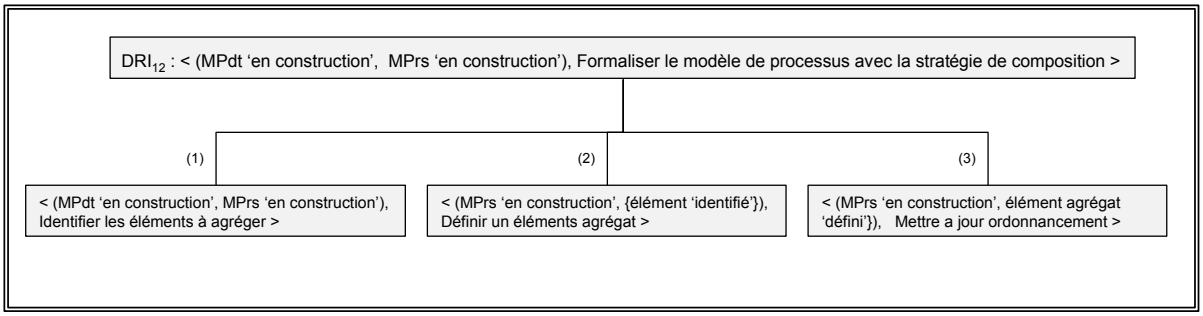


Figure 43. Structure de la DRI_{12}

(1) < *(MPdt 'en construction', MPrs 'en construction'), Identifier les éléments à agréger* > : Cette directive suggère à l'ingénieur de méthodes d'analyser le modèle de processus en cours de construction dans le but d'identifier des éléments atomiques ayant un objectif commun et nécessitant d'être rassemblés dans un élément agrégat.

(2) < *(MPrs 'en construction', {élément 'identifié'}), Définir un élément agrégat* > : Cette directive invite l'ingénieur de méthodes à définir un élément agrégat pouvant contenir les éléments identifiés par la directive précédente.

(3) < *(MPrs 'en construction', élément agrégat 'défini'), Mettre à jour ordonnancement* > : Cette directive suggère à l'ingénieur de méthodes de mettre à jour l'ordonnancement des éléments du modèle de processus en cours de construction pour inclure l'élément agrégat défini par la directive précédente.

3.2.13. Réaliser « Formaliser le modèle de processus avec la Stratégie de décomposition » (DRI_{13})

La DRI_{13} est associée à la section <*Formaliser le modèle de processus, Formaliser le modèle de processus, Stratégie de décomposition* > de la carte de rétro-ingénierie. Cette directive est sélectionnée dans le cas ou des éléments du modèle de processus en cours de construction possèdent un niveau de granularité trop élevé et nécessitent d'être décomposés en éléments plus fins pour pouvoir guider l'ingénieur d'application dans la construction du produit souhaité par l'application de la méthode. La DRI_{13} est une directive tactique de type plan, la Figure 44 illustre la structure de cette directive.

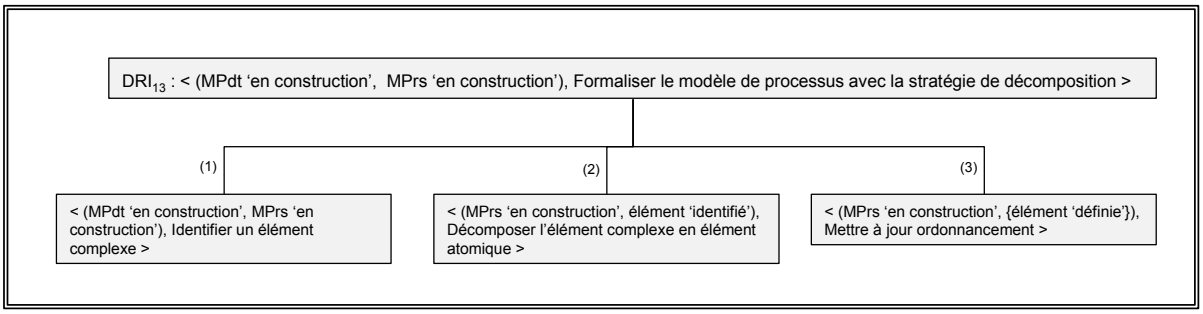


Figure 44. Structure de la DRI_{13}

(1) < *(MPdt 'en construction', MPrs 'en construction'), Identifier élément complexe* > : Cette directive suggère à l'ingénieur de méthodes d'analyser le modèle de processus en cours de construction dans le but d'identifier les éléments de ce modèle nécessitant d'être décomposés.

(2) < *(MPrs 'en construction', élément 'identifié'), Décomposer l'élément complexe en élément atomique* > : Cette directive invite l'ingénieur de méthodes à décomposer l'élément de processus identifié par la directive précédente en éléments plus atomiques permettant de mieux assister l'ingénieur d'application dans la construction de la ou des parties de produit cible.

(3) < *(MPrs 'en construction', {élément 'définie'}), Mettre à jour ordonnancement* > : Cette directive suggère à l'ingénieur de méthodes de mettre à jour l'ordonnancement des éléments du modèle de processus en cours de construction pour tenir compte des nouveaux éléments définis.

3.2.14. Réaliser « Arrêter avec la Stratégie de validation » (DRI_{14})

La DRI_{14} est associée à la section C14 de la carte de rétro-ingénierie, l'objectif de cette directive est de valider la méthode obtenue par l'application du modèle de processus de rétro-ingénierie *MIME*.

La Figure 45 présente la structure de cette directive de type plan. Ce plan propose de (1) vérifier la cohérence des modèles définis pour la méthode (2) Tester la méthode en l'appliquant pour le développement de 'SI test' et (3) Documenter la méthode pour faciliter sa compréhension et son apprentissage par les utilisateurs.

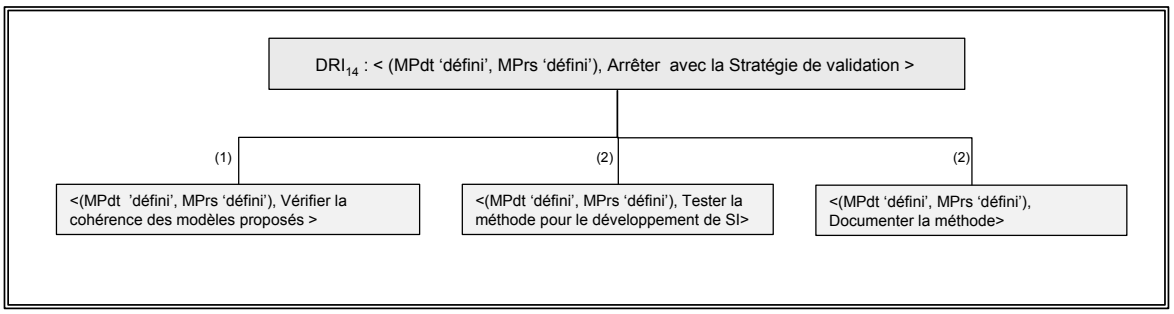


Figure 45. Structure de la DRI_{14}

3.2.15. Réaliser « Arrêter avec la Stratégie de validation » (DRI_{15})

La DRI_{15} est associée à la section <*Formaliser le modèle de processus, Arrêter, Stratégie de validation*> de la carte de rétro-ingénierie. Cette directive est similaire à la précédente, elle est décrite par un plan identique à celui présenté à la Figure 45.

4. Conclusion

Dans ce chapitre nous avons proposé une démarche pour assister l'ingénieur de méthodes dans le projet de rétro-ingénierie de méthode. Le modèle de processus de cette démarche a été formalisé à l'aide d'une directive stratégique c'est à dire avec une *carte* et un ensemble de *directives*. Ce modèle est ainsi facilement extensible par l'ajout de nouvelles stratégies pour permettre l'identification des éléments de la méthode et réaliser la formalisation des modèles de produit et de processus de cette dernière.

CHAPITRE 5 :
EVOLUTION DES METHODES

1. Introduction

Ce chapitre, intitulé *Evolution des méthodes,* présente le deuxième modèle de processus de la méthode MIME à savoir le *modèle de processus d'évolution*. Cette *démarche d'évolution* est intégrée à un modèle de processus générique d'ingénierie de méthodes qui permet de la positionner dans le contexte général de l'ingénierie des méthodes.

Le plan de ce chapitre est organisé comme suit : la première partie est consacrée à la présentation de la notion d'évolution des méthodes. La deuxième partie présente le modèle de processus générique d'ingénierie de méthodes. La troisième partie, quant à elle, est dédiée à la présentation du modèle de processus d'évolution sous la forme d'une *carte* et d'un ensemble de *directives.*

2. La notion de l'évolution d'une méthode

Le Petit Robert [Robert 86] propose la définition suivante du terme *Evolution* :

> Evolution : dérivé du latin *evolutio.* Suite de transformations dans un même sens; transformation graduelle assez lente, ou formée de changements successifs insensibles.

A la lumière de cette définition et de celle de la notion de *méthode* nous proposons pour l'évolution de méthode la définition suivante : « *L'évolution d'une méthode consiste à apporter un ensemble de transformations aux modèles de produit et de processus d'une méthode existante afin de produire une nouvelle méthode satisfaisant les objectifs d'évolution* ».

Pour développer une démarche d'évolution de méthodes conforme à cette définition, on s'est basé sur le principe suivant : Utiliser un modèle initial (le modèle *As-Is*) comme base d'évolution pour aboutir au modèle souhaité (le modèle *To-Be*) en appliquant un ensemble de transformations. Une méthode étant composée d'un ou plusieurs modèles de produit et d'un ou plusieurs modèles de processus, ce principe doit alors être appliqué à l'ensemble de ces modèles afin d'obtenir la méthode résultat (*la méthode To-Be*).

Comme nous l'avons énoncé dans la problématique, le projet d'évolution d'une méthode est un projet complexe à mener. De ce fait, nous proposons que toute démarche visant à guider l'ingénieur de méthodes dans ce type de projet respecte les contraintes suivantes :

(1) Tenir compte des spécificités de chaque méthode, du contexte de son application, ainsi que de la nature de ses modèles de produit et de processus.

(2) Respecter les règles de construction définies par le méta-modèle de méthodes de référence. En effet, la méthode *To-Be* doit être une instance de ce méta-modèle.

(3) Garantir la cohérence et la validité de la méthode *To-Be* en offrant des mécanismes de contrôle et de validation permettant de tester et de corriger la méthode résultat.

(4) Assurer la continuité avec la méthode *As-Is*. En effet, en faisant évoluer une méthode il faut veiller à garantir la continuité des services offerts par la méthode d'origine.

(5) Respecter les objectifs et les exigences de l'évolution. La démarche proposée doit alors offrir un guidage à l'ingénieur de méthodes pour capturer ces exigences et doit assurer par la suite le respect de ces exigences tout au long du processus d'évolution.

La Figure 46 résume la vue générale de l'évolution de méthodes présentée ci-dessus.

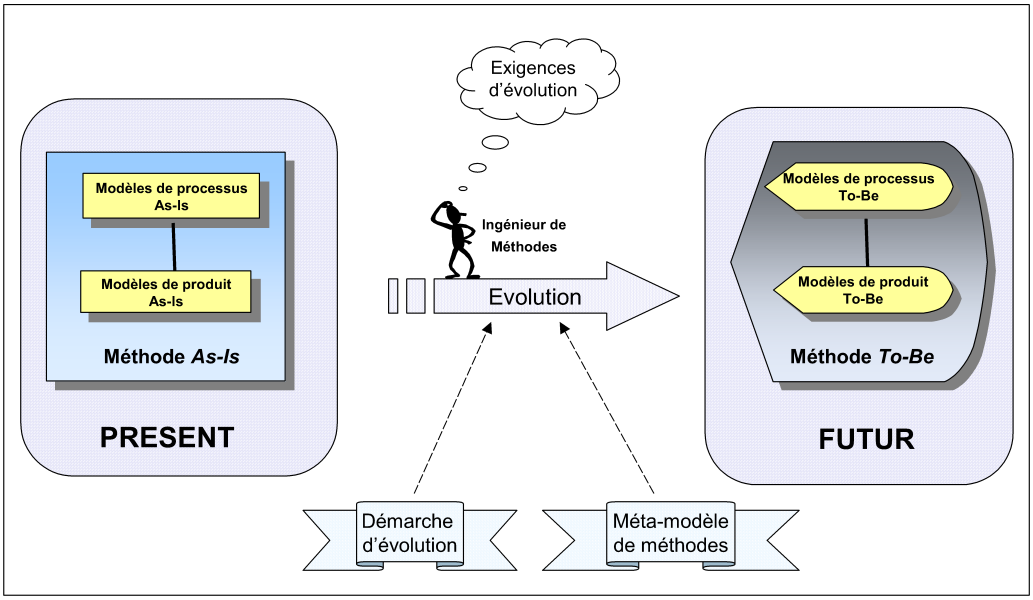


Figure 46. Vue générale de l'évolution de méthodes

Pour proposer une démarche d'évolution de méthodes dans l'optique présentée au paravent, nous avons cherché à identifier les différentes manières possibles pour réaliser l'évolution des modèles d'une méthode. En résultat, nous avons dégagé quatre possibilités: *Par adaptation de modèle*, *Par évolution de paradigme, Par abstraction de modèle* ou *Par instanciation de méta-modèle*.

- *Par adaptation de modèle:*

Cette manière de conduire l'évolution se base sur l'adaptation des modèles *As-Is* d'une méthode pour produire ceux de la méthode *To-Be*. L'adaptation de modèles est un sujet largement abordé en ingénierie de méthodes et en ingénierie des systèmes, différentes techniques ont été en fait proposées pour réaliser cette adaptation [Tolvanen 98], [Terrasse 03], [Boyd 04], [Rolland 04]. Sur la base de ces travaux et notamment ceux de J.Ralyté [Ralyté 04] nous proposons dans ce chapitre un ensemble d'opérateurs génériques pour l'adaptation de modèles.

- *Par évolution de paradigme* :

Cette manière de conduire l'évolution d'une méthode consiste à réaliser la ré-ingénierie d'un ou plusieurs modèles de la méthode en changeant le paradigme de modélisation utilisé. Cette technique vise généralement à améliorer l'expressivité du modèle à faire évoluer en utilisant un paradigme de modélisation plus riche, plus flexible ou plus modulaire.

Par exemple, un modèle de produit utilisant le formalisme E/R peut évoluer vers un modèle utilisant le paradigme orienté-objet. Un modèle de processus utilisant le paradigme *orienté activité* peut évoluer vers un modèle de processus contextuel mettant l'accent sur les contextes et les décisions conduisant à l'exécution des activités.

- *Par instanciation de méta-modèle* :

La technique d'instanciation de méta-modèle présentée au chapitre 2 peut être utilisée pour l'évolution de méthodes. Cette manière de conduire l'évolution consiste à remplacer un modèle de la méthode *As-Is*, jugé inadéquat, par une instance d'un méta-modèle existant adapté à la nature du produit fabriqué et du processus proposé par la méthode To-Be.

- *Par abstraction de modèle :*

Cette manière de conduire l'évolution vise à abstraire à partir d'un modèle de la méthode *As-Is* une nouvelle couche de modèle ayant un niveau d'abstraction plus élevé. Cette technique permet de séparer des aspects dissociables jusque là confondus dans un même modèle.

Les quatre techniques évoquées ci-dessus sont applicables dans la démarche d'évolution de méthodes que nous proposons. Elles sont intégrées comme des stratégies alternatives pour conduire l'évolution des modèles de produit ou de processus de la méthode *As-Is*. Ces stratégies sont développées en détail à la section 4 de ce chapitre. On verra que certaines s'appliquent à l'évolution des modèles de produit, d'autres à celle des modèles de processus et certaines à l'évolution des deux types de modèles.

3. Modèle de processus générique d'ingénierie de méthodes

Un certain nombre d'approches d'ingénierie de méthodes ont été proposées dans la littérature (Cf. Chapitre 2). L'objectif de cette section est de positionner la démarche d'évolution que nous proposons dans le contexte général de l'ingénierie de méthodes. Pour le faire, nous utilisons le modèle de processus générique d'ingénierie de méthodes proposé dans [Ralyté 03].

L'objectif de ce modèle est de mettre en évidence les *intentions génériques d'ingénierie* ; c'est-à-dire celles qui sont présentes dans toute démarche d'ingénierie de méthodes quelle que soit sa spécificité et de proposer des *stratégies possibles* pour atteindre ces intentions. Le recensement des stratégies n'est sans doute pas exhaustif mais l'utilisation d'une *carte* pour formaliser ce modèle permet d'en ajouter de nouvelles chaque fois qu'on le souhaite.

Le modèle de processus générique d'ingénierie de méthodes est formalisé par la carte présentée à la Figure 47 qui est aussi appelée *carte générique d'IM*. On considère ici que tout processus d'ingénierie de méthodes est basé sur deux activités principales : (1) la définition de l'objectif d'ingénierie et (2) la construction d'une méthode permettant d'atteindre cet objectif. Dans la carte générique, ces activités sont représentées par deux intentions génériques : *Définir l'objectif de l'ingénierie* et *Construire une méthode*. La première intention correspond à l'élicitation par l'ingénieur de méthodes du but qu'il fixe pour le projet d'ingénierie de méthodes. La seconde permet de construire une méthode qui correspond à ce but.

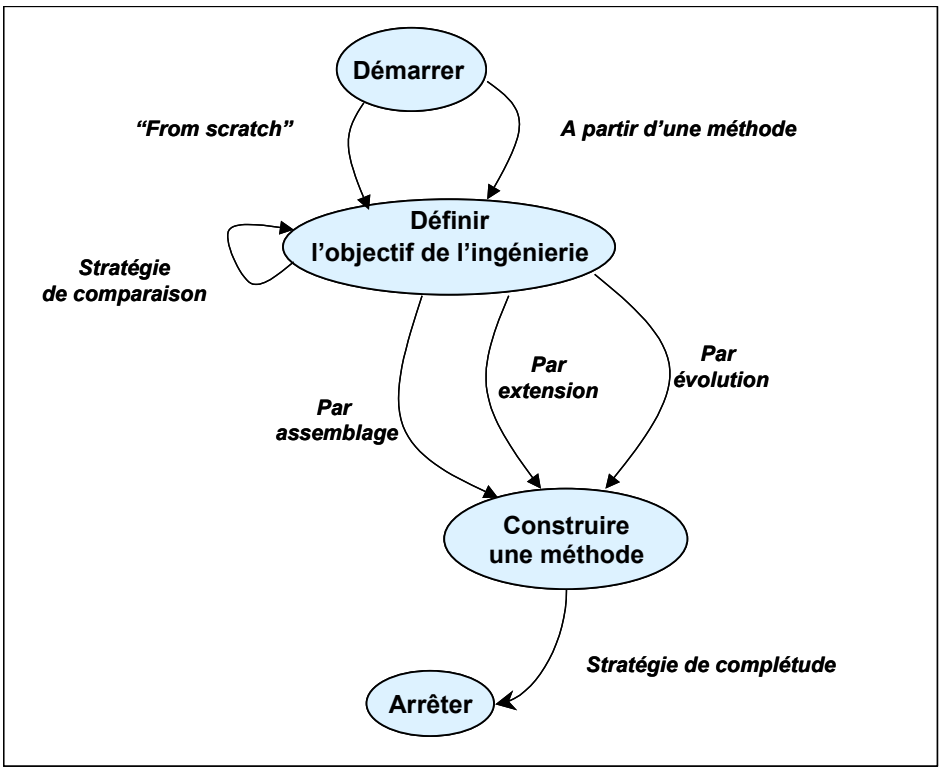


Figure 47. Modèle de processus générique d'ingénierie de méthode

Comme le montre la Figure 47, deux stratégies alternatives permettent d'atteindre l'intention *Définir l'objectif de l'ingénierie* : la stratégie *A partir d'une méthode* qui est applicable lorsque l'ingénieur est dans une situation de recherche d'adaptation d'une méthode connue et la stratégie *From scratch* lorsqu'il part 'de rien'. La sélection d'une des deux stratégies dépend de la situation du projet en cours. Si l'ingénieur de méthodes considère que la méthode habituelle est appropriée pour le projet en cours mais nécessite tout de même une certaine adaptation il est conseillé d'appliquer la stratégie *A base d'une méthode.* Au contraire s'il est convaincu qu'aucune méthode disponible n'est adéquate pour le projet en question et qu'il est nécessaire de construire une nouvelle méthode il peut opter pour la stratégie *From scratch*.

Trois stratégies permettent d'atteindre l'intention *Construire une méthode* correspondant à trois démarches d'ingénierie de méthodes différentes : *Par évolution, Par assemblage* et *Par extension*. Le choix parmi ces stratégies est motivé par les objectifs de l'ingénieur de méthodes spécifiés ci-dessus. La première stratégie correspond à la démarche d'évolution de méthodes que nous proposons. La deuxième stratégie (*Par assemblage)*, présentée dans [Ralyté 01], permet de construire une nouvelle méthode ou d'enrichir une méthode existante par assemblage de composants de méthodes stockés dans une base de méthodes. La dernière stratégie (*Par extension)* consiste à étendre une méthode existante en appliquant des patrons d'extension [Deneckere 98], [Deneckere 01]. Une présentation des deux dernières stratégies est fournie au Chapitre 2, le développement de la stratégie *Par évolution* pour atteindre l'intention *Construire une méthode* fait l'objet de la section suivante.

4. Modèle de processus de l'évolution de méthodes

Comme nous l'avons énoncé ci-dessus, la démarche d'évolution proposée a pour objectif de guider l'ingénieur de méthodes dans l'activité d'évolution d'une méthode existante (*la méthode As-Is*) vers une nouvelle méthode (*la méthode To-Be*) satisfaisant ainsi les exigences d'évolution de départ et les règles de construction de méthodes. Cette section présente le modèle de processus formalisant cette démarche. Ce modèle est décrit à l'aide d'une directive stratégique, c'est-à-dire à l'aide d'une *carte*, *la carte d'évolution* et d'un ensemble de directives décrivant les différentes sections de celle-ci.

4.1. La carte d'évolution

La Figure 48 présente la *carte d'évolution*, cette carte affine la section <*Définir l'objectif de l'ingénierie, Construire une méthode, Par évolution*> de la carte générique d'IM. La carte d'évolution a pour signature <*(MPdt As-Is*[5]*, MPrs As-Is*[6]*), Construire une méthode par évolution*>,

[5] Modèle de produit As-Is
[6] Modèle de processus As-Is

elle comporte trois intentions principales: *Spécifier les exigences d'évolution, Faire évoluer le modèle de produit* et *Faire évoluer le modèle de processus*.

- L'intention *Spécifier les exigences d'évolution* vise la capture et la spécification des exigences d'évolution
- L'intention *Faire évoluer le modèle de produit* vise l'évolution du modèle de produit *As-Is*
- L'intention *Faire évoluer le modèle de processus* vise l'évolution du modèle de processus *As-Is* après que le produit ait été transformé.

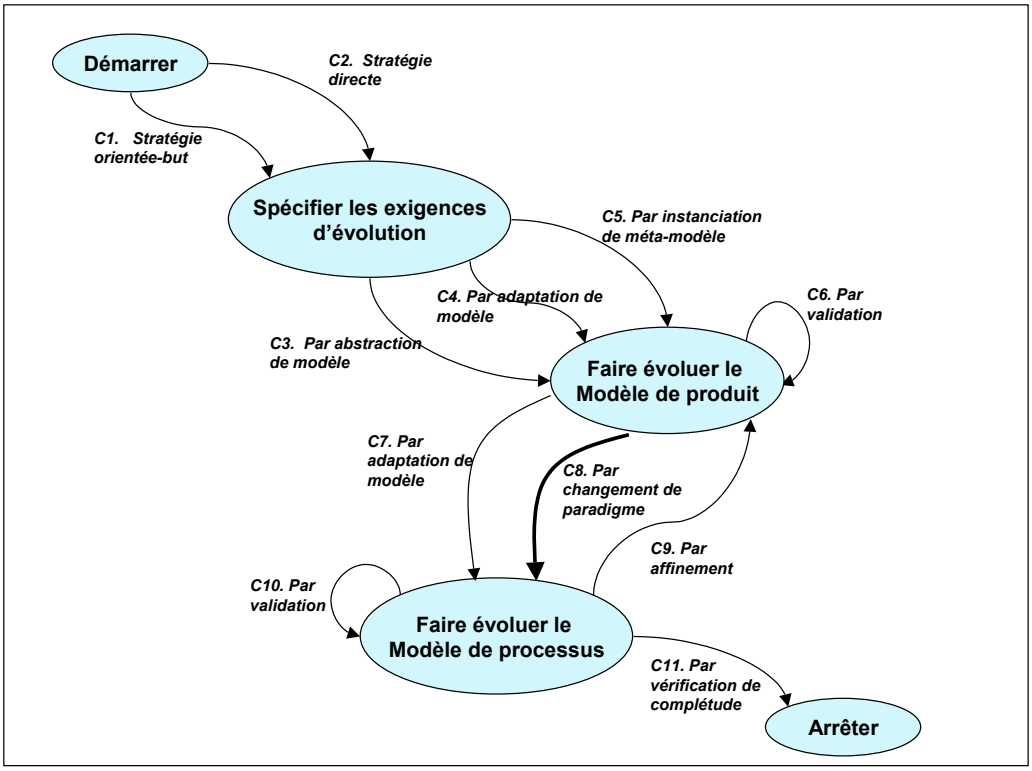


Figure 48. La carte d'évolution

La carte d'évolution comporte onze sections présentées au Tableau 1. Ces sections sont numérotées de C1 à C11, elles proposent des stratégies pour la réalisation des intentions composant cette carte. La section C_8 (en gras) est un cluster qui sera décomposé en quatre sections mutuellement exclusives.

C_1 : < Démarrer, Spécifier les exigences d'évolution, Stratégie orientée-but >
C_2 : < Démarrer, Spécifier les exigences d'évolution, Stratégie directe >
C_3 : < Spécifier les exigences d'évolution, Faire évoluer le modèle de produit, Par abstraction de modèle >
C_4 : < Spécifier les exigences d'évolution, Faire évoluer le modèle de produit, Par adaptation de modèle >
C_5 : < Spécifier les exigences d'évolution, Faire évoluer le modèle de produit, Par instantiation de méta-modèle >
C_6 : < Faire évoluer le modèle de produit, Faire évoluer le modèle de produit, Par validation >
C_7 : < Faire évoluer le modèle de produit, Faire évoluer le modèle de processus, Par adaptation de modèle >

C_8 : < Faire évoluer le modèle de produit, Faire évoluer le modèle de processus, Par changement de paradigme >
C_9 : < Faire évoluer le modèle de processus, Faire évoluer le modèle de produit, Par affinement >
C_{10} : < Faire évoluer le modèle de processus, Faire évoluer le modèle de processus, Par validation >
C_{11} : < Faire évoluer le modèle de processus, Arrêter, Par vérification de complétude >

Tableau 5. Les sections de la carte d'évolution

L'intention *Spécifier les exigences d'évolution* est le point de départ du processus d'évolution d'une méthode. Les exigences d'évolution complètent les objectifs d'ingénierie identifiés lors de l'exécution de la carte générique d'IM (Figure 47).

Les exigences d'évolution émanent de l'expérience d'utilisation de la méthode à faire évoluer et traduisent d'une part, les limites constatées dans la méthode *As-Is* auxquelles il faut remédier et d'autre part, les attentes concernant la méthode *To-Be*. Ces exigences peuvent être exprimées de manière informelle ou sous la forme d'une *carte des exigences* utilisant le formalisme MAP.

Pour atteindre l'intention *Spécifier les exigences d'évolution* deux stratégies alternatives correspondant à deux situations d'ingénierie différentes sont proposées : *Stratégie directe* (section C2) et *stratégie orientée-but* (section C1). La première stratégie est appliquée dans le cas où la situation d'ingénierie permet de cerner dès le début du projet d'évolution, la liste et la nature des transformations à apporter à la méthode *As-Is*. L'ingénieur de méthodes peut alors formuler directement le document d'exigences. La deuxième stratégie doit être retenue dans le cas où les exigences d'évolution ne sont pas claires et qu'il faut avoir recours à une découverte des buts que doit satisfaire la méthode *To-Be.* L'ingénieur de méthodes doit transformer ces buts en transformations à apporter à la méthode *As-Is*.

Une fois spécifiées, les exigences d'évolution sont utilisées comme base dans le reste du processus d'évolution. A chaque étape de l'évolution de la méthode *As-Is*, l'ingénieur de méthodes doit veiller à respecter ces exigences.

Pour atteindre l'intention *Faire évoluer le modèle de produit* cinq stratégies sont proposées: *Par abstraction de modèle* (section C3), *Par adaptation de modèle* (section C4)*, Par instanciation de méta-modèle* (section C5), *Par validation* (section C8) et *Par affinement* (section C9). La stratégie *Par abstraction de modèle* doit être sélectionnée dans le cas où l'on constate, lors de l'expérience d'utilisation de la méthode *As-Is,* qu'un modèle de produit de cette dernière contient des éléments n'ayant pas tous le même niveau d'abstraction. Ce cas de figure rend difficile la compréhension de la méthode par ses utilisateurs et pose des problèmes lors de son application. La directive associée à cette stratégie offre un guidage à l'ingénieur de méthodes pour abstraire à partir du modèle de produit

As-Is deux ensembles d'éléments correspondant chacun à un niveau d'abstraction. Le modèle de produit *To-Be* comportera deux ensembles de concepts en relation hiérarchique.

La stratégie *Par adaptation de modèle* est sélectionnée dans le cas où la satisfaction des exigences d'évolution suggère d'opérer un certain nombre d'adaptations au modèle de produit *As-Is*. Ce cas de figure est très fréquent dans les projets d'évolution de méthode. La directive associée à cette stratégie offre un guidage basé sur un ensemble d'opérateurs génériques d'adaptation de modèles.

La stratégie *Par instanciation de méta-modèle* est sélectionnée dans le cas où le modèle de produit *As-Is* se limite à une description informelle des concepts de la méthode ou lorsque celui-ci est jugé inadéquat. Le modèle de produit *To-Be* est alors obtenu par instanciation d'un méta-modèle de produit existant.

La stratégie *Par affinement* permet, après avoir fait évoluer le modèle de processus, de répercuter les transformations subies par les éléments de ce modèle sur les éléments du modèle de produit correspondant.

Finalement, pour terminer le processus d'évolution du modèle de produit, la stratégie *Par validation* est appliquée pour s'assurer de la validité du modèle de produit *As-Is*.

L'intention *Faire évoluer le modèle de processus* succède à celle de l'évolution du modèle de produit dans le processus d'évolution de méthodes. Deux stratégies permettent d'atteindre cette intention : *Par adaptation de modèle* (section C7) et *Par changement de paradigme* (section C8). La stratégie réflexive *Par validation* (section C10) est appliquée pour tester et garantir la validité du ou des modèles de processus obtenus par évolution.

La stratégie *Par adaptation de modèle* a un double rôle : (1) répercuter les évolutions subies par les éléments du modèle de produit sur les éléments du modèle de processus qui les manipule et (2) assurer l'évolution du modèle de processus *As-Is* conformément aux exigences d'évolution. Les adaptations réalisées par cette stratégie utilisent les opérateurs génériques d'adaptation de modèles.

La section C8, correspondant à la stratégie par *changement de paradigme* pour faire évoluer le modèle de processus, est un cluster. Elle est décomposée en quatre sections mutuellement exclusives correspondant aux quatre stratégies *Orientée-activité*, *Orientée-stratégie*, *Orientée-contexte* et *Orientée-patron*. La Figure 49 illustre la décomposition de ce cluster.

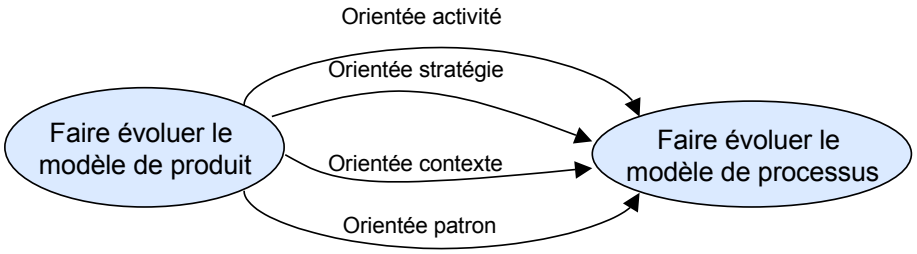


Figure 49. Décomposition du cluster de la section C8

Ces quatre stratégies correspondent à l'approche 'par évolution de paradigme' introduite à la deuxième section de ce chapitre. En effet, l'application de l'une ou l'autre des stratégies permet de réaliser la ré-ingénierie du modèle de processus *As-Is* en utilisant un nouveau paradigme de modélisation de processus. La construction du modèle de processus *To-Be* est alors basée sur les éléments du modèle de processus *As-Is* et sur les exigences d'évolution conformément au paradigme choisi.

Une fois l'évolution des modèles de la méthode est achevée, la stratégie *par vérification de complétude* est sélectionnée pour arrêter le processus.

La carte d'évolution propose plusieurs démarches alternatives pour réaliser l'évolution d'une méthode. En effet, plusieurs chemins permettent de progresser de l'intention *Démarrer* à l'intention *Arrêter* dans la carte. Durant l'exécution de la carte d'évolution la sélection d'un chemin se fait d'une manière dynamique et contextuelle. A tout moment de l'exécution de la carte l'ingénieur de méthodes peut décider quelle intention réaliser et quelle stratégie utiliser selon les spécificités de la méthode As-Is, la situation des modèles en cours d'évolution et les exigences d'évolution.

Un guidage est offert à l'ingénieur de méthode pour cette prise de décision avec les directives de progression de la carte (Directives de Sélection d'Intention et Directives de Sélection de Stratégie). La présentation des DSI et DSS de la carte d'évolution fait l'objet des deux prochaines sections de ce chapitre.

4.1.1 Directives de Sélection d'Intention associées à la carte d'évolution

La carte d'évolution contient deux Directives de Sélection d'Intention (DSI), une pour l'intention I_3 : *Faire évoluer le modèle de produit* et une pour l'intention I_4 : *Faire évoluer le modèle de processus*. Le Tableau 2 présente les DSI associées à cette carte.

Intention	DSI
I_1 : Démarrer	N/A
I_2 : Spécifier les exigences d'évolution	N/A
I_3 : Faire évoluer le modèle de	DSI_1 : < (MPdt 'évolué', MPrs As-Is), Progresser depuis Faire évoluer le

produit	modèle de produit >
I_4 : Faire évoluer le modèle de processus	DSI_2 : <(MPdt 'évolué', MPrs 'évolué'), Progresser depuis Faire évoluer le modèle de processus>
I_5 : Arrêter	N/A

Tableau 6. Les DSI de la carte d'évolution

4.1.1.1. Progresser depuis « Faire Evoluer le modèle de produit » (DSI_1)

La DSI_1, présentée à la Figure 50, est associée à l'intention *Faire évoluer le modèle de produit* de la carte d'évolution. Elle aide l'ingénieur de méthodes à progresser dans la carte après avoir réalisé cette intention. Comme le montre la Figure 50, l'ingénieur de méthodes a deux possibilités : atteindre l'intention *Faire évoluer le modèle de produit* (boucle) ou atteindre l'intention *Faire évoluer le modèle de processus*. Le premier choix est réalisé en sélectionnant la DRI_6 associée la *Stratégie de validation*. Le deuxième choix est réalisé en sélectionnant la DSS_3 proposant deux stratégies : *Par adaptation de modèle* et *Par changement de paradigme*.

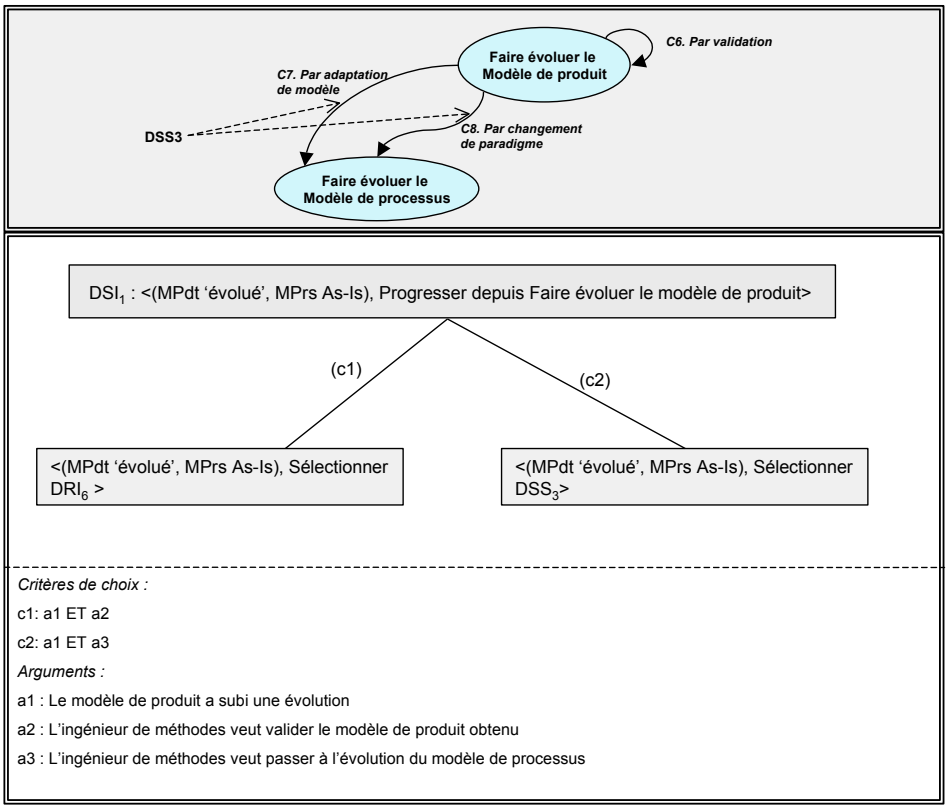


Figure 50. La structure de la DSI_1

4.1.1.2. Progresser depuis « Faire évoluer le modèle de processus » (DSI_2)

La DSI_2 est une directive tactique de type choix, elle offre un guidage à l'ingénieur de méthodes pour progresser dans la carte d'évolution à partir de l'intention *Faire évoluer le modèle de processus.* Comme le montre la Figure 51, cette directive propose trois alternatives : (1) réaliser l'intention

Faire évoluer le modèle de processus en sélectionnant la DRI_{13} (2) réaliser l'intention *Faire évoluer le modèle de produit* en sélectionnant la DRI_{12} (3) réaliser l'intention *Arrêter* en sélectionnant la DRI_{14}.

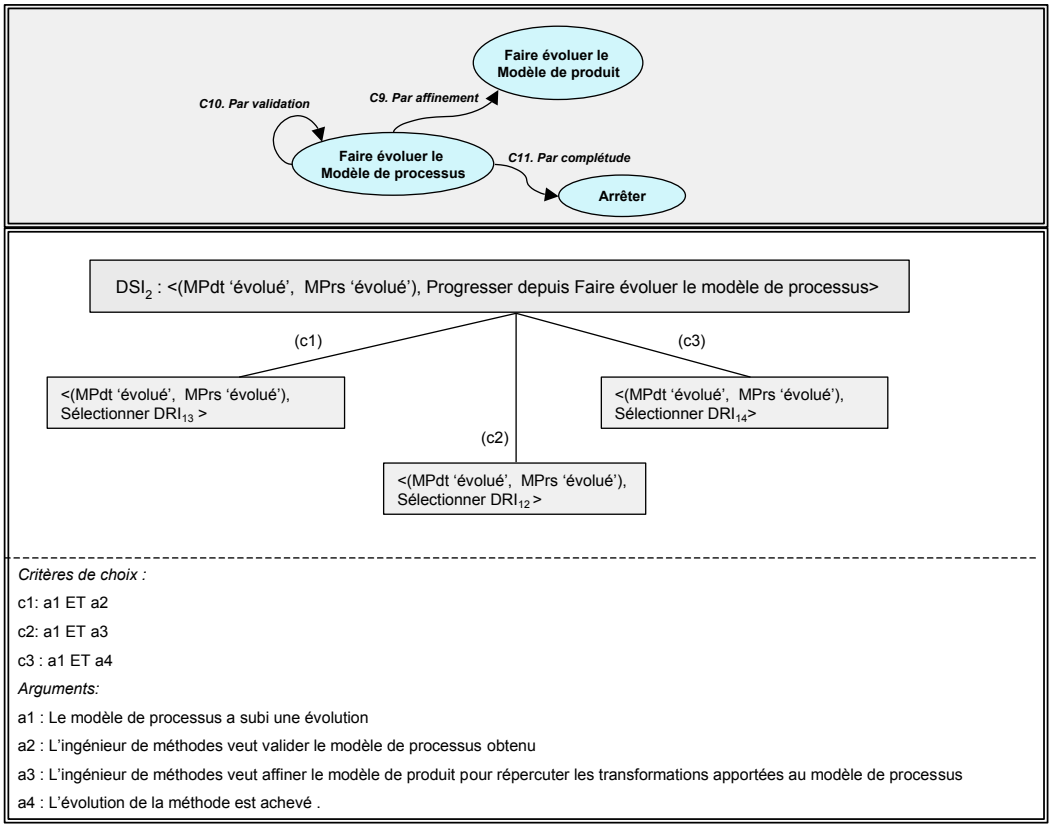


Figure 51. La structure de la DSI_2

4.1.2. Directives de Sélection de Stratégie associées à la carte d'évolution

Pour tout ensemble de sections parallèles dans la carte d'évolution on associe une Directive de Sélection de Stratégie (DSS) qui aide l'ingénieur de méthodes à choisir une stratégie parmi l'ensemble des stratégies proposées pour satisfaire l'intention cible. Le Tableau 3 présente les DSS associées à la carte d'évolution.

Paire d'intentions	DSS
(I_1 - I_2) : (Démarrer, Spécifier les exigences d'évolution)	DSS_1 : < (MPdt As-Is, MPrs As-Is), Progresser vers Spécifier les exigences d'évolution>
(I_2 – I_3) : (Spécifier les exigences d'évolution, Faire évoluer le modèle de produit)	DSS_2 : < (MPdt As-Is, MPrs As-Is), Progresser vers Faire évoluer le modèle de produit>
(I_3 – I_4) : (Faire évoluer le modèle de produit, Faire évoluer le modèle de processus)	DSS_3 : <(MPdt 'évolué', MPrs As-Is), Progresser vers Faire évoluer le modèle de processus>

Tableau 7. Les DSS de la carte d'évolution

4.1.2.1. Progresser vers « Spécifier les exigences d'évolution » (DSS_1)

La DSS_1 aide l'ingénieur de méthodes à démarrer le processus d'évolution en lui proposant deux stratégies alternatives *Stratégie orientée-but* et *Stratégie du processus guidé*. La Figure 52 illustre la DSS_1 qui est une directive tactique de type choix offrant deux possibilités :

(1) Sélectionner la directive DRI_1 pour appliquer la *Stratégie orientée-but.*

(2) Sélectionner la directive DRI_2 pour appliquer la *Stratégie directe.*

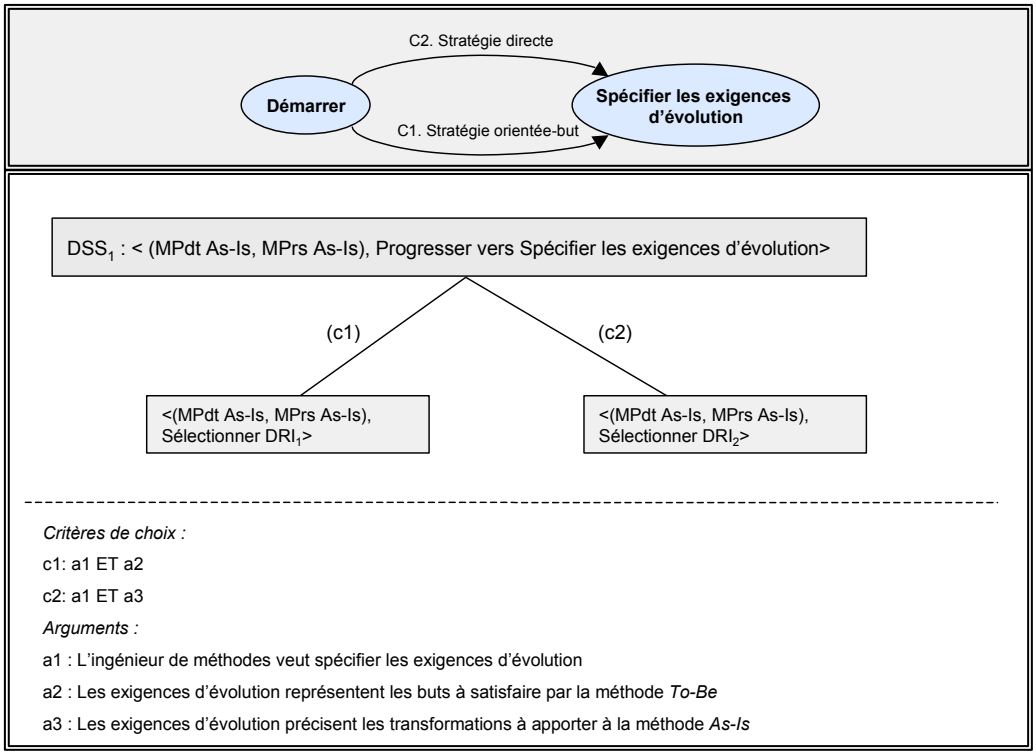


Figure 52. La structure de la DSS_1

4.1.2.2. Progresser vers « Faire évoluer le modèle de produit » (DSS_2)

Une fois les exigences d'évolution spécifiées, la DSS_2 aide l'ingénieur de méthodes à choisir la stratégie à appliquer pour atteindre l'intention *Faire évoluer le modèle de produit*. La Figure 53 présente la DSS_2 qui est une directive tactique de type choix, elle propose à l'ingénieur de méthodes trois possibilité : opter pour la stratégie *Par abstraction de modèle* en sélectionnant la $DRI_{3.3}$, opter pour la stratégie *Par instantiation de méta-modèle* en sélectionnant la $DRI_{3.5}$, ou opter pour la stratégie *Par adaptation de modèle* en sélectionnant la $DRI_{3.4}$. Le choix entre ces trois stratégies dépend de la nature du modèle de produit de la méthode initiale et des exigences d'évolution.

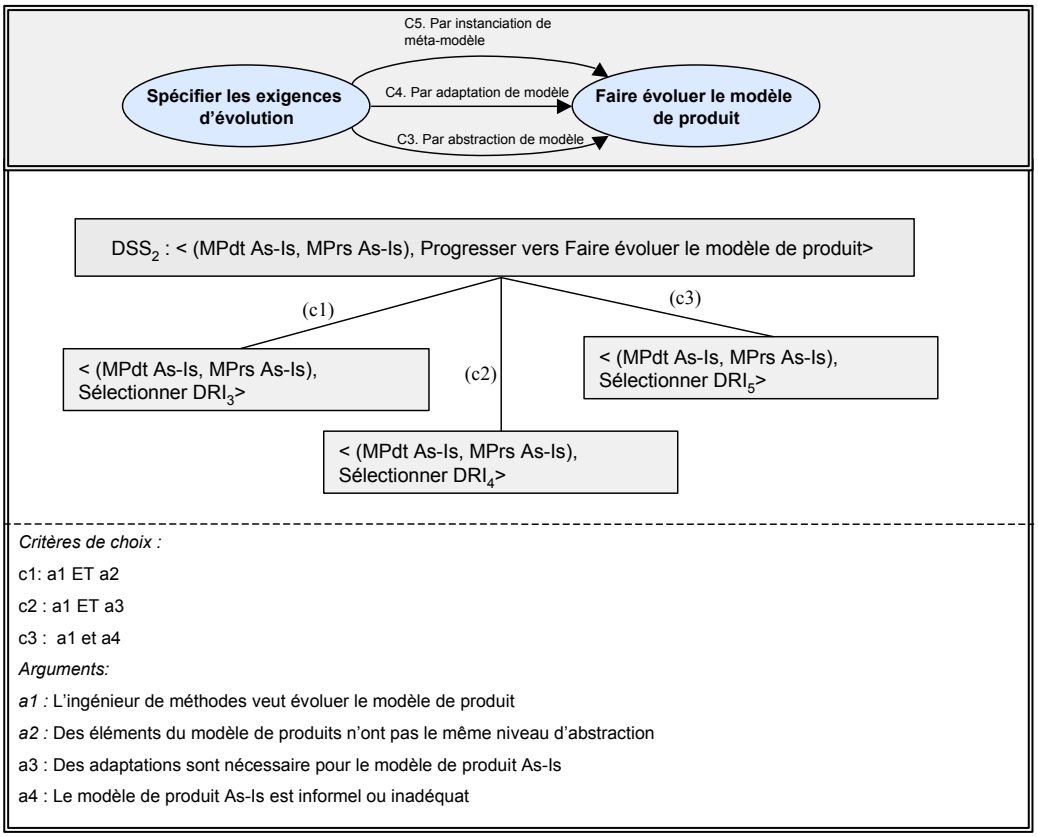


Figure 53. La structure de la DSS_2

4.1.2.3. Progresser vers « Faire évoluer le modèle de processus » (DSS_3)

La DSS_3 est de type choix, elle guide l'ingénieur de méthodes dans la sélection de stratégie pour progresser de l'intention *Faire évoluer le modèle de produit* vers l'intention *Faire évoluer le modèle de processus* de la carte d'évolution. La section C8 de cette carte étant décomposée en quatre stratégies : *Orientée activité*, *Orientée stratégie*, *Orientée contexte* et *Orientée patron*. Comme le montre la Figure 54 la DSS_3 intègre cette décomposition et offre à l'ingénieur de méthodes cinq possibilités de progression :

- Sélectionner la DRI_7 associée à la stratégie *Par adaptation de modèle,*
- Sélectionner la DRI_8 associée à la stratégie *Orientée activité,*
- Sélectionner la DRI_9 associée à la stratégie *Orientée stratégie,*
- Sélectionner la DRI_{10} associée à la stratégie *Orientée contexte,*
- Sélectionner la DRI_{11} associée à la stratégie *Orientée patron.*

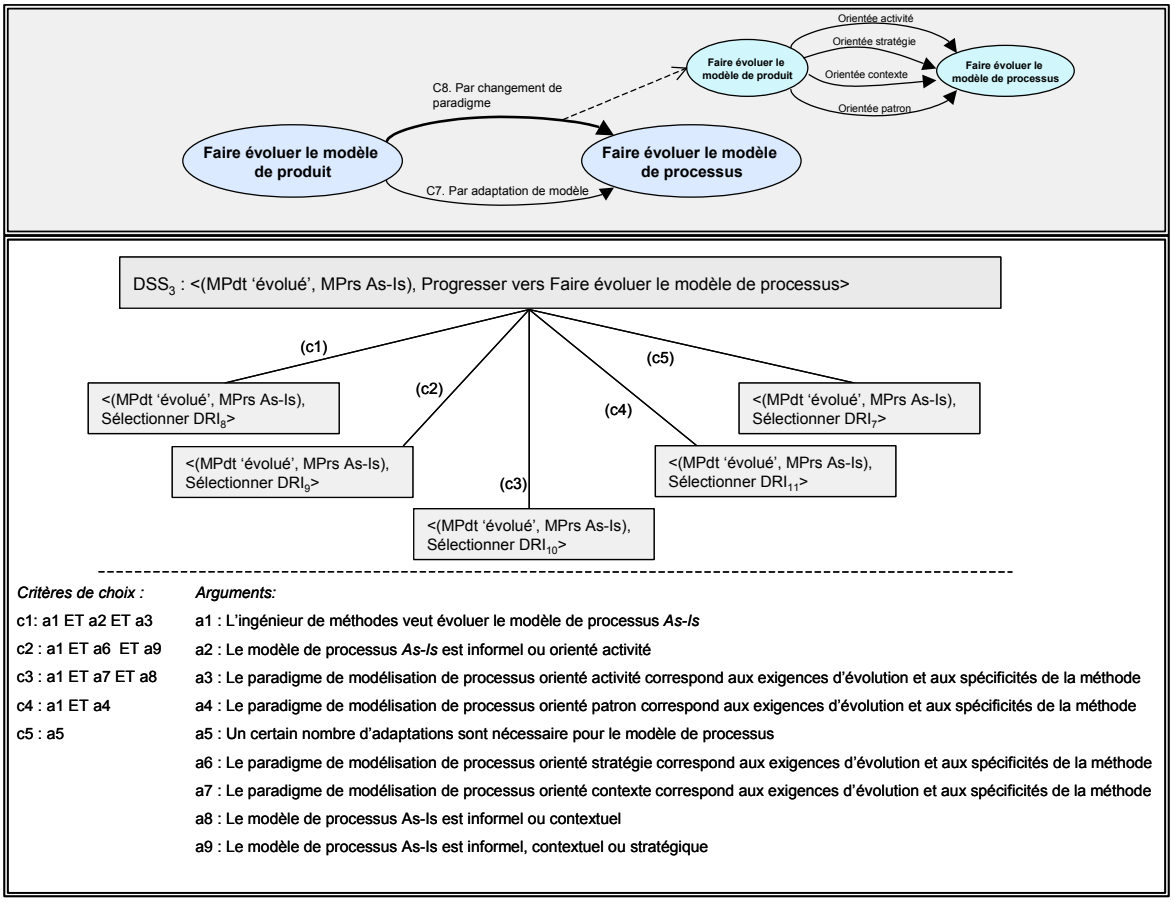


Figure 54. La structure de la DSS_3

4.1.3. Tableau des DRI associées à la carte d'évolution

La carte d'évolution comporte 11 sections, à chacune d'entre elles on associe une DRI (excepté à la section C8 de type cluster à laquelle on associe quatre DRI correspondant aux quatre stratégies composant ce cluster). Le Tableau 8 présente ces DRI.

Identifiant	Signature de DRI	Type
DRI_1	< (MPdt As-Is, MPrs As-Is), Spécifier les exigences d'évolution avec la stratégie orientée but >	Tactique
DRI_2	< (MPdt As-Is, MPrs As-Is), Spécifier les exigences d'évolution avec la stratégie directe >	Tactique
DRI_3	< (MPdt As-Is, MPrs As-Is), Faire évoluer le modèle de produit Par abstraction de modèle >	Stratégique
DRI_4	< (MPdt As-Is, MPrs As-Is), Faire évoluer le modèle de produit Par adaptation de modèle >	Tactique
DRI_5	< (MPdt As-Is, MPrs As-Is), Faire évoluer le modèle de produit Par instanciation de méta-modèle >	Tactique
DRI_6	< (MPdt 'évolué'), Faire évoluer le modèle de produit Par validation >	Tactique
DRI_7	< (MPdt 'évolué', MPrs As-Is), Faire évoluer le modèle de processus Par adaptation de modèle >	Tactique

DRI_8	< (MPdt 'évolué', MPrs As-Is), Faire évoluer le modèle de processus avec la stratégie orientée-activité>	Tactique
DRI_9	< (MPdt 'évolué', MPrs As-Is), Faire évoluer le modèle de processus avec la stratégie orientée-stratégie >	Stratégique
DRI_{10}	< (MPdt 'évolué', MPrs As-Is), Faire évoluer le modèle de processus avec la stratégie orientée-contexte >	Tactique
DRI_{11}	< (MPdt 'évolué', MPrs As-Is), Faire évoluer le modèle de processus avec la stratégie orientée-patron >	Stratégique
DRI_{12}	< (MPdt 'évolué', MPrs 'évolué'), Faire évoluer le modèle de produit Par affinement >	Tactique
DRI_{13}	< (MPrs 'évolué'), Faire évoluer le modèle de processus Par validation >	Tactique
DRI_{14}	< (MPdt To-Be, MPrs To-Be), Arrêter Par vérification de complétude >	Tactique

Tableau 8. Les DRI de la carte d'évolution

4.2. Sections de la carte d'évolution

Dans cette section, nous décrivons les 14 DRI associées aux sections de la carte d'évolution identifiées au Tableau 8. Chacune des sous-sections suivantes décrit l'une de ces DRI. Les DRI tactiques sont présentées dans cette section. Les DRI stratégiques sont partiellement présentées dans cette section. En effet, pour chaque DRI stratégique (DRI_3, DRI_9 et DRI_{11}), on présente sa carte, ses DSI et DSS et on identifie ses DRI. Ces dernières sont détaillées en section 4.3.

4.2.1. Réaliser « Spécifier les exigences d'évolution avec la stratégie orientée-but » (DRI_1)

La DRI_1 est associée à la section C1 de la carte d'évolution. L'objectif de cette directive est d'offrir un guidage à l'ingénieur de méthodes pour la capture et la spécification des exigences d'évolution. Comme nous l'avons défini ci-dessus la stratégie *orientée-but* est sélectionnée dans le cas où la spécification des exigences d'évolution est dirigée par la capture des buts que doit satisfaire la méthode *To-Be* et les différentes manières d'atteindre ces buts.

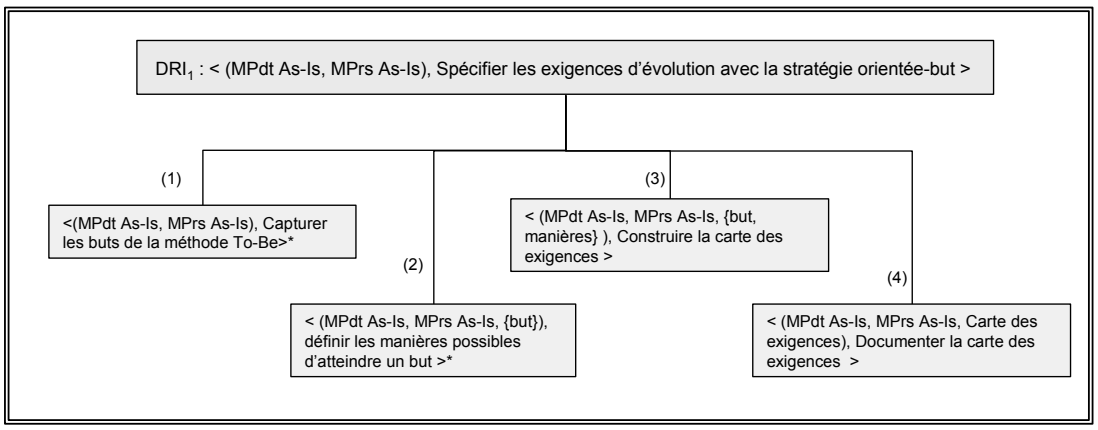


Figure 55. La structure de la DRI_1

La DRI_1 est modélisée par la directive plan présentée à la Figure 55. Ce plan est décomposé en quatre intentions supportées par quatre directives :

(1) *Capturer les buts de la méthode To-Be* : la directive associée à cette intention est de type plan, elle a pour objectif de guider l'ingénieur de méthodes dans la capture des buts qu'on souhaite atteindre par l'application de la méthode *To-Be*. Cette directive préconise l'utilisation de la technique d'interview ou celle d'organisation d'une séance de brainstorming avec les utilisateurs et/ou les créateurs de la méthode. La Figure 56 présente cette directive proposant à l'ingénieur de méthode de (1.1) préparer l'interview en définissant l'ensemble des questions à poser, (1.2) sélectionner le public avec le quel l'interview va être réalisé (1.3) réaliser l'interview (1.4) formuler les buts de la méthode To-Be en se basant sur une analyse des résultats de l'interview.

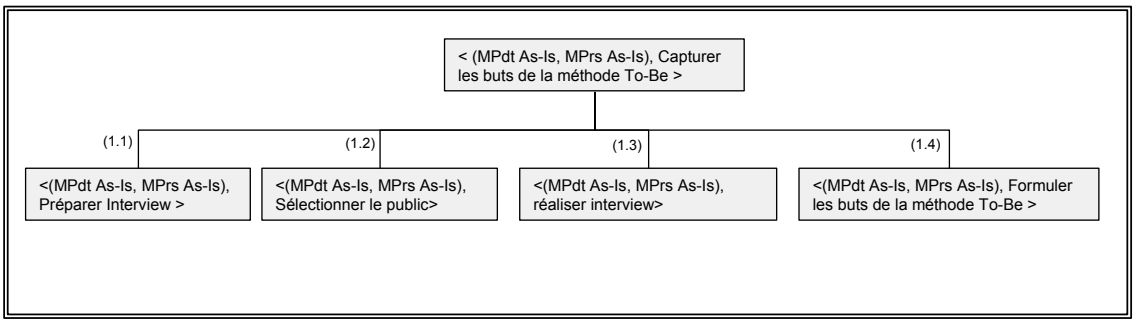


Figure 56. < (MPdt As-Is, MPrs As-Is), Capturer les buts de la méthode To-Be >

(2) *Définir les manières possibles* : la directive associée à cette intention suggère à l'ingénieur de méthodes de définir les différentes manières possibles permettant d'atteindre chacun des buts découverts ci-dessus. Cette directive est de type plan, elle est décrite à la Figure 57 et se décompose en deux sous-directives : < (but), Spécifier les manières d'atteindre but > et <({buts, manières}), Valider buts et manières >. La première sous-directive permet de spécifier les différentes manières d'atteindre un but en se basant sur le résultat des interviews réalisées pendant l'exécution de la directive précédente. La deuxième sous-directives suggère à l'ingénieur de méthode de valider les buts et les manières définis avec les utilisateurs et/ou les créateurs de la méthode.

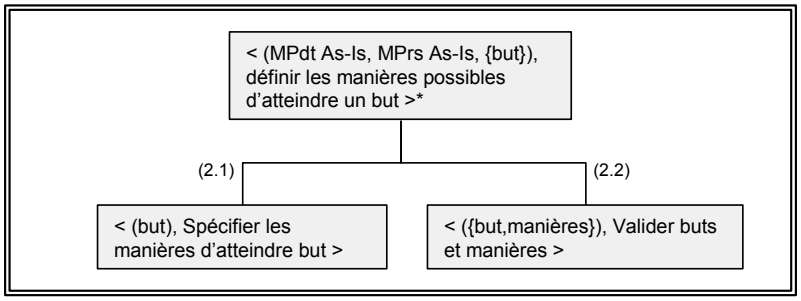


Figure 57. < (MPdt As-Is, MPrs As-Is, {but}), définir les manières possibles d'atteindre un but >

(3) Construire la carte des exigences : l'objectif de la directive correspondant à cette intention est la construction d'une carte des exigences reprenant les buts et les stratégies exprimés par l'application des deux directives précédentes. Elle suggère à l'ingénieur de méthodes de formaliser les buts capturés sous la forme d'intentions et les différentes manières pour les atteindre sous forme de stratégies dans une *carte* en ajoutant les deux stratégies *Démarrer* et *Arrêter*. La technique de construction d'une carte est présentée en détail à la section 4.3.2 de ce chapitre.

(4) *Documenter la carte des exigences :* N'ayant pas besoin de directives associées à la carte des exigences, l'ingénieur de méthodes est invité en appliquant cette directive à documenter la carte obtenue en décrivant chacune des sections la composant.

4.2.2. Réaliser « Spécifier les exigences d'évolution avec la stratégie directe » (DRI_2)

La DRI_2 est associée à la section C2 de la carte d'évolution. Cette directive est appliquée dans le cas où les exigences dévolution s'expriment par des recommandations précises de transformation à apporter à la méthode *As-Is*. Cette directive de type plan est présentée à la Figure 58.

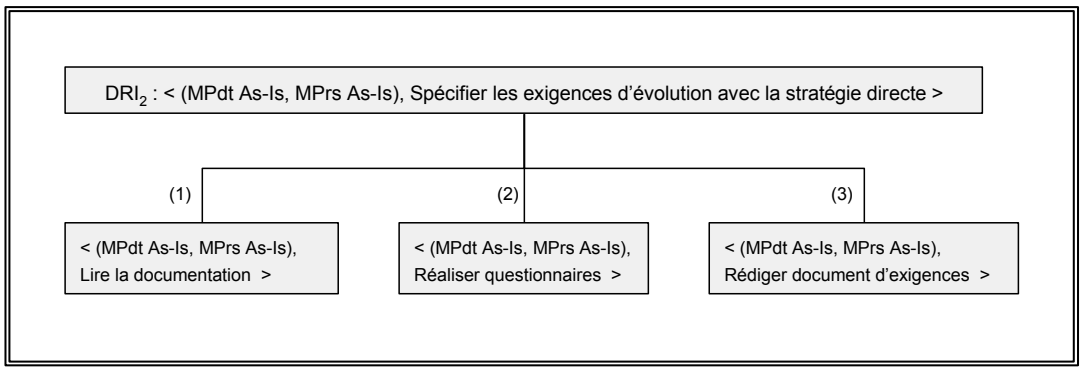


Figure 58. Structure de la DRI_2

Comme la montre cette figure, la DRI_2 est décomposée comme suit :

(1) < *(MPdt As-Is, MPrs As-Is), Lire la documentation* > : cette directive suggère à l'ingénieur de méthodes d'effectuer une lecture de la documentation disponible présentant l'expérience d'utilisation

de la méthode As-Is, les problèmes rencontrés dans les différents projets utilisant la cette méthode et les transformations qu'on souhaite lui apporter.

(2) < *(MPdt As-Is, MPrs As-Is), Réaliser questionnaires* > : cette directive propose de réaliser des questionnaires avec les utilisateurs de la méthode As-Is pour affiner la liste préliminaire des exigences identifiées en exécutant la directive précédente

(3) < *(MPdt As-Is, MPrs As-Is), Rédiger document d'exigences* > : la dernière directive composant la DRI_2 a pour objectif la rédaction d'un document d'exigences final qui sera utilisé comme base dans le processus d'évolution. En rédigeant ce document l'ingénieur de méthodes doit veiller à étudier et noter l'impact de chaque transformation voulu sur tous les éléments de la méthode.

Les exigences qu'on peut obtenir par l'application de cette directive représentent généralement les adaptations à apporter aux modèles de la méthode *As-Is*, les nouveaux aspects du SI qu'on souhaite modéliser par la méthode *To-Be* et éventuellement le type de paradigme à utiliser pour la construction des modèles de la méthode.

4.2.3. Réaliser « Faire évoluer le modèle de produit Par abstraction de modèle » (DRI_3)

La directive DRI_3 est associée à la stratégie *Par abstraction de modèle* pour faire évoluer le modèle de produit. Cette directive offre un guidage à l'ingénieur de méthodes pour abstraire à partir d'un modèle de produit *As-Is* une nouvelle organisation du modèle en couches, chaque couche comportant des éléments qui possèdent le même niveau d'abstraction. Cette directive met en œuvre le principe d'abstraction.

Cette directive est modélisée par la carte présentée à la Figure 59. Cette carte (*la carte C3*) contient une seule intention centrale *Définir un élément de produit.* La réalisation de cette intention est assurée par un ensemble de stratégies, deux de ces stratégies appelées *Abstraction dirigée par le produit* et *Abstraction dirigée par le processus* sont utilisées pour le début du processus d'évolution, la première stratégie est en relation avec le modèle de produit *As-Is*, la deuxième stratégie est en relation avec le modèle de processus *As-Is*.

Dans le méta-modèle de produit présenté au chapitre 3 nous avons défini deux manières orthogonales pour classer un *élément de produit*. La première classification fait la distinction entre *élément de produit composé* et *élément de produit atomique*. La deuxième classification différencie les éléments *Liens* et *Non-Liens*. Dans cette directive nous désignons par élément de produit une *classe* du modèle de produit ou un *lien* (d'association, d'agrégation ou d'héritage) entre les classes du modèle.

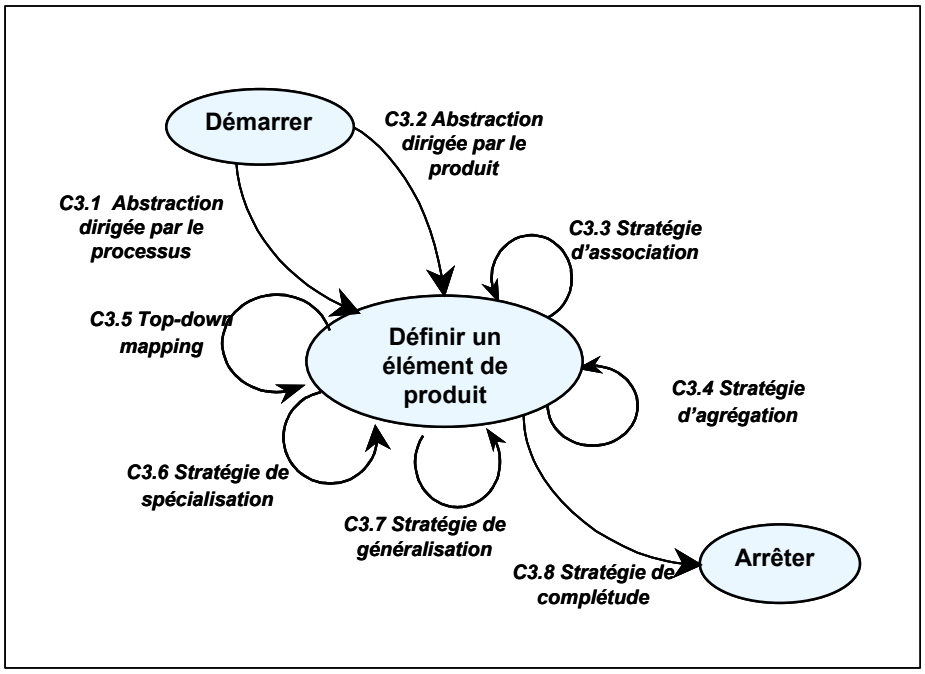


Figure 59. La carte C3 (DRI_3)

La stratégie *Abstraction dirigée par le produit* consiste à analyser le modèle de produit *As-Is* dans le but d'identifier les éléments qui peuvent être représentés par des concepts d'un niveau d'abstraction plus élevé. Les éléments identifiés seront définis et ajoutés à la nouvelle couche du modèle de produit en cours de construction.

La stratégie *Abstraction dirigée par le processus* propose d'analyser le modèle de processus *As-Is* et d'effectuer, si nécessaire, une abstraction de certaines de ses activités. Les éléments de produit correspondant aux nouvelles activités obtenues sont définis et ajoutés à la nouvelle couche du modèle de produit en cours de construction.

La stratégie *Top-down mapping* est utilisée pour modéliser le lien entre les éléments des deux couches du modèle de produit *To-Be*. Les stratégies de *généralisation*, *spécialisation* et *agrégation* sont utilisées pour compléter et affiner le modèle de produit en cours de construction. La *Stratégie d'association* est utilisée pour relier les différents éléments de ce modèle.

La *Stratégie de complétude* permet d'arrêter le processus d'évolution du modèle de produit par abstraction.

Dans ce qui suit on va présenter les DSI, les DSS et le tableau des DRI associées à la carte C3. Les DRI sont présentées en détail dans la section 4.3.1 de ce chapitre.

4.2.3.1. Directives de Sélection d'Intention associées à la carte C3

Une seule DSI, la $DSI_{3.1}$ est associée à l'intention *Définir un élément de produit*.

Intention	DSI
I_1 : Démarrer	N/A
I_2 : Définir une élément de produit	$DSI_{3.1}$: <(MPdt 'en évolution'), Progresser depuis Définir un élément de produit >
I_3 : Arrêter	N/A

Tableau 9. DSI de la carte C3

4.2.3.1.1. Progresser depuis « Définir un élément de produit » ($DSI_{3.1}$)

La $DSI_{3.1}$ est de type choix. Elle permet de progresser à partir de l'intention *Définir un élément de produit* en proposant d'atteindre une intention cible. Elle propose de réaliser soit l'intention *Définir un élément de produit* (boucle) soit l'intention *Arrêter*. Comme le montre la Figure 60, pour réaliser le premier choix, on sélectionne la $DSS_{3.2}$ qui propose cinq stratégies (*Stratégie d'association*, *Stratégie d'agrégation*, *Stratégie de généralisation*, *Stratégie de spécialisation*, *Top-down mapping*). Pour faire le deuxième choix, on sélectionne la $DRI_{3.8}$ qui correspond à la *Stratégie de complétude* pour arrêter le processus d'évolution du modèle de produit. La Figure 60 présente aussi les arguments qui dirigent le choix de chacune des deux possibilités.

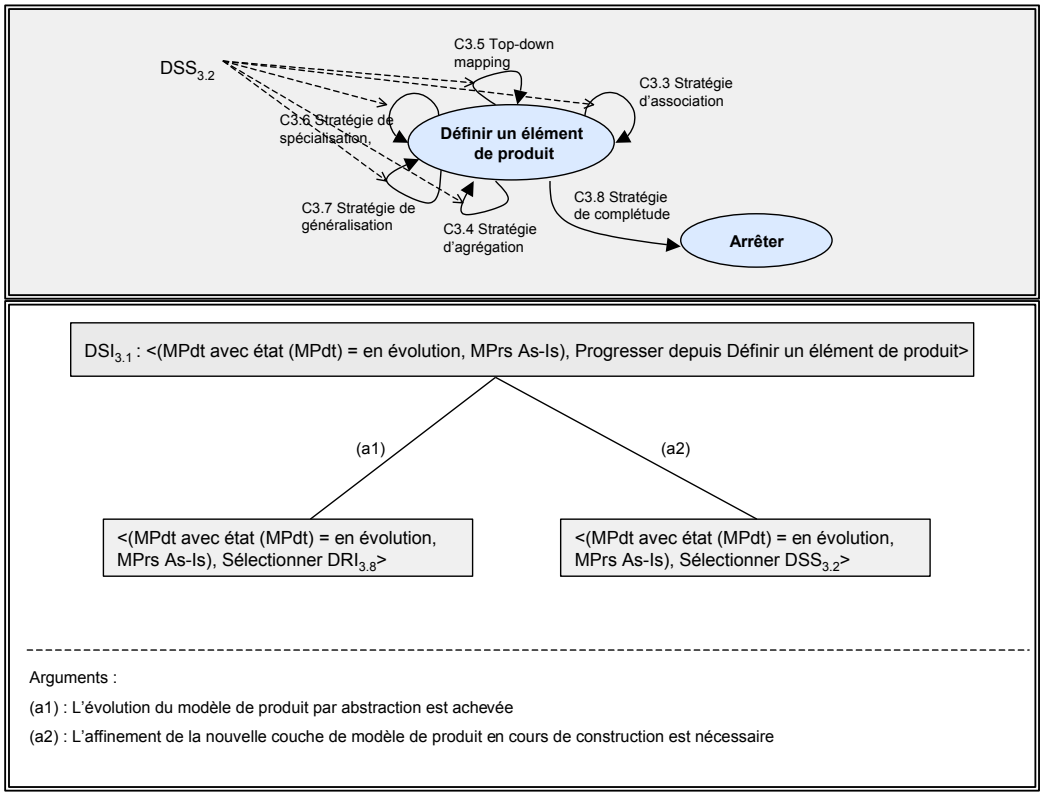


Figure 60. La structure de la $DSI_{3.1}$

4.2.3.2. Directives de Sélection de Stratégie associées à la carte C3

Deux DSS sont associée à la carte C3 :

Paire d'intentions	DSS
($I_1 – I_2$) : (Démarrer, Définir un élément de produit)	$DSS_{3.1}$: <(MPdt As-Is, MPrs As-Is), Progresser vers Définir un élément de produit>
($I_2 – I_2$) : (Définir un élément de produit, Définir un élément de produit)	$DSS_{3.2}$: <(MPdt 'en évolution'), Progresser vers Définir un élément de produit >

Tableau 10. Les DSI de la carte C3

4.2.3.2.1. Progresser vers « Définir un élément de produit » ($DSS_{3.1}$)

La $DSS_{3.1}$ permet de guider l'ingénieur de méthodes pour le choix de stratégies pour progresser de l'intention *Démarrer* vers l'intention *Définir un élément de produit* de la carte C3. La Figure 61 présente cette directive proposant deux choix : (1) Sélectionner la $DRI_{3.1}$ correspondant à la stratégie *Abstraction dirigée par le processus,* ou (2) Sélectionner la $DRI_{3.2}$ correspondant à la stratégie *Abstraction dirigée par le produit*. Deux arguments (a1) et (a2) assistent l'ingénieur de méthodes pour opter pour l'un des choix.

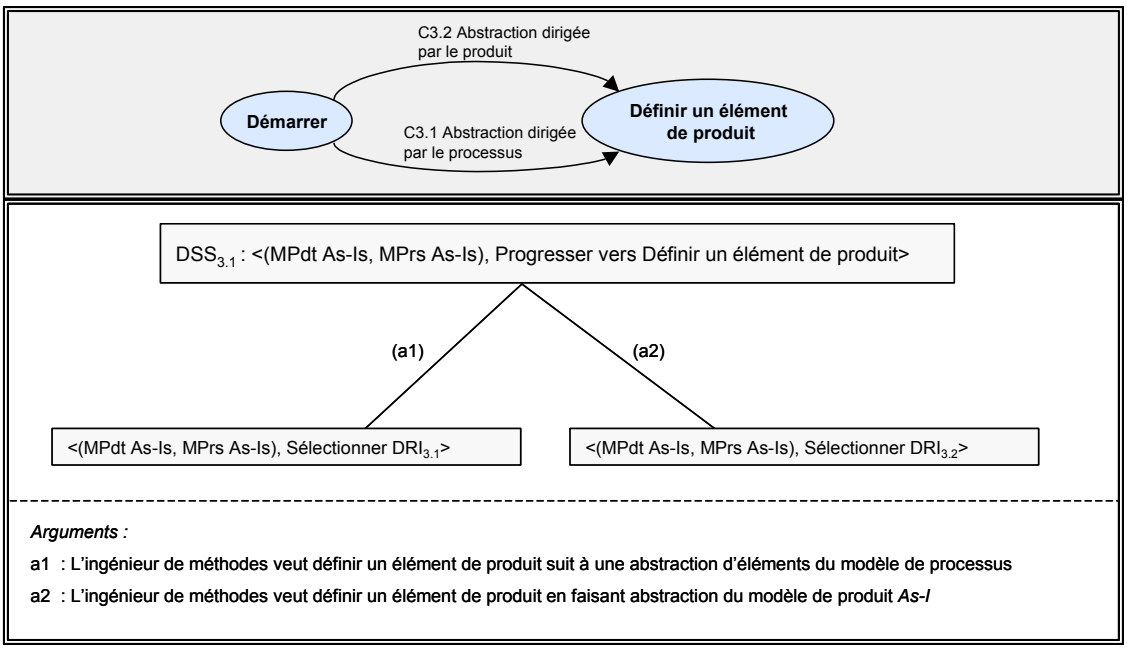


Figure 61. Structure de la $DSS_{3.1}$

4.2.3.2.2. Progresser vers « Définir un élément de produit » ($DSS_{3.2}$)

La $DSS_{3.2}$ est de type choix, elle permet de progresser à partir de l'intention *Définir un élément de produit* vers l'intention *Définir un élément de produit*, c'est-à-dire qu'elle permet de boucler sur l'intention elle-même. Elle offre cinq choix : sélectionner la directive $DRI_{3.3}$ relative à la *Stratégie d'association*, sélectionner la directive $DRI_{3.4}$ associée à la *Stratégie d'agrégation*, sélectionner la $DRI_{3.5}$ associée à la stratégie *Top-down mapping,* sélectionner la $DRI_{3.6}$ associée à la *Stratégie de*

spécialisation, sélectionner la $DRI_{3.7}$ associée à la *Stratégie de généralisation*. La Figure 62 montre la structure ce cette DSS ainsi que les arguments pour le choix d'une stratégie plutôt que l'autre.

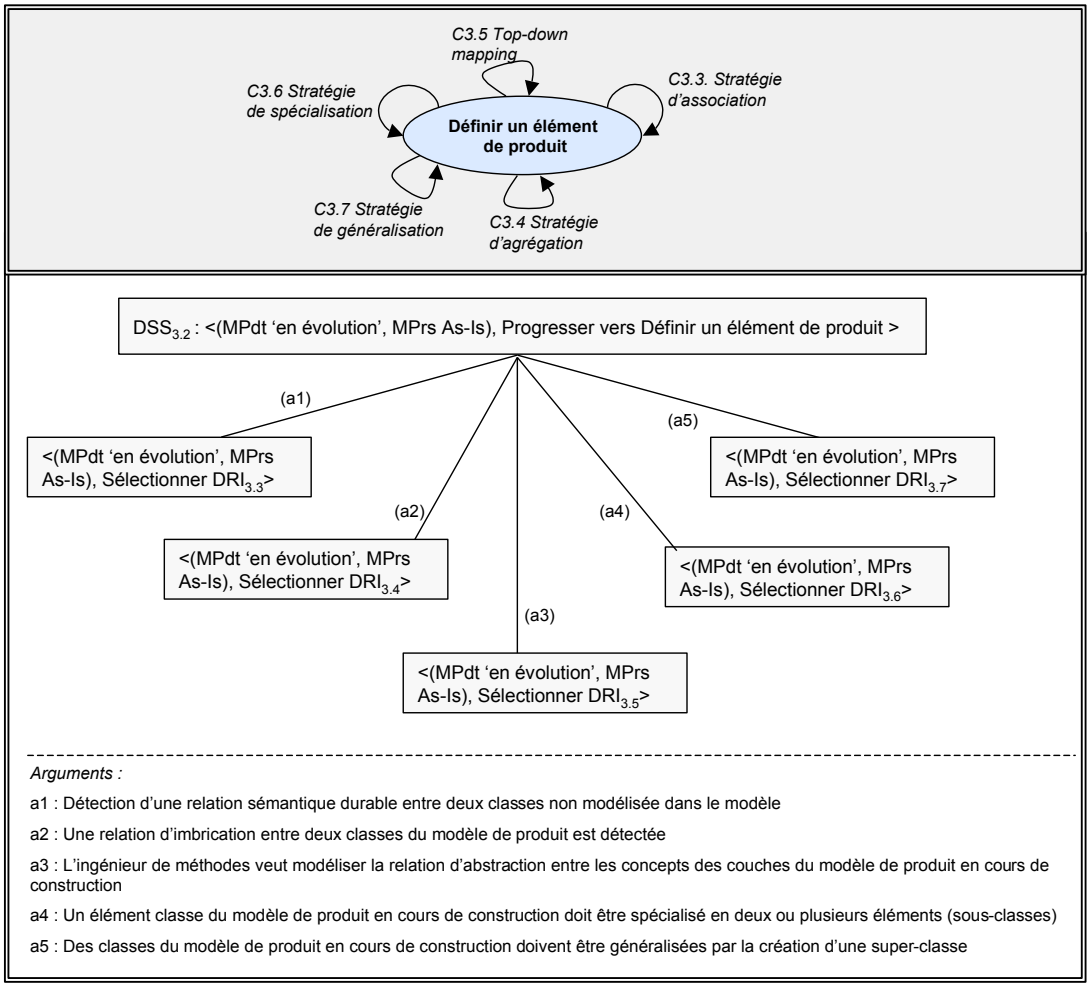


Figure 62. Structure de la $DSS_{3.2}$

4.2.3.3. Tableau des DRI associées à la carte C3

La carte C3 Comprend 8 sections. A chaque section est affectée une DRI. Les huit DRI associées à cette carte sont présentées dans le Tableau 11.

Identifiant	Interface de DRI	Type
$DRI_{3.1}$	< (MPdt As-Is, MPrs As-Is), Définir un élément de produit avec la stratégie abstraction dirigée par le processus>	Tactique
$DRI_{3.2}$	< (MPdt As-Is), Définir un élément de produit avec la stratégie abstraction dirigée par le produit>	Tactique
$DRI_{3.3}$	< (MPdt 'en évolution'), Définir un élément de produit avec la stratégie d'association >	Tactique
$DRI_{3.4}$	< (MPdt 'en évolution'), Définir un élément de produit avec la stratégie d'agrégation>	Tactique
$DRI_{3.5}$	< (MPdt 'en évolution'), Définir un élément de produit avec la stratégie Top-down mapping >	Tactique

$DRI_{3.6}$	< (MPdt 'en évolution'), Définir un élément de produit avec la stratégie de spécialisation>	Tactique
$DRI_{3.7}$	< (MPdt 'en évolution'), Définir un élément de produit avec la stratégie de généralisation >	Tactique
$DRI_{3.8}$	< (MPdt 'en évolution'), Arrêter avec la stratégie de complétude>	Informelle

Tableau 11. Les DRI de la carte C3

4.2.4. Réaliser « Faire évoluer le modèle de produit Par adaptation de modèle » (DRI_4)

La DRI_4 est associée à la section <*Spécifier les exigences d'évolution, Faire évoluer le modèle de produit, Par adaptation de modèle*> de la carte d'évolution. En sélectionnant cette directive, le modèle de produit *To-Be* est obtenu par adaptation du modèle de produit *As-Is* en appliquant des opérateurs d'adaptation.

Pour la définition de ces opérateurs on s'est basé sur le méta-modèle de produit présenté au chapitre 3. Selon ce méta-modèle, tout modèle de produit est composé d'éléments. Deux manières orthogonales permettent de classer ces éléments. La première classification fait la distinction entre éléments *composé* et *atomique*, la deuxième classification distingue les éléments *Lien* et *Non-lien*.

Sur la base de cette typologie, nous proposons quatre ensembles d'opérateurs pour l'adaptation de modèle selon la nature de l'élément qui va subir l'adaptation. Le Tableau 12 récapitule les opérateurs applicables à chaque type d'élément (*Elément atomique*, *Elément composé*, *Propriété* ou *Elément lien)*.

Objet	Opérateur	Description
Elément atomique	*Renommer*	Changer le nom d'un élément
	Ajouter	Ajouter un nouvel élément dans le modèle
	Supprimer	Supprimer un élément du modèle
	Fusionner	Fusionner deux ou plusieurs éléments distincts en un seul
	Fractionner	Fractionner un élément du modèle en deux ou plusieurs éléments
	Remplacer	Remplacer un élément du modèle par un autre différent
	Généraliser	Un élément est crée comme généralisation de deux ou plusieurs éléments
	Spécialiser	Un élément est spécialisé en deux ou plusieurs sous-éléments
	Retyper	Changer le type d'un élément
Elément composé	*Ajouter composant*	Ajouter un composant à un élément composé
	Supprimer composant	Supprimer un composant d'un élément composé
	Déplacer composant	Un composant est repositionner dans la structure d'un élément composé

Propriété	*Attribuer*	Ajouter une propriété à un élément
	Retirer	Supprimer une propriété d'un élément
	Modifier	Modifier une propriété d'un élément
Lien	*Changer(source, cible)*	Changer la source et ou la cible d'un élément lien

Tableau 12. Opérateurs d'adaptation de modèles

La DRI_4 guide l'ingénieur de méthodes dans l'application des opérateurs présentée ci-dessus. La Figure 63 présente cette directive.

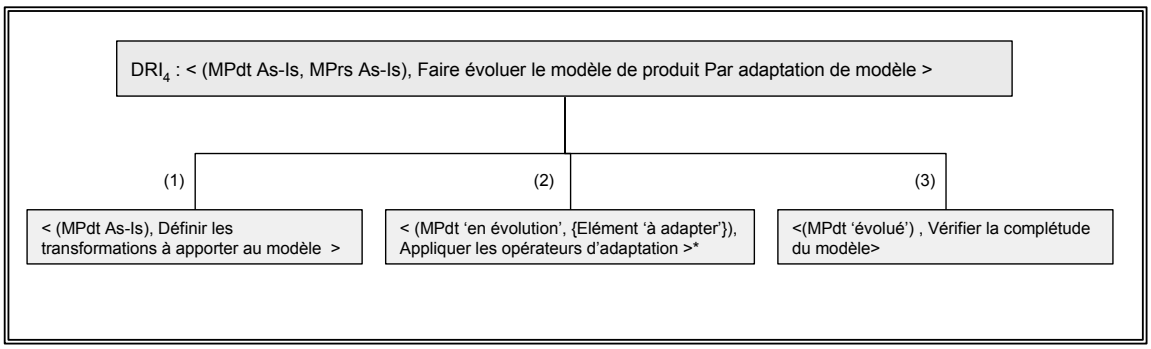


Figure 63. Structure de la DRI_4

Comme le montre cette figure, La DRI_4 est de type plan décomposé comme suit :

- *< (MPdt As-Is), Définir les transformations à apporter au modèle >:* Cette directive consiste à définir, en se basant sur les exigences capturées en réalisant l'intention *Spécifier les exigences d'évolution*, la liste des transformations à apporter au modèle de produit *As-Is*. Ces transformations doivent indiquer les éléments du modèle à modifier et la nature de la modification à opérer

- *< (MPdt 'en évolution', {Elément 'à adapter'}), Appliquer les opérateurs d'adaptation > :* Cette directive a pour objectif d'assister l'ingénieur de méthode dans l'application des opérateurs d'adaptations aux éléments identifiés par l'application de la directive précédente. La Figure 64 présente cette directive de type plan, elle propose à l'ingénieur de méthodes de : (2.1) Identifier l'opérateur d'adaptation adéquat selon la nature de l'élément et le type de la transformation à réaliser, (2.1) Appliquer l'opérateur sélectionné sur l'élément du modèle.

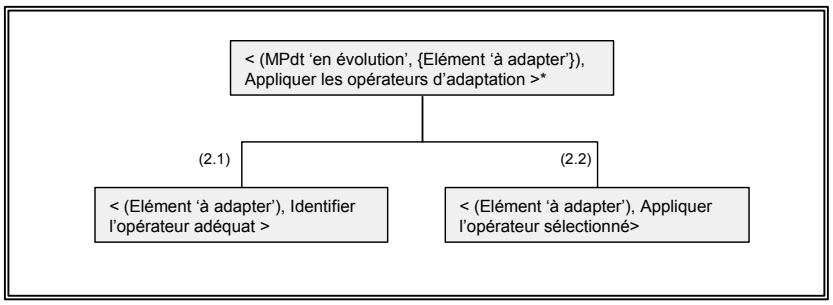


Figure 64. < (MPdt 'en évolution', {Elément 'à adapter'}), Appliquer opérateur d'adaptation >

- *<(MPdt 'évolué'), Vérifier la complétude du modèle>* : Cette directive suggère à l'ingénieur de méthodes de compléter l'évolution du modèle de produit en s'assurant de la cohérence et de la complétude du modèle suite aux transformations apportées par l'exécution de la directive précédente.

4.2.5. Réaliser « Faire évoluer le modèle de produit Par instanciation de méta-modèle » (DRI_5)

La DRI_5 est associée à la section C5 de la carte d'evolution. L'objectif de cette directive est de guider l'ingénieur de méthodes dans la construction du modèle de produit *To-Be* en utilisant la technique d'instanciation de méta-modèle.

En appliquant cette directive, le modèle de produit *To-Be* est obtenu par instanciation d'un méta-modèle existant. Le méta-modèle de produit présenté au chapitre 3 est recommandé mais d'autres méta-modèles disponibles dans la littérature [Grundy 96], [Hofstede 93], [Prakash 02], [Saeki 94] peuvent également être utilisés. L'adoption de l'un ou l'autre des méta-modèles est guidé par la nature du produit que l'on souhaite construire par l'application de la méthode et par les exigences d'évolution exprimées au début du projet d'évolution.

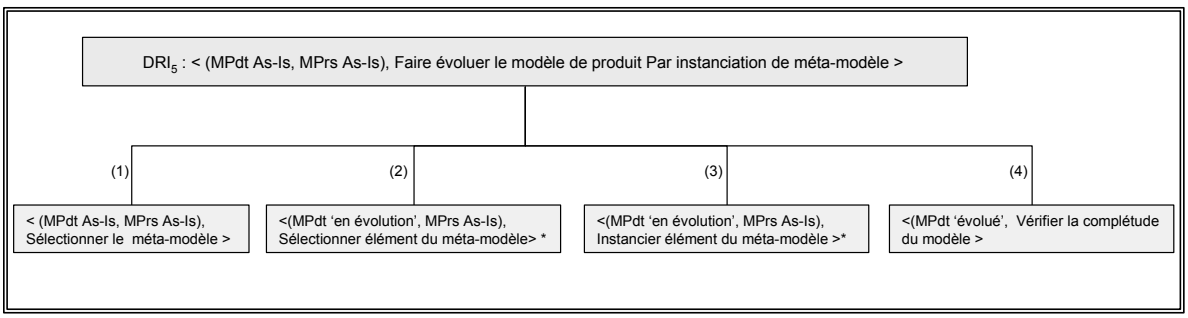


Figure 65. Structure de la DRI_5

La directive DRI_5, présenté à la Figure 65, est de type plan. Ce plan est décomposé comme suit :

(1) < *(MPdt As-Is, MPrs As-Is), Sélectionner le méta-modèle* > : L'objectif de cette directive est de sélectionner le méta-modèle de produit à instancier. Le choix du méta-modèle est guidé par la nature du produit qu'on souhaite obtenir par l'application de la méthode.

(2) < *(MPdt 'en évolution', MPrs As-Is), Sélectionner élément du méta-modèle*> : Cette directive suggère à l'ingénieur de méthodes de sélectionner un élément du méta-modèle à instancier pour modéliser un concept de la méthode.

(3) <*(MPdt 'en évolution', MPrs As-Is), Instancier élément du méta-modèle* > : L'application de cette directive permet d'instancier l'élément du méta-modèle sélectionné par la directive précédente. Cette directive et la précédente seront exécutées n fois jusqu'à la construction complète du modèle de produit *To-Be*.

(4) < *(MPdt 'évolué', Vérifier la complétude du modèle* > : Cette directive suggère à l'ingénieur de méthodes de compléter et d'affiner le modèle de produit obtenu par évolution en ajoutant d'éventuels liens d'association, d'héritage ou d'abstraction entre les éléments de ce modèle.

4.2.6. Réaliser « Faire évoluer le modèle de produit Par validation » (DRI_6)

La DRI_6 est associée à la section C6 de la carte d'évolution. L'objectif de cette directive est de guider l'ingénieur de méthodes dans la validation du modèle de produit obtenu par évolution.

La DRI_6 est de type plan, ce plan propose 3 sous-directives exécutables correspondant aux trois étapes de validation à appliquer au modèle de produit. La Figure 66 présente la structure de cette directive.

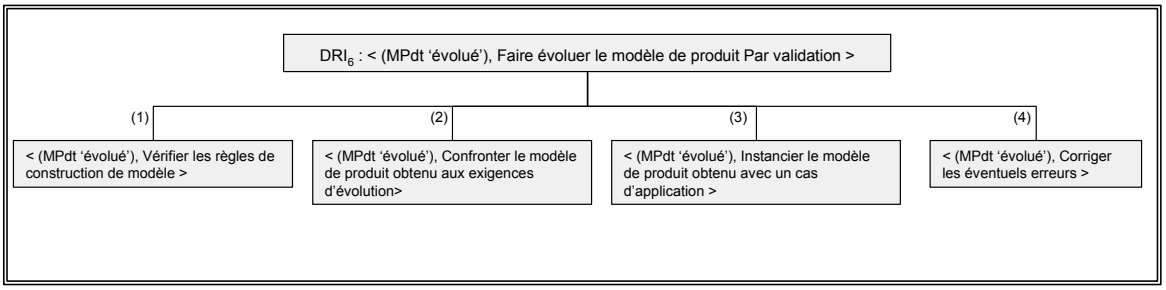


Figure 66. Structure de la DRI_6

(1) < *(MPdt 'évolué'), Vérifier les règles de construction de modèle* > : Cette directive suggère à l'ingénieur de méthodes de s'assurer que le modèle de produit obtenu respecte les règles de construction de modèles définies dans le chapitre 3.

(2) < *(MPdt 'évolué'), Confronter le modèle de produit obtenu aux exigences d'évolution*> : La deuxième étape de validation proposée par cette directive, consiste à s'assurer que le modèle

de produit obtenu satisfait toutes les exigences d'évolution exprimées au début du projet d'ingénierie.

(3) *<(MPdt 'évolué'), Instancier le modèle de produit obtenu avec un cas d'étude >* : Cette directive propose d'appliquer un troisième type de validation consistant à instancier le modèle de produit obtenu avec un cas d'application.

(4) *< (MPdt 'évolué'), Corriger les éventuels erreurs >* : Cette directive suggère à l'ingénieur de méthodes de corriger les éventuelles erreurs détectées dans le modèle, par l'application des trois directives précédentes.

4.2.7. Réaliser « Faire évoluer le modèle de processus Par adaptation de modèle » (DRI_7)

La DRI_7 est associée à la section C7 de la carte d'évolution. Elle a pour objectif de faire évoluer le modèle de processus As-Is en lui apportant un ensemble d'adaptations.

Cette directive est similaire à la directive DRI_4, elle a la même structure et elle se base sur les mêmes opérateurs d'adaptation de modèles que ceux présentés précédemment.

4.2.8. Réaliser « Faire évoluer le modèle de processus avec la stratégie orientée-activité » (DRI_8)

La DRI_8 est associée à la stratégie *Orientée-activité* pour l'évolution du modèle de processus. Cette directive offre un guidage pour l'évolution du modèle de processus existant de la méthode vers un modèle de processus utilisant le paradigme de modélisation *Orienté activité*. Ce paradigme de modélisation est adapté pour présenter des processus opérationnels dont l'objectif est de planifier les étapes à suivre pour construire le produit. Le modèle de processus *To-Be* est alors construit comme un processus séquentiel composé d'une succession d'activités permettant d'atteindre les objectifs définis pour la méthode *To-Be*. La Figure 67 présente la structure de la DRI_8 qui est une directive tactique de type plan.

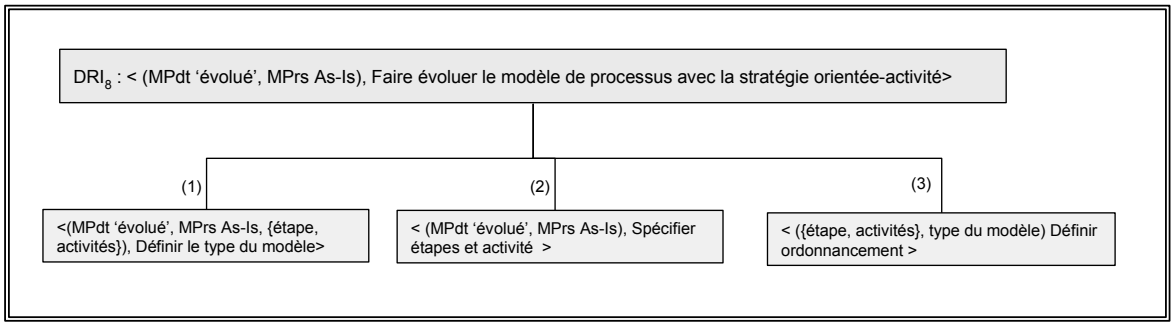


Figure 67. Structure de la DRI8

(1) < *(MPdt 'évolué', MPrs As-Is, {étape, activités}), Définir le type du modèle* > : Cette directive propose à l'ingénieur de méthodes de choisir le type du modèle de processus orienté-activité qu'il souhaite définir. Le type du modèle peut être aussi défini par les exigences d'évolution. Plusieurs types de modèles proposant des manières différentes de définir l'ordonnancement des activités du modèle de processus sont disponibles parmi les quels on peut citer : le modèle en *Cascade* [Royce 70], le modèle en *Spirale* [Boehm 90], Le modèle *Hiérarchique en spirale* [Iivari 90] ou le modèle en *Fontaine* [Henderson-Sellers 90].

(2) < *(MPdt 'évolué', MPrs As-Is), Spécifier étapes et activité* > : Cette directive suggère à l'ingénieur de méthodes d'utiliser la technique de décomposition fonctionnelle pour définir les étapes et les activités composant le modèle de processus To-Be. Cette décomposition se base sur les buts fixés pour la méthode *To-Be* au moment de la spécification des exigences de l'évolution et sur la description plus ou moins structurée du modèle de processus *As-Is*.

(3) < *({étape, activités}, type du modèle), Définir ordonnancement* > : Cette directive suggère à l'ingénieur de méthodes de définir l'ordonnancement des étapes et des activités définies par la directive précédente en utilisant le type du modèle de processus sélectionné.

4.2.9. Réaliser « Faire évoluer le modèle de processus avec la stratégie orientée-stratégie » (DRI9)

La DRI9 est associée à la section <*Faire évoluer le modèle de produit, Faire évoluer le modèle de processus, Orientée-stratégie*> de la carte d'évolution. Le but de cette directive est de faire évoluer un modèle de processus *As-Is* en réalisant sa ré-ingénierie par l'utilisation du paradigme de modélisation *orienté-stratégie* permettant ainsi d'avoir un modèle de processus intentionnel et multi-démarche pour la construction du produit. Le modèle de processus *To-Be* sera alors formalisé par une *carte* et un ensemble de directives.

La Figure 68 présente la *carte C9* modélisant la directive DRI9. Cette directive, inspirée de la méta-démarche proposée par A.Benjamen [Benjamen 99], est composée de deux intentions principales

Définir une section et *Définir une directive*. La première vise la découverte et la définition d'une section composée du triplet <*Intention source, Intention cible, Stratégie*>. La deuxième a comme objectif d'aider l'ingénieur de méthode dans la définition des directives associées aux sections.

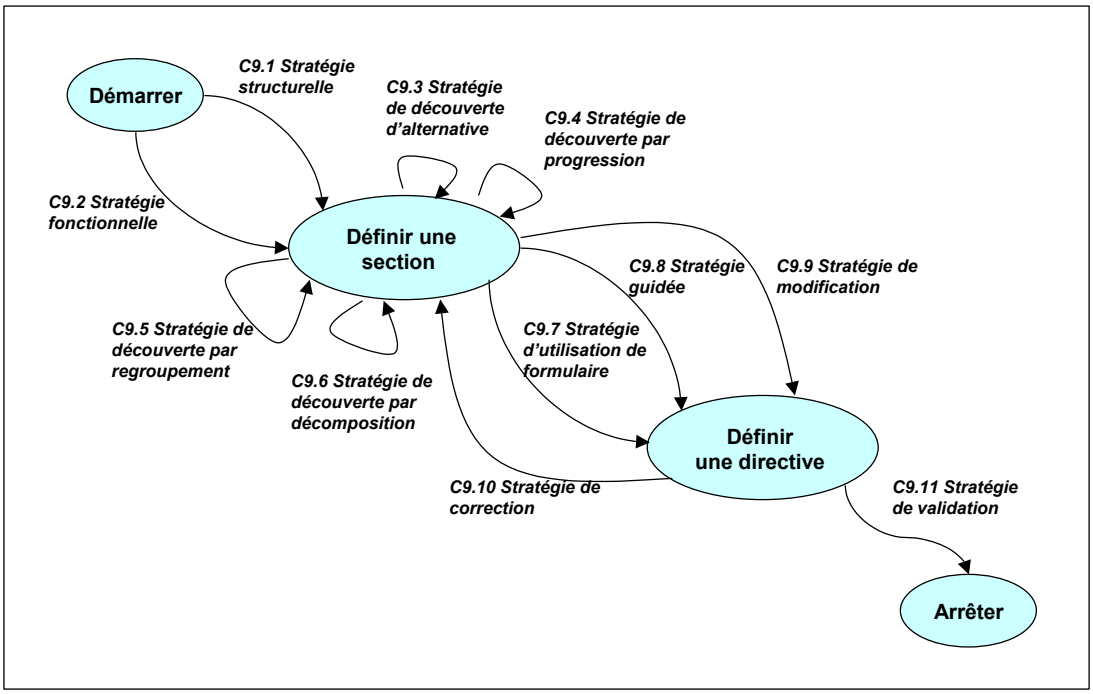


Figure 68. La carte C9 (DRI_9)

Deux stratégies alternatives permettent d'atteindre l'intention *Définir une section* : la *Stratégie structurelle* et la *Stratégie fonctionnelle*. La première est utilisée dans le cas où la méthode *As-Is* ne dispose pas initialement de modèle de processus ou si celui-ci se limite à une description informelle du produit à construire. La deuxième stratégie est utilisée dans le cas où la méthode *As-Is* possède un modèle de processus plus formalisé, composé d'un certain nombre d'étapes et de fonctions à réaliser et proposant différents moyens et/ou différentes manières pour les effectuer afin d'obtenir le produit. Cette stratégie fournit un guidage à l'ingénieur de méthodes pour l'identification des sections composant le modèle de processus *To-Be* en se basant sur ces étapes, fonctions, moyens et manières et sur les exigences d'évolution exprimés au début du processus d'évolution.

Quatre stratégies sont proposées pour la découverte de sections à partir de celles déjà définies : (1) la *Stratégie de découverte d'alternative* qui offre un guidage pour déterminer à partir d'une section, de nouvelles sections alternatives en terme d'intention et de stratégie, (2) la *Stratégie de découverte par progression* qui propose de trouver de nouvelles sections permettant de continuer dans la démarche à partir d'une section existante, (3) la *Stratégie de découverte par regroupement* qui consiste à regrouper un ensemble de sections en une seule et enfin (4), la *Stratégie de découverte par décomposition* qui consiste à décomposer une section en plusieurs sections.

Pour atteindre l'intention *Définir une directive* deux stratégies sont fournies, *Stratégie d'utilisation de formulaire* et *Stratégie guidée*. La *Stratégie d'utilisation de formulaire* propose d'utiliser des formulaires prédéfinis selon le type de la directive que l'ingénieur de méthodes veut définir pour la construction du modèle de processus *To-Be*. La *Stratégie guidée* propose d'aider l'ingénieur à trouver le type de directive à utiliser et à construire la directive concernée.

Les deux stratégies : *Stratégie de modification* et *Stratégie de correction* entre les deux intentions *Définir une section* et *Définir une directive* offrent un guidage à l'ingénieur de méthodes pour affiner et corriger le modèle de processus en cours d'évolution. En effet, la première stratégie permet de répercuter les modifications apportées aux sections sur les directives associées, par contre la deuxième stratégie permet de corriger une section sur la base de la description de la directive associée.

4.2.9.1. Directives de Sélection d'intention associées à la carte C9

Deux DSI ($DSI_{9.1}$ et $DSI_{9.2}$) sont associée à la carte C9, le Tableau 13 présente ces DSI.

Intention	DSI
I_1 : Démarrer	N/A
I_2 : Définir une section	$DSI_{9.1}$: < (MPdt 'évolué', MPrs As-Is, {Section 'définie'}), Progresser depuis Définir une section >
I_3 : Définir une directive	$DSI_{9.2}$: < (MPdt 'évolué', {Section 'définie'}, {Directive 'définie'}), Progresser depuis Définir une directive >
I_4 : Arrêter	N/A

Tableau 13. Les DSI de la carte C9

4.2.9.1.1. Progreser depuis « Définir une section » ($DSI_{9.1}$)

La $DSI_{9.1}$ est de type choix. Elle permet de progresser à partir de l'intention *Définir une section* en proposant d'atteindre une intention cible. Elle propose de réaliser soit l'intention *Définir une section* (boucle) soit l'intention *Définir une directive*. Comme le montre la Figure 69, pour réaliser le premier choix, on sélectionne la $DSS_{9.2}$ qui propose quatre stratégies (*Stratégie de regroupement*, *Stratégie de décomposition*, *Stratégie de progression et Stratégie d'alternative*). Pour faire le deuxième choix, on sélectionne la $DSS_{9.3}$ qui propose trois stratégies (*Stratégie d'utilisation de formulaire*, *Stratégie guidée* et *Stratégie de modification*). La Figure 69 montre aussi les arguments qui dirigent le choix de chacune des deux possibilités.

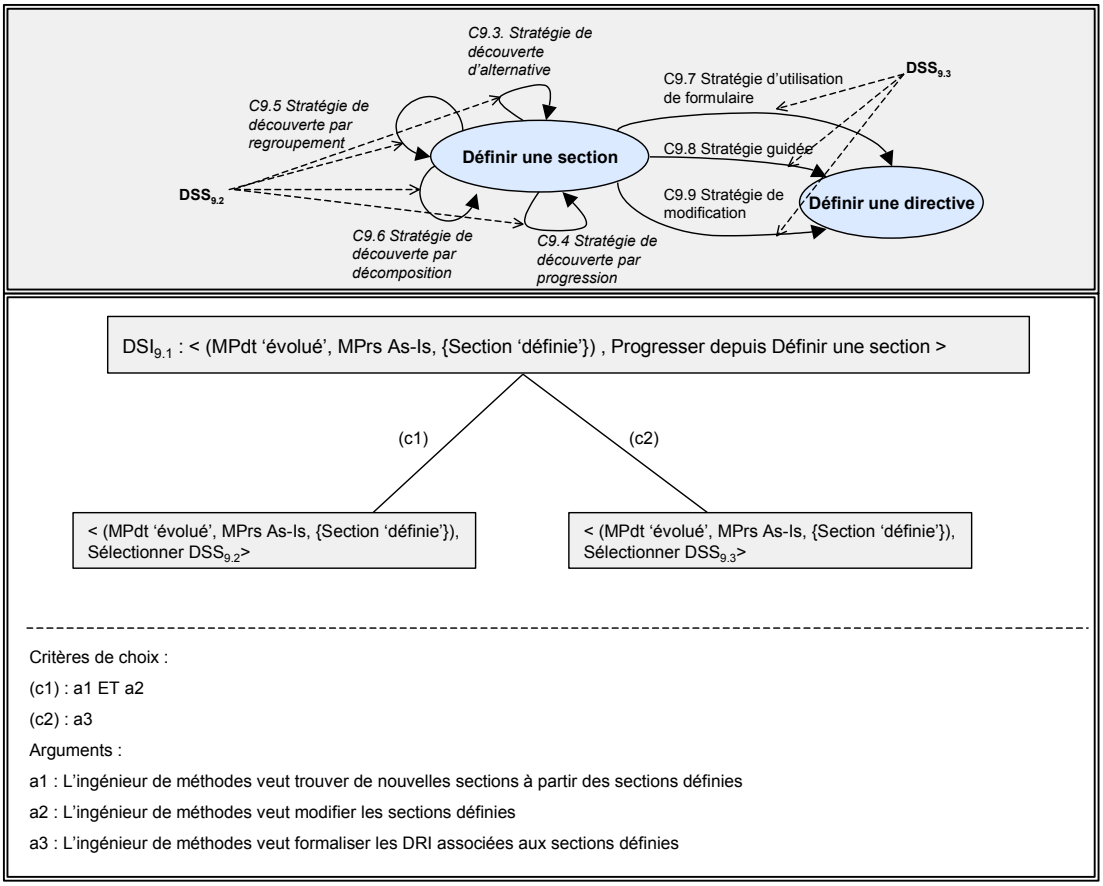


Figure 69. Structure de la $DSI_{9.1}$

4.2.9.1.2. Progresser depuis « Définir une directive » ($DSI_{9.2}$)

Lorsque l'ingénieur de méthodes a défini une directive de la carte de méthode, il peut soit corriger les sections définies si la description des directives a mis des erreurs en évidence soit, arrêter le processus en sélectionnant la stratégie de complétude. La $DSI_{9.2}$ décrite à la Figure 70 guide l'ingénieur de méthodes dans ce choix. Pour réaliser le premier choix on sélectionne la $DRI_{9.10}$ correspondante à la stratégie de correction. Pour réaliser le deuxième choix on sélectionne la $DRI_{9.11}$ correspondante à la stratégie de complétude.

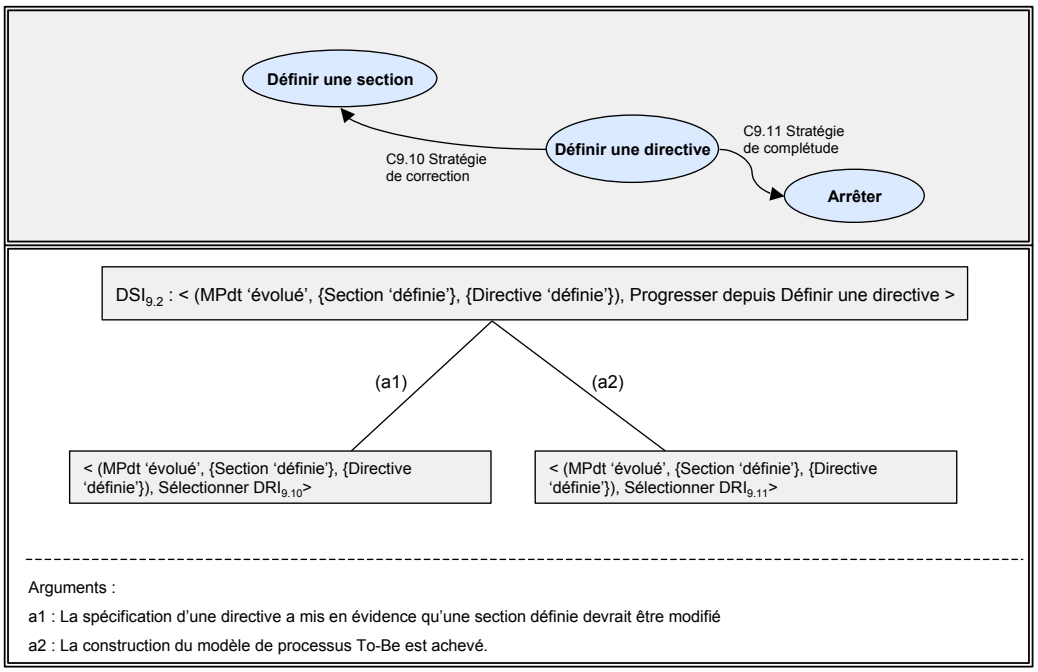


Figure 70. Structure de la $DSI_{9.2}$

4.2.9.2. Directives de Sélection de Stratégie associées à la carte C9

Le Tableau 14 présente les DSS associées à la carte C9.

Paire d'intentions	DSS
$(I_1 - I_2)$: (Démarrer, Définir une section)	$DSS_{9.1}$: < (MPdt 'évolué', MPrs As-Is), Progresser vers Définir une section>
$(I_2 – I_2)$: (Définir une section, Définir une section)	$DSS_{9.2}$: < (MPdt 'évolué', MPrs As-Is, {Section 'définie'}), Progresser vers Définir une section >
$(I_2 – I_3)$: (Définir une section, Définir une directive)	$DSS_{9.3}$: <(MPdt 'évolué', MPrs As-Is, {Section 'définie'}), Progresser vers Définir une directive>

Tableau 14. Les DSS de la carte C9

4.2.9.2.1. Progresser vers « Définir une section » ($DSS_{9.1}$)

La $DSS_{9.1}$ aide l'ingénieur de méthodes à démarrer le processus d'évolution du modèle de processus en lui proposant deux possibilités : la *stratégie structurelle* et la *stratégie fonctionnelle*. Comme le montre la Figure 71, le choix de la stratégie dépend en partie de la nature du modèle de processus *As-Is*. Si ce modèle est décrit de manière informelle, l'ingénieur de méthodes est invité à sélectionner la $DRI_{9.1}$ correspondante à la *stratégie structurelle*. Par contre, si le modèle de processus *As-Is* est décrit de manière plus ou moins structurée, l'ingénieur de méthodes peut choisir la stratégie fonctionnelle en sélectionnant la $DRI_{9.2}$.

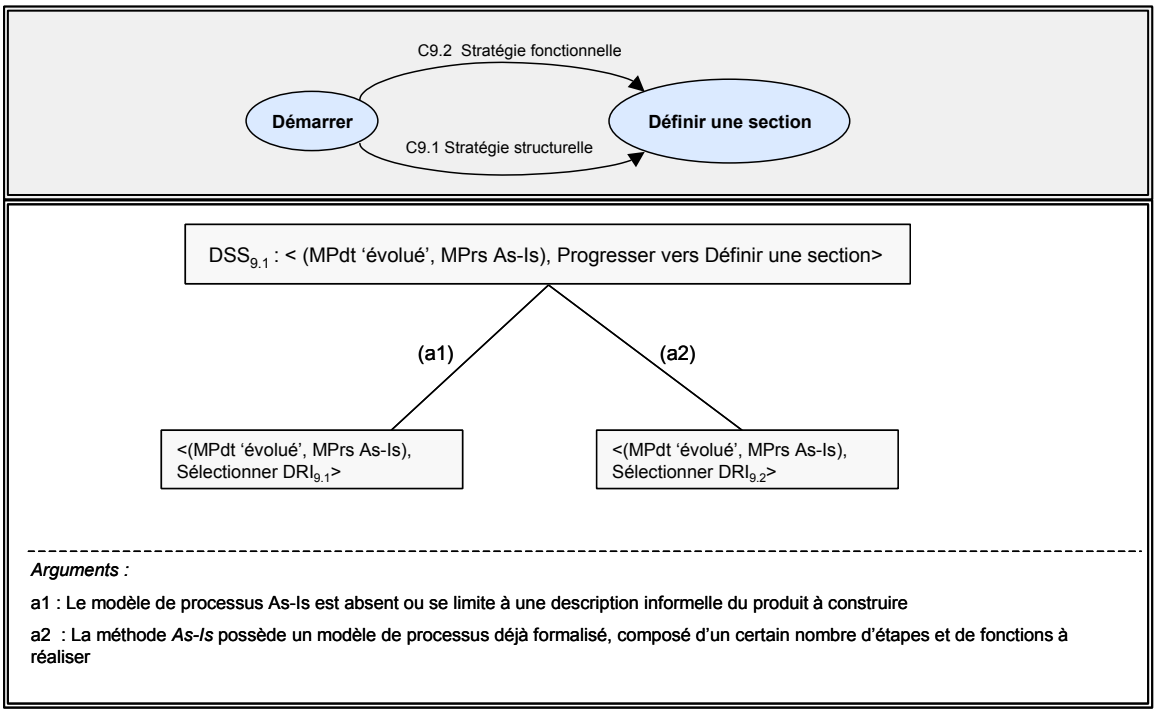


Figure 71. Structure de la $DSS_{9.1}$

4.2.9.2.2. Progresser vers « Définir une section » ($DSS_{9.2}$)

La $DSS_{9.2}$ est de type choix, elle assiste l'ingénieur de méthodes dans la sélection de stratégie pour progresser à partir de l'intention *Définir une section* vers l'intention *Définir une section* (boucle). Elle offre quatre stratégies pour réaliser cela : *Stratégie de découverte d'alternative, Stratégie de découverte par progression, Stratégie de découverte par décomposition* et *Stratégie de découverte par regroupement*. La Figure 72 présente la structure de la directive $DSS_{11.2}$ et les arguments en faveur de chaque choix.

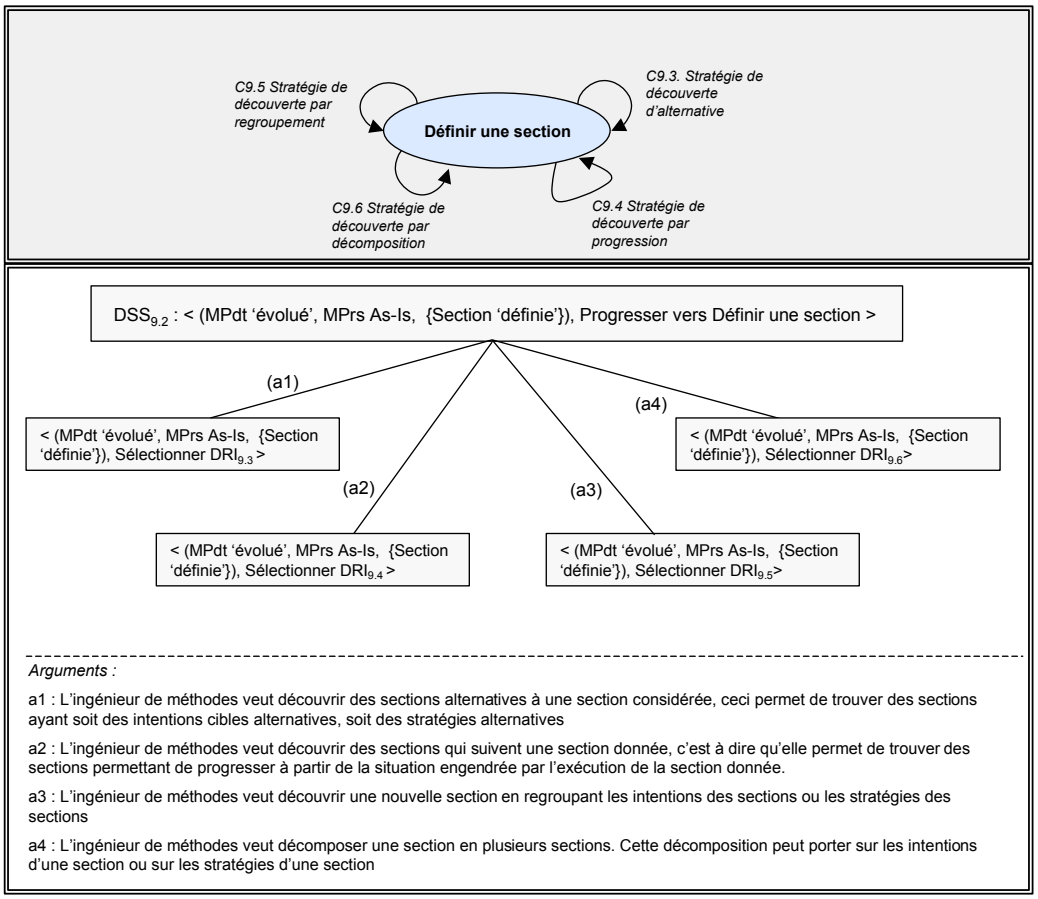


Figure 72. Structure de la $DSS_{9.2}$

4.2.9.2.3. Progresser vers « Définir une directive » ($DSS_{9.3}$)

La directive $DSS_{9.3}$ guide l'ingénieur de méthode dans le choix de stratégie pour atteindre l'intention *Définir une directive* à partir de l'intention *Définir une section*. Comme le montre la Figure 73 la $DSS_{9.3}$ propose trois choix : (1) Sélectionner la directive $DRI_{9.7}$ correspondante à la *Stratégie d'utilisation de formulaire*, (2) Sélectionner la directive $DRI_{9.8}$ correspondante à la *Stratégie guidée*, (3) Sélectionner la directive $DRI_{9.9}$ correspondante à la *Stratégie de modification*.

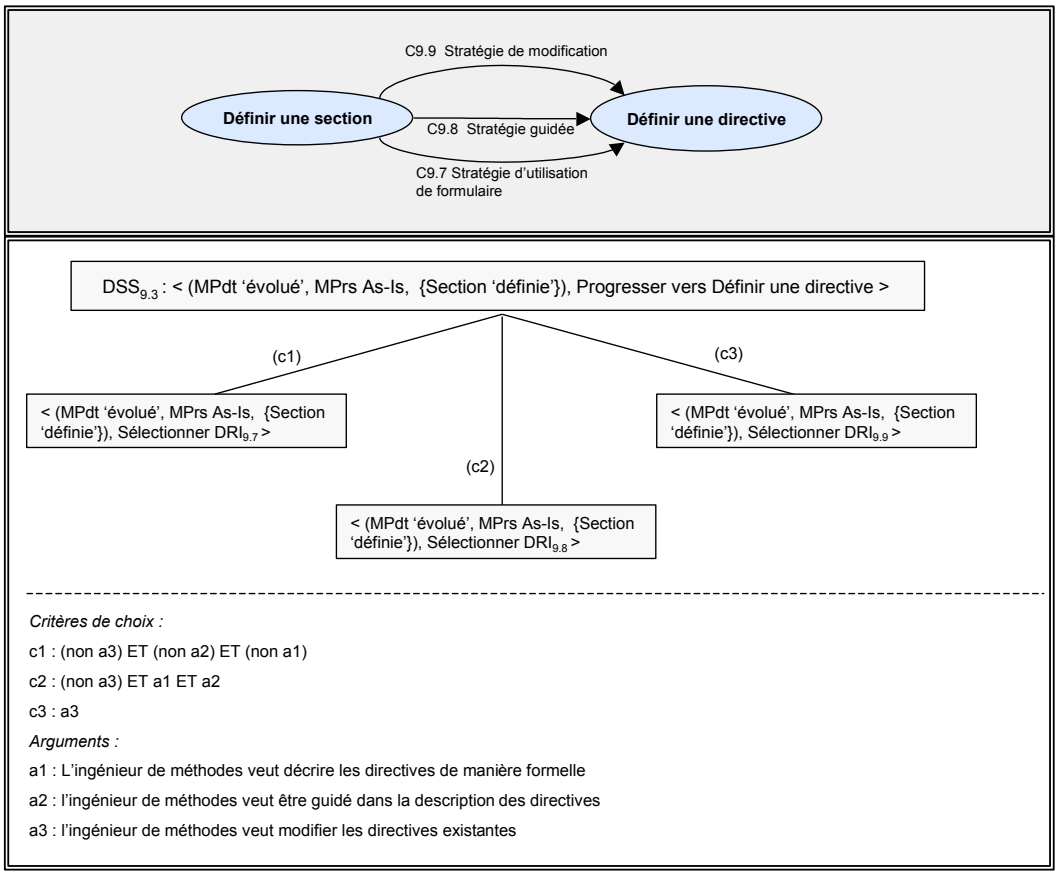


Figure 73. Structure de la $DSS_{9.3}$

4.2.9.3. Tableau des DRI associées à la carte C9

La carte C9 comprend 11 sections. A chaque section on associe une DRI, le Tableau 15 présente ces DRI.

Identifiant	Signature de DRI	Type
$DRI_{9.1}$	< (MPdt 'évolué', MPrs As-Is), Définir une section avec la stratégie structurelle>	Tactique
$DRI_{9.2}$	< (MPdt 'évolué', MPrs As-Is), Définir une section avec la stratégie fonctionnelle>	Tactique
$DRI_{9.3}$	< (MPdt 'évolué', MPrs As-Is, {Section 'définie'}), Définir une section avec la stratégie de découverte d'alternative >	Tactique
$DRI_{9.4}$	< (MPdt 'évolué', MPrs As-Is, {Section 'définie'}), Définir une section avec la stratégie de découverte par progression >	Tactique
$DRI_{9.5}$	< (MPdt 'évolué', MPrs As-Is, {Section 'définie'}), Définir une section avec la stratégie de découverte par regroupement >	Tactique
$DRI_{9.6}$	< (MPdt 'évolué', MPrs As-Is, {Section 'définie'}), Définir une section avec la stratégie de découverte par décomposition >	Tactique
$DRI_{9.7}$	< (MPdt 'évolué', Section 'définie'), Définir une directive avec la stratégie d'utilisation de formulaire >	Tactique
$DRI_{9.8}$	< (MPdt 'évolué', Section 'définie'), Définir une directive avec la stratégie guidée >	Tactique

$DRI_{9.9}$	< (MPdt 'évolué', Section 'définie'), Définir une directive avec la stratégie de modification>	Tactique
$DRI_{9.10}$	< (MPdt 'évolué', Section 'définie'), Définir une section avec la stratégie de correction>	Tactique
$DRI_{9.11}$	< (MPdt 'évolué', MPrs 'évolué', Arrêter avec la stratégie de validation >	Informelle

Tableau 15. Les DRI de la carte C9

Les directives présentées ci-dessus sont décrites à la section 4.3.2 de ce chapitre.

4.2.10. Réaliser «Faire évoluer le modèle de processus avec la stratégie orientée-contexte » (DRI_{10})

La DRI_{10} est associée à la stratégie *Orientée-Contexte* pour l'évolution du modèle de processus. Cette directive offre un guidage à l'ingénieur de méthodes, pour construire à partir du modèle de processus *As-Is,* un nouveau modèle de processus basé sur le paradigme de modélisation de processus contextuel tel que défini dans NATURE [Rolland 95], [Plihon 96], [Jarke 99].

En appliquant cette directive le modèle de processus est formalisé par une hiérarchie de contextes. Un contexte étant défini comme un couple <*Situation, Intention*>, la situation représente la partie de produit en cours de transformation et l'intention représente le but à atteindre dans cette situation. L'avantage de l'utilisation du paradigme *orienté-contexte* est de présenter à la fois les activités composant un processus et les décisions et les contextes qui conduisent à les exécuter.

La Figure 74 présente la DRI_{10} qui est une directive tactique de type plan.

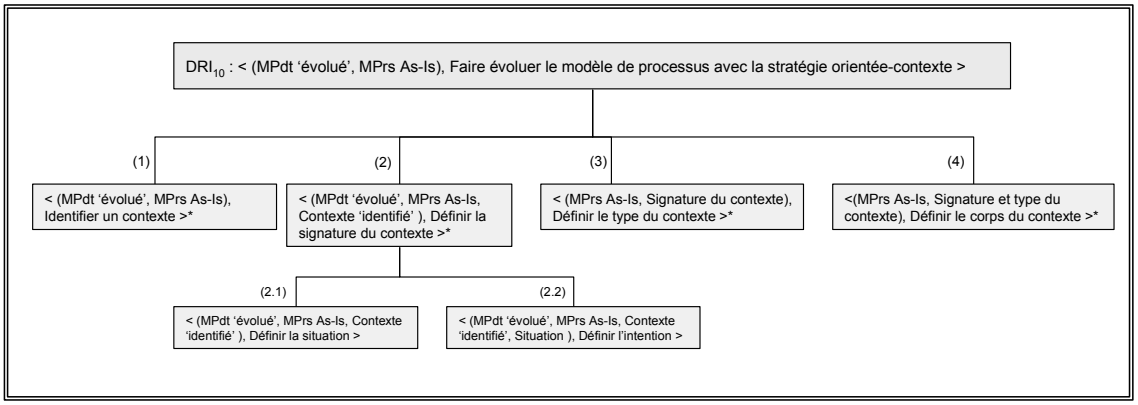


Figure 74. Structure de la DRI_{10}

(1) < *(MPdt 'évolué', MPrs As-Is), Identifier un contexte* > : Cette directive suggère à l'ingénieur de méthodes d'identifier un contexte qui permettra de construire une partie de produit de la méthode. L'identification des contextes se base sur le modèle de processus *As-Is* et sur les buts de la méthode *To-Be* définis par les exigences d'évolution. Cette opération consiste à analyser ces deux ensembles et à identifier un but pour chaque couple (*but*, *situation*).

(2) < *(MPdt 'évolué', MPrs As-Is, Contexte 'identifié'), Définir la signature du contexte* > : L'application de cette directive permet de définir la signature du contexte identifié, elle propose un plan proposant de : (2.1) *définir la situation* et (2.2) *définir l'intention*.

(3) < *(MPrs As-Is, Signature du contexte), Définir le type du contexte* > : Cette directive demande à l'ingénieur de méthodes de définir le type du contexte identifié. Un contexte peut être de type plan ou choix.

(4) <*(MPrs As-Is, Signature et type du contexte), Définir le corps du contexte* > : Cette directive suggère à l'ingénieur de méthodes de définir le corps du contexte. La définition du corps passe par la définition des sous-contextes du contexte à définir. Ces sous-contexte peuvent être de type exécutable, plan ou choix.

4.2.11. Réaliser « Faire évoluer le modèle de processus avec la stratégie orientée-patron » (DRI_{11})

La DRI_{11} est associée à la section <*Faire évoluer le modèle de produit, Faire évoluer le modèle de processus, Orientée-patron*> de la carte d'évolution. L'objectif de cette directive est de guider l'évolution du modèle de produit *As-Is* vers un modèle de processus prenant la forme d'un catalogue de patrons. Chaque patron identifie un problème générique rencontré lors de la construction du produit et propose une solution générique applicable chaque fois que ce problème apparaît. La partie solution d'un patron peut être exprimée par n'importe quel type de paradigme de modélisation de processus présenté dans cette section.

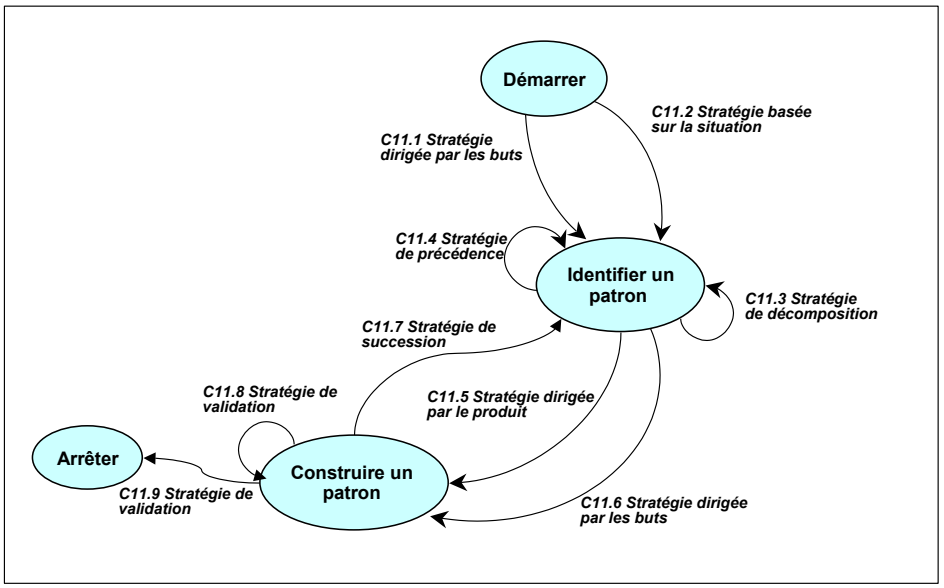


Figure 75. La carte C11 (DRI_{11})

La Figure 75 présente la *carte C11* modélisant la directive DRI_{11}. Elle contient deux intentions *Identifier un patron* et *Construire un patron*. L'intention *identifier un patron* vise à identifier un

problème pour lui associer un patron. L'identification du problème peut être basée sur l'identification de situations typiques récurrentes dans la construction du produit ou sur des buts génériques dans le contexte de la méthode. Ces deux cas de figure sont illustrés par les deux stratégies: *Stratégie basée sur la situation* et *Stratégie dirigée par les buts*.

Dans le cas d'identification d'un problème complexe nécessitant d'être décomposé en sous-problèmes pour pouvoir associer une solution à chaque sous-problème, la *Stratégie de décomposition* est utilisée pour identifier plusieurs patrons atomiques et les combiner dans un patron composé apportant la solution au problème complexe.

L'identification d'une situation conduisant à la création d'un patron nous amène à supposer que l'application d'un autre patron a crée cette situation, la *Stratégie de précédence* est utilisée pour l'identification et l'ordonnancement de patrons dans ce cas de figure.

L'intention *construire un patron* vise à formaliser le problème (le but et la situation) et de construire la solution à ce problème. Deux stratégies permettent d'atteindre cette intention : *Stratégie dirigée par le produit* et *Stratégie dirigée par les buts*. La première stratégie construit la solution du patron en formalisant le processus permettant la production des éléments du produit décrit dans son but à partir de ceux décrits dans sa situation. La deuxième stratégie construit la solution du patron en réduisant son but en un ensemble d'actions atomiques permettant d'opérationaliser ce but.

La *Stratégie de succession* permet de considérer le produit obtenu par l'application du patron défini comme une situation potentielle pour l'identification d'un autre patron.

La *Stratégie de validation* est sélectionnée pour tester la validité des patrons construits. La *Stratégie de complétude* est appliquée pour vérifier la cohérence du modèle de processus *To-Be*.

4.2.11.1. Directives de Sélection d'Intention associées à la carte C11

Le Tableau 16 présente les DSI associées à la carte C11

Intention	DSI
I_1 : Démarrer	N/A
I_2 : Identifier un patron	$DSI_{11.1}$: < (MPdt 'évolué', MPrs As-Is, {Patron 'identifié'}), Progresser depuis Identifier un patron >
I_3 : Construire un patron	$DSI_{11.2}$: < (MPdt 'évolué', {Patron 'défini'}), Progresser depuis Construire un patron >
I_4 : Arrêter	N/A

Tableau 16. Les DSI de la carte C11

4.2.11.1.1. Progresser depuis « Identifier un patron » ($DSI_{11.1}$)

La $DSI_{11.1}$ est de type choix, elle guide l'ingénieur de méthodes à progresser depuis l'intention *Identifier un patron*. Elle offre deux choix : sélectionner l'intention *Identifier un patron* ou sélectionner l'intention *Construire un patron*. Comme l'illustre la Figure 76 pour réaliser le premier choix on sélectionne la $DSS_{11.2}$ qui propose deux stratégies : *Stratégie de décomposition* et *Stratégie de précédence*. Pour réaliser le deuxième choix on sélectionne la $DSS_{11.3}$ qui permet de choisir entre les deux stratégies : *Stratégie dirigée par le produit* et *Stratégie dirigée par les buts*.

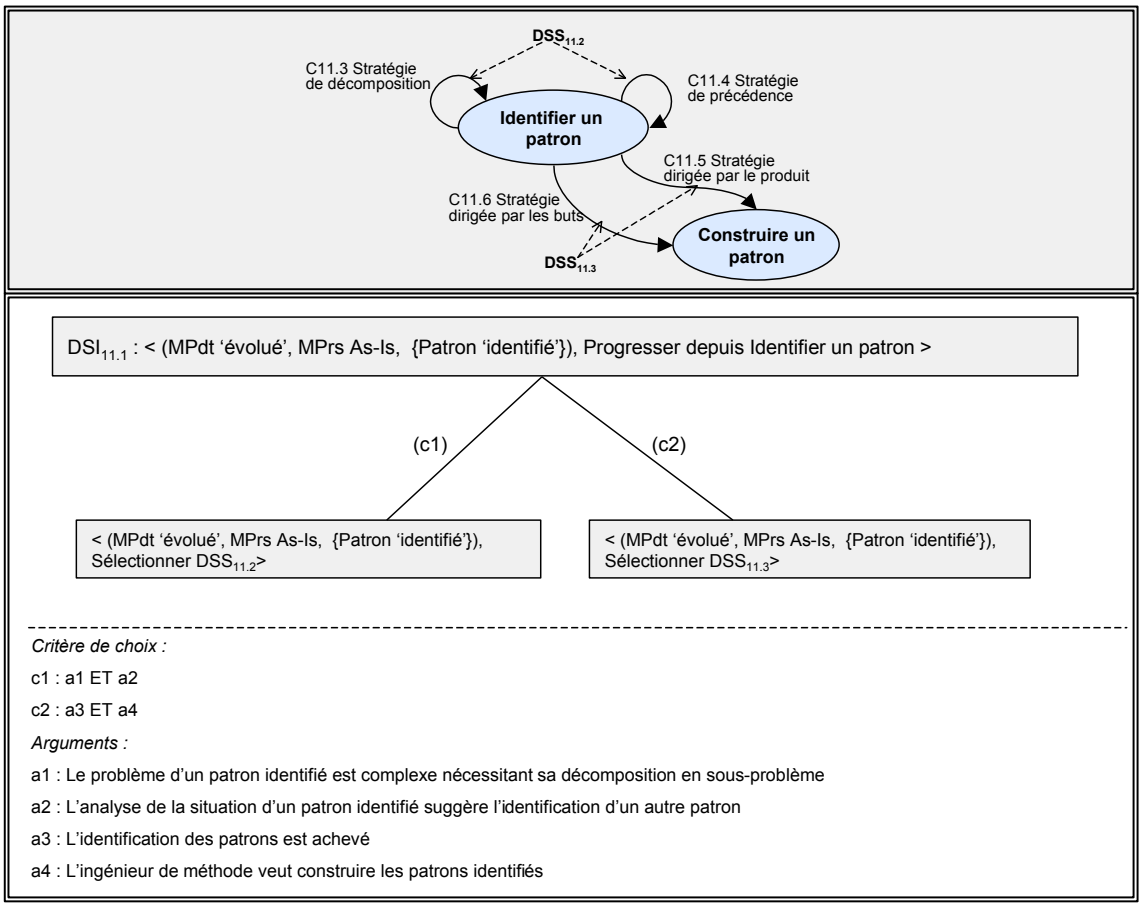


Figure 76. Structure de la $DSI_{11.1}$

4.2.11.1.2. Progresser depuis « Construire un patron » ($DSI_{11.2}$)

La $DSI_{11.2}$ est de type choix, elle guide l'ingénieur de méthodes à progresser dans la carte C11 depuis l'intention *Construire un patron*. La Figure 77 présente cette directive, elle offre trois possibilités : réaliser l'intention *Identifier un patron* en sélectionnant la $DRI_{11.7}$, réaliser l'intention *Construire un patron* en sélectionnant la $DRI_{11.8}$ ou atteindre l'intention *Arrêter* en sélectionnant la $DRI_{11.9}$.

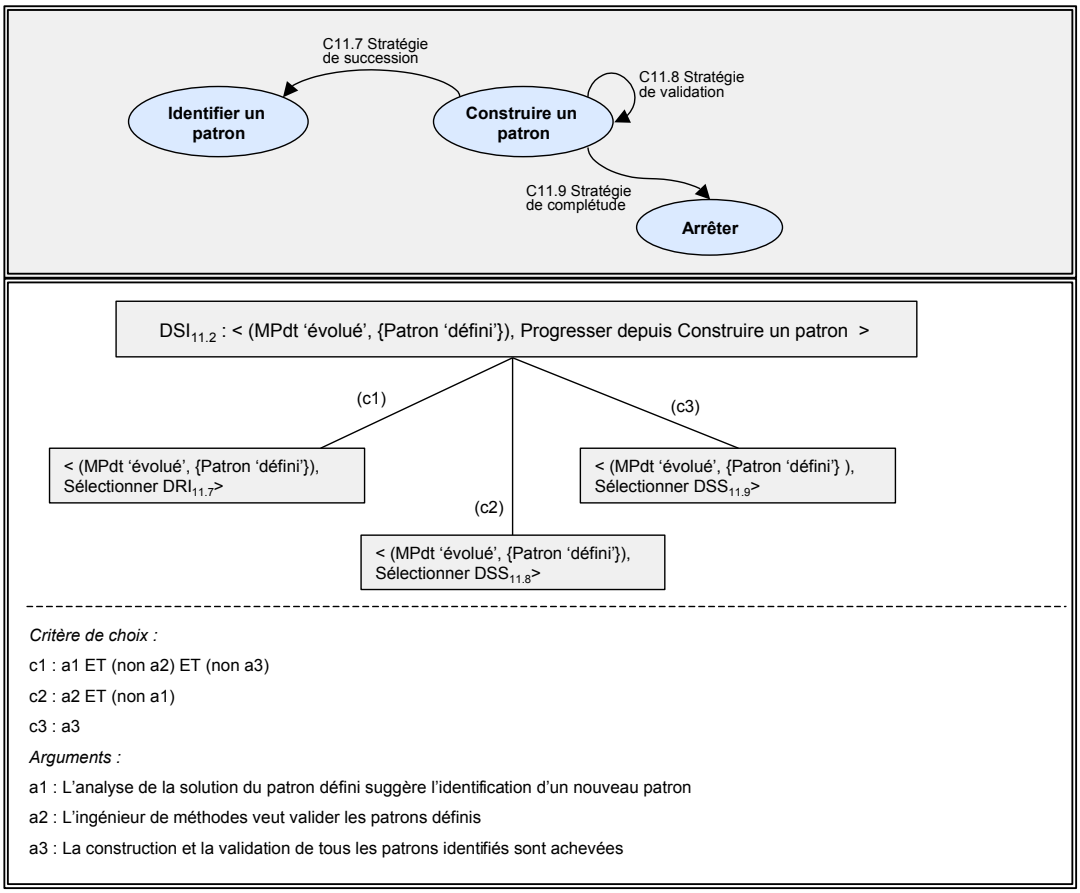


Figure 77. Structure de la $DSI_{11.2}$

4.2.11.2. Directives de Sélection de Stratégie associées à la carte C11

Le Tableau 17 présente les DSS associées à la carte C11.

Paire d'intentions	DSS
$(I_1 - I_2)$: (Démarrer, Identifier un patron)	$DSS_{11.1}$: < (MPdt 'évolué', MPrs As-Is), Progresser vers Identifier un patron>
$(I_2 – I_2)$: (Identifier un patron, Identifier un patron)	$DSS_{11.2}$: < (MPdt 'évolué', MPrs As-Is, {Patron 'identifié'}), Progresser vers Identifier un patron>
$(I_2 – I_3)$: (Identifier un patron, Construire un patron)	$DSS_{11.3}$: <(MPdt 'évolué', MPrs As-Is, {Patron 'identifié'}), Progresser vers Construire un patron>

Tableau 17. Les DSS de la carte C11

4.2.11.2.1. Progresser vers « Identifier un patron » ($DSS_{11.1}$)

La directive $DSS_{11.1}$ est de type choix, elle propose deux choix pour réaliser l'intention *Identifier un patron* : Sélectionner la directive $DRI_{11.1}$ correspondante à la *Stratégie dirigée par les buts,* ou sélectionner la directive $DRI_{11.2}$ correspondante à la *Stratégie basée sur la situation*. La Figure 78 présente la structure de la $DSS_{11.1}$.

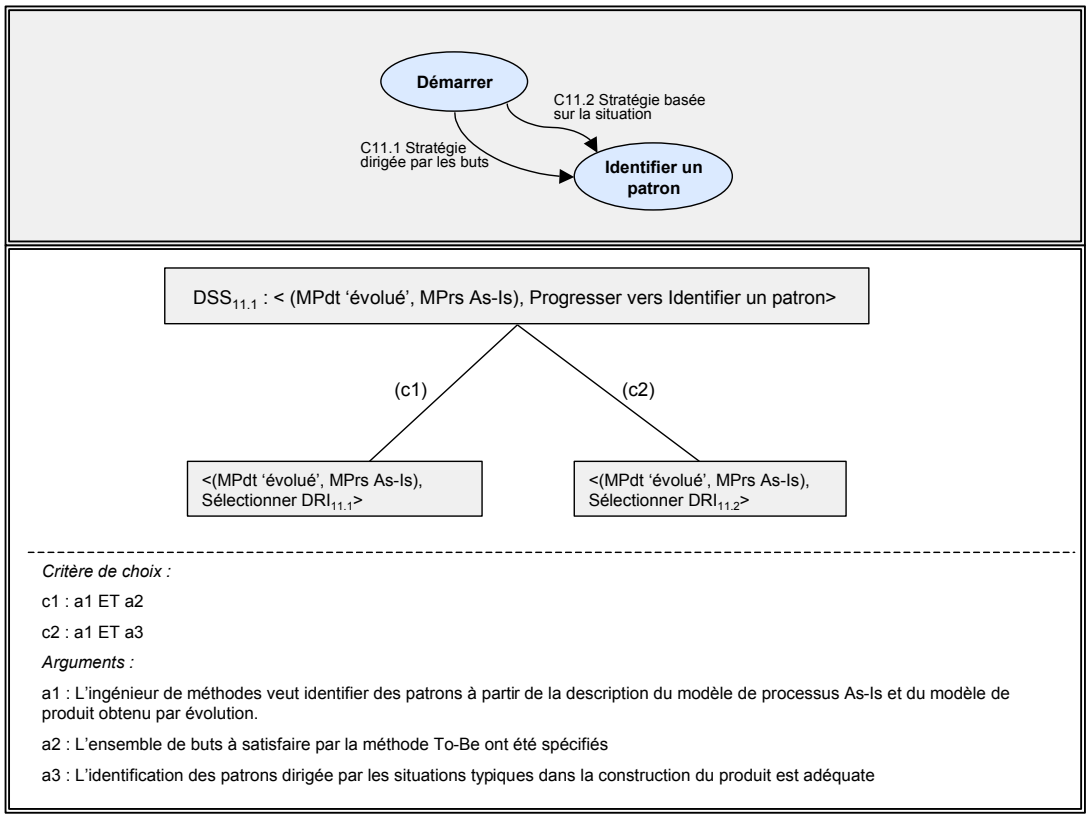


Figure 78. Structure de la $DSS_{11.1}$

4.2.11.2.2. Progresser vers « Identifier un patron » ($DSS_{11.2}$)

La $DSS_{11.2}$ est de type choix, elle permet de progresser à partir de l'intention *Identifier un patron* vers l'intention *Identifier un patron*, permettant ainsi de boucler sur l'intention elle-même pour identifier des patrons à partir de ceux déjà identifiés. Elle offre deux stratégies pour réaliser cela : *Stratégie de décomposition* et *Stratégie de précédence*. La Figure 79 présente la structure de la directive $DSS_{11.2}$.

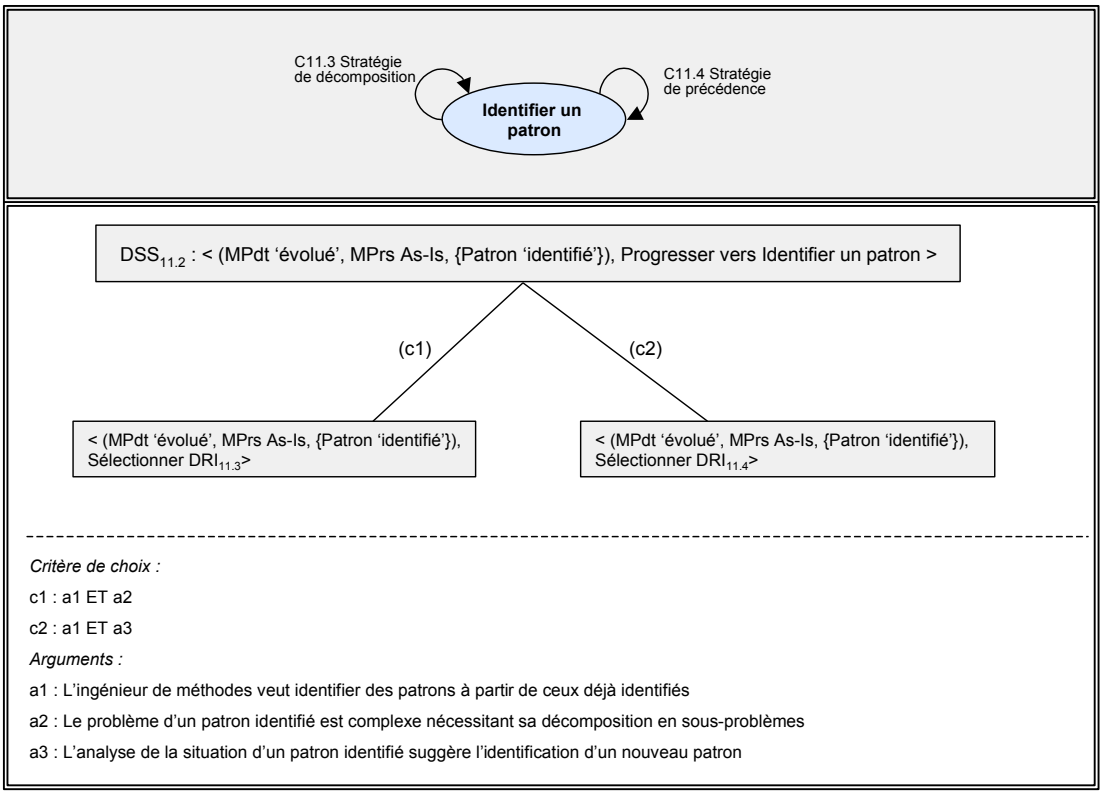


Figure 79. Structure de la $DSS_{11.2}$

4.2.11.2.3. Progresser vers « Construire un patron » ($DSS_{11.3}$)

La $DSS_{11.3}$ est de type choix, elle guide l'ingénieur de méthodes dans le choix de stratégie pour progresser de l'intention *Identifier un patron* vers l'intention *Construire un patron*. Comme le montre la Figure 80 cette directive offre deux choix : (1) Sélectionner la $DRI_{11.5}$ correspondante à la *Stratégie dirigée par le produit*, (2) Sélectionner la $DRI_{11.6}$ correspondante à la Stratégie dirigée par les buts.

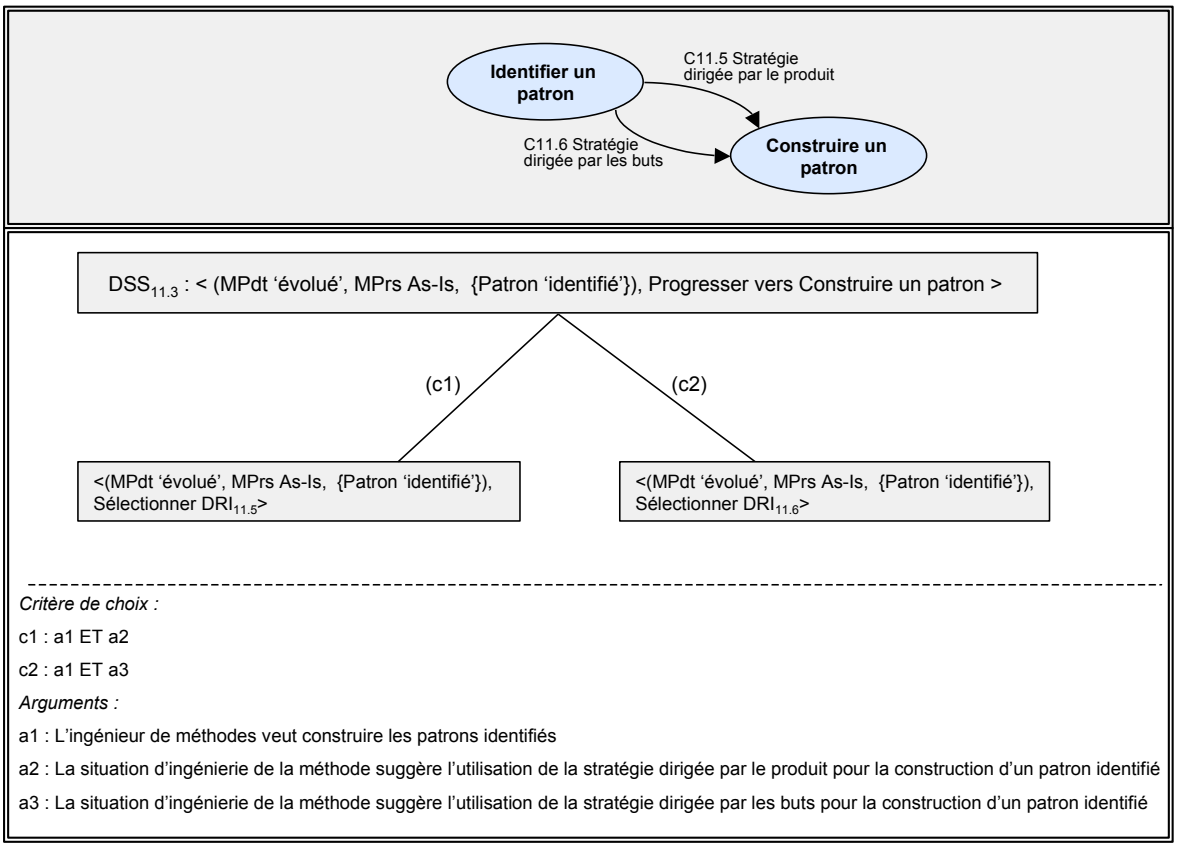


Figure 80. Structure de la $DSS_{11.3}$

4.2.11.3. Tableau des DRI associées à la carte C11

La carte C11 correspondant à la stratégie orientée-patron pour l'évolution du modèle du processus. Elle comporte neuf sections, le Tableau 18 présente les DRI associées à ces sections.

Identifiant	Signature de DRI	Type
$DRI_{11.1}$	< (MPdt 'évolué', MPrs As-Is), Identifier un patron avec la stratégie dirigée par les buts >	Tactique
$DRI_{11.2}$	< (MPdt 'évolué', MPrs As-Is), Identifier un patron avec la stratégie basée sur la situation >	Tactique
$DRI_{11.3}$	< (MPdt 'évolué', MPrs As-Is, {Patron 'identifié'}), Identifier un patron avec la stratégie de décomposition >	Tactique
$DRI_{11.4}$	< (MPdt 'évolué', MPrs As-Is, {Patron 'identifié'}), Identifier un patron avec la stratégie de précédence >	Tactique
$DRI_{11.5}$	< (MPdt 'évolué', Patron 'identifié'), Construire un patron avec la stratégie dirigée par le produit >	Tactique
$DRI_{11.6}$	< (MPdt 'évolué', Patron 'identifié'), Construire un patron avec la stratégie dirigée par les buts >	Tactique
$DRI_{11.7}$	< (MPdt 'évolué', {Patron 'défini'}), Identifier un patron avec la stratégie de succession >	Tactique
$DRI_{11.8}$	< (MPdt 'évolué', {Patron 'défini'}), Construire un patron avec la stratégie de validation >	Informelle

$DRI_{11.9}$	< (MPdt 'évolué', MPrs 'évolué'), Arrêter avec la stratégie de complétude >	Informelle

Tableau 18. Les DRI de la carte C11

Les directives présentées ci-dessus sont décrites à la section 4.3.3 de ce chapitre.

4.2.12. Réaliser « Faire évoluer le modèle de produit Par affinement » (DRI_{12})

La DRI_{12} est associée à la section C9 de la carte d'évolution. Cette directive à pour objectif d'affiner le modèle de produit obtenu par évolution pour tenir compte des évolutions subites par le modèle de processus. Pour réaliser l'affinement du modèle de produit, cette directive utilise les opérateurs d'adaptation de modèles présentée dans la DRI_4.

La Figure 81 présente la structure de la DRI_{12} qui est une directive tactique de type plan.

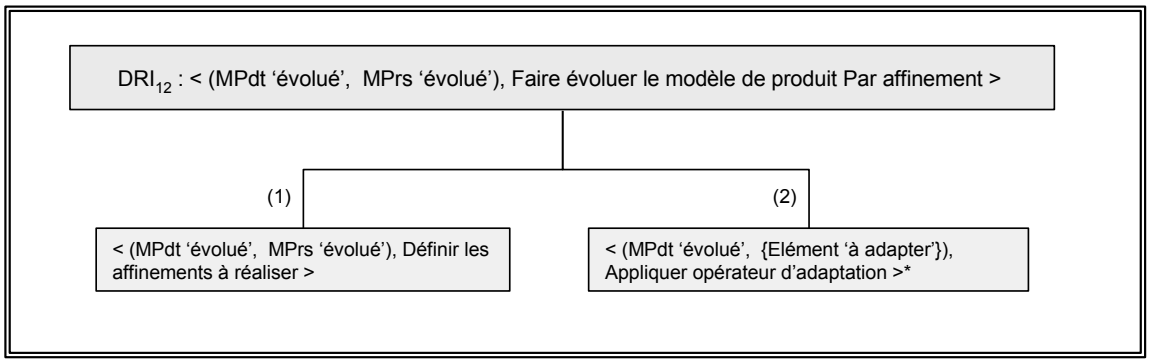


Figure 81. Structure de la DRI_{12}

(1) < (MPdt 'évolué', MPrs 'évolué'), Définir les affinements à réaliser > : Cette directive suggère à l'ingénieur de méthodes de réfléchir à l'impact des transformations subies par le modèle du processus sur le modèle de produit et à dresser la liste des affinements nécessaires.

L'ingénieur de méthodes est aussi invité à réaliser une confrontation du modèle de processus et du modèle de produit obtenus par évolution pour détecter les cas de discordance et définir les affinements nécessaires du modèle de produit pour y remédier.

(2) *< (MPdt 'évolué', {Elément 'à adapter'}), Appliquer opérateur d'adaptation >* : Cette directive a pour objectif d'appliquer les opérateurs d'adaptation pour réaliser les affinements définis par la directive précédente. Elle propose le plan présenté à la Figure 82 permettant de (2.1) sélectionner l'opérateur adéquat pour réaliser la transformation souhaitée et (2.2) d'appliquer l'opérateur sélectionné.

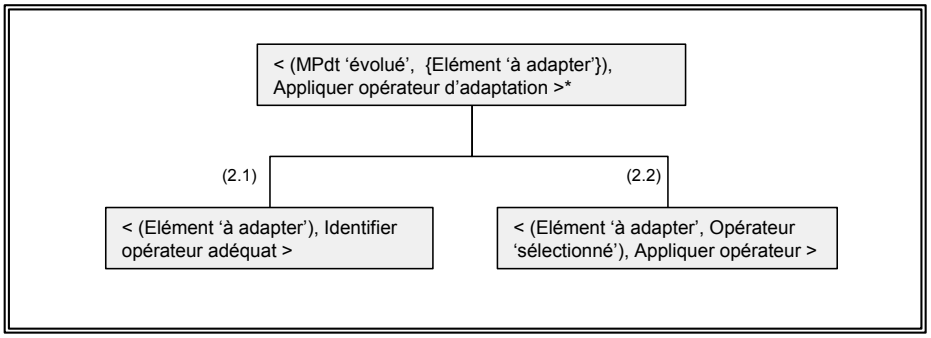


Figure 82. < (MPdt 'évolué', {Elément 'à adapter'}), Appliquer opérateur d'adaptation >

4.2.13. Réaliser « Faire évoluer le modèle de processus Par validation » (DRI_{13})

La DRI_{13} est associée à la section C10 de la carte d'évolution. Cette directive a pour objectif de valider le modèle de processus obtenu par validation. Cette directive est de type plan, elle est présentée à la Figure 83.

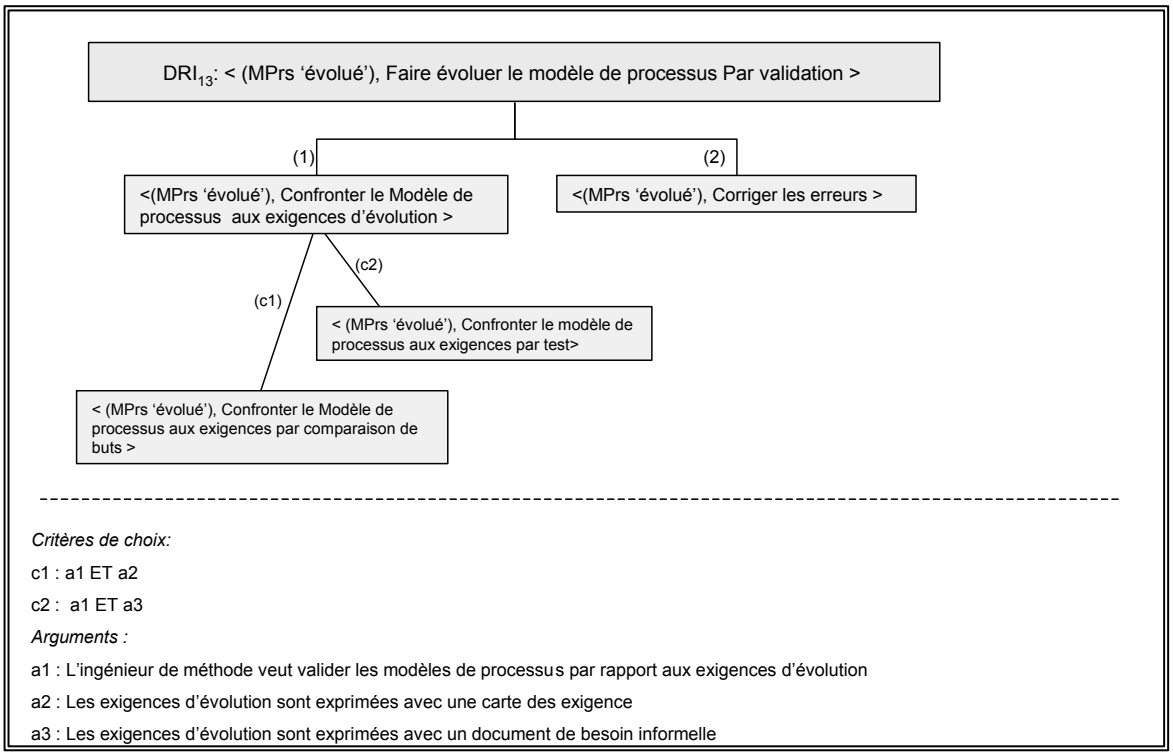


Figure 83. Structure de la DRI_{13}

(1) *< (MPrs 'évolué'), Confronter le Modèle de processus aux exigences d'évolution >* : Cette directive est de type choix, elle propose deux manières de valider le modèle de processus par rapport aux exigences d'évolution (par comparaison de buts et par test),

(2) *< (MPrs 'évolué'), Corriger les erreurs >* : Cette directive suggère à l'ingénieur de méthodes de corriger les éventuelles erreurs détectées par l'application de la directive précédente.

4.2.1.14. Réaliser « Arrêter Par vérification de complétude » (DRI_{14})

La DRI_{14} est associée à la section C11 de la carte d'évolution. Cette directive est la dernière étape dans le processus d'évolution de méthodes. Elle consiste en une étape de validation et de test de cohérence de la méthode *To-Be* avant sa livraison aux utilisateurs finaux.

La Figure 84 présente la DRI_{14} qui est une directive tactique de type plan. Ce plan propose à l'ingénieur de méthodes de : (1) Tester la méthode *To-Be* en l'appliquant pour le développement de '*SI test*' (2) Documenter la méthode *To-Be* pour faciliter sa compréhension et son apprentissage par les utilisateurs.

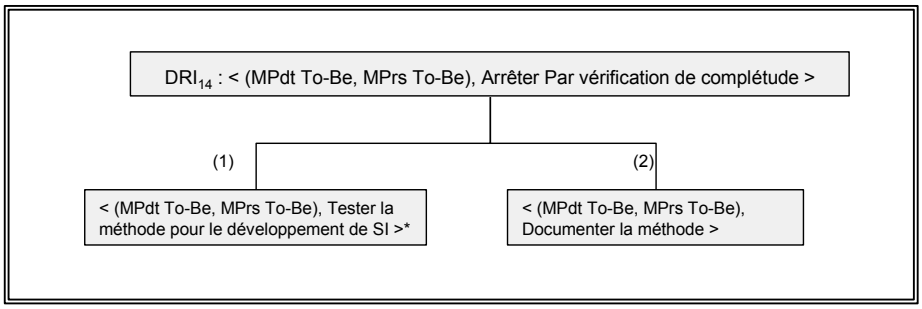


Figure 84. Structure de la DRI_{14}

4.3. Section des sous-cartes de d'évolution

4.3.1. Carte C3 : « Faire évoluer le modèle de produit Par abstraction de modèle » (DRI_3)

Cette section est consacrée à la présentation des Directives de Réalisation d'Intention, associées aux sections de la carte C3. Celle-ci est rappelée à la Figure 85.

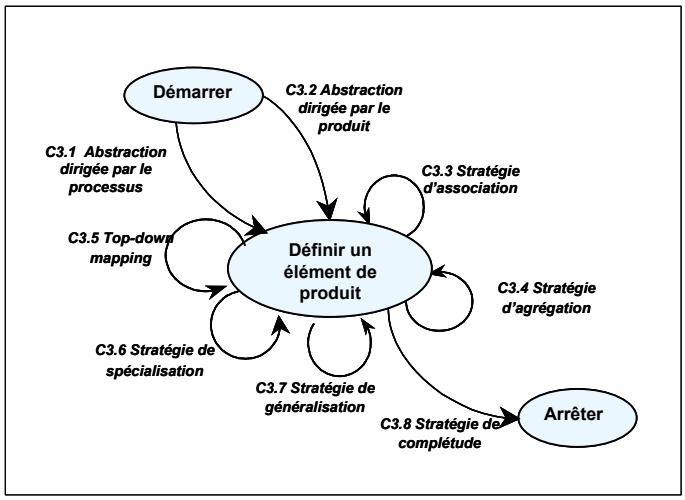


Figure 85. La carte C3

4.3.1.1. Réaliser « Définir un élément de produit par abstraction dirigée par le processus » ($DRI_{3.1}$)

La $DRI_{3.1}$ est associée à la section <*Démarrer, Définir un élément de produit, Abstraction dirigée par le processus*> de la carte C3. Cette directive est sélectionnée dans le cas où une abstraction réalisée sur le modèle de processus *As-Is* fait apparaître des activités manipulant des concepts ne faisant pas partie du modèle de produit *As-Is*. L'objectif de cette directive est de définir ces concepts et de les intégrer dans le modèle de produit en cours d'évolution.

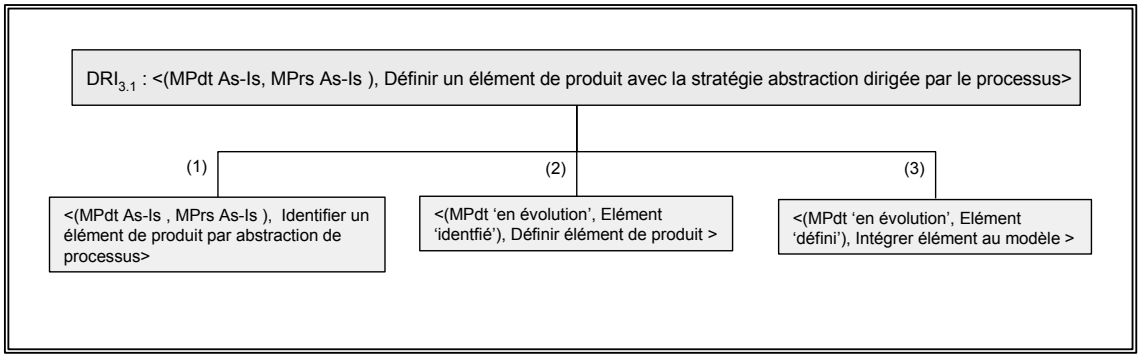


Figure 86. Structure de la $DRI_{3.1}$

La Figure 86 présente la directive $DRI_{3.1}$ de type plan décomposée comme suit :

(1) *Identifier un élément de produit par abstraction de processus :* Cette directive suggère à l'ingénieur de méthodes d'identifier un élément de produit correspond à une nouvelle activité du modèle de processus obtenue par abstraction.

(2) *Définir élément de produit* : L'application de cette directive permet de définir l'élément de produit identifié par la directive précédente. Cette directive est de type plan, elle est présentée à la Figure 35. Ce plan propose de (2.1) créer une classe pour modéliser l'élément identifié, (2.2) définir le nom de la classe et (2.3) définir ses attributs.

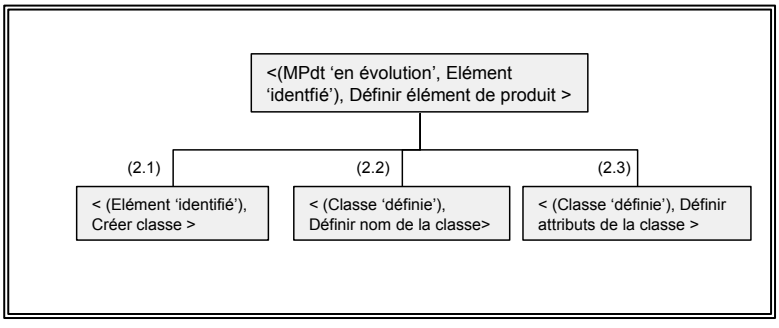


Figure 87. <(MPdt 'en évolution', Elément 'identifié'), Définir élément de produit >

(3) *Intégrer élément au modèle* : Une fois l'élément de produit défini, cette directive invite l'ingénieur de méthodes à l'intégrer dans la nouvelle couche du modèle de produit en cours de

construction. Cette intégration passe par l'ajout d'éventuels liens d'association, d'agrégation ou d'héritage entre le nouvel élément et les éléments existant dans cette couche.

4.3.1.2. Réaliser « Définir un élément de produit par abstraction dirigée par le produit » ($DRI_{3.2}$)

La $DRI_{3.2}$ est associée à la stratégie abstraction dirigée par le produit pour identifier un élément de produit. L'objectif de cette directive est de définir une nouvelle organisation du modèle de produit As-Is en couches. Chaque couche comporte des éléments qui possèdent le même niveau d'abstraction. La Figure 88 présente cette directive de type plan.

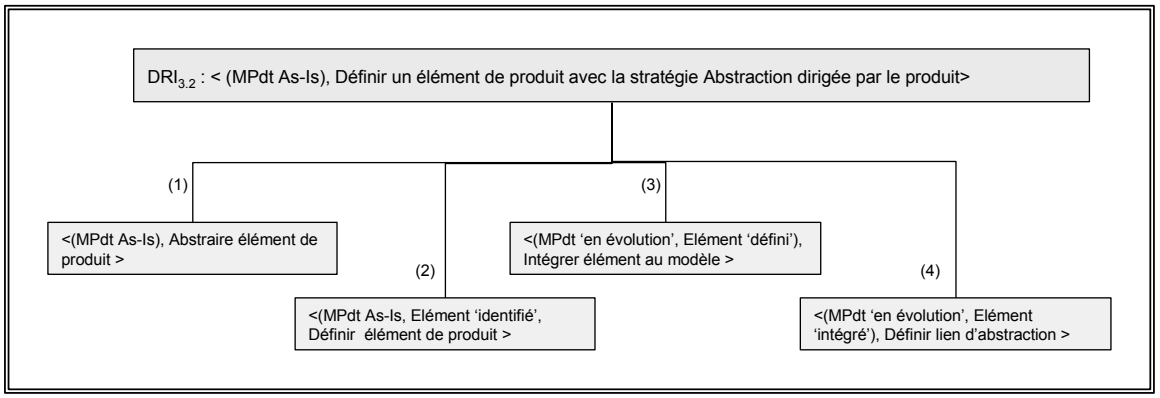


Figure 88. Structure de la $DRI_{3.2}$

Le plan décrivant la $DRI_{3.2}$ est décomposé comme suit :

(1) *Abstraire élément de produit* : Cette directive suggère à l'ingénieur de méthode d'analyser le modèle de produit *As-Is* dans le but d'identifier des éléments de ce modèle pouvant être représenté par des concept d'un niveau d'abstraction plus élevé.

(2) *Définir élément de produit* : Cette directive est semblable à celle présentée à la Figure 35 de la $DRI_{3.1}$.

(3) *Intégrer élément au modèle* : L'application de cette directive permet d'intégrer l'élément défini dans la nouvelle couche du modèle en cours de construction. Cette intégration passe par l'ajout d'éventuels liens d'association, d'agrégation ou d'héritage entre le nouvel élément et les éléments existant dans cette couche.

(4) *Définir lien d'abstraction* : Cette directive suggère à l'ingénieur de méthode de modéliser le lien d'abstraction entre le nouvel élément crée l'élément créer et l'élément à partir du quel on a fait abstraction. L'application de cette directive permet ainsi de définir des liens d'abstraction entre les éléments des deux couches du modèle.

4.3.1.3. Réaliser « Définir un élément de produit avec la stratégie d'association » ($DRI_{3.3}$)

La $DRI_{3.3}$ est associée à la *stratégie d'association* pour la définition d'un élément de produit. Cette directive est sélectionnée pour compléter le modèle de produit en cours de construction en ajoutant des liens d'association entre les éléments déjà définis dans le modèle. La $DRI_{3.3}$, présentée à la Figure 89, est de type plan. Ce plan est décrit comme suit :

(1) *Identifier un lien d'association entre deux classes* : Le but de cette directive est d'analyser la nouvelle couche de modèle en cours de construction dans le but d'identifier d'éventuelles relations sémantiques durables entre deux classes non modélisées par un lien.

(2) *Définir un lien d'association* : L'application de cette directive permet de définir le lien d'association défini par l'application de la directive précédente. Cette directive est de type plan, elle propose de : (2.1) Créer le lien d'association, (2.2) Définir le nom du lien et (2.3) Définir les cardinalités associées au lien.

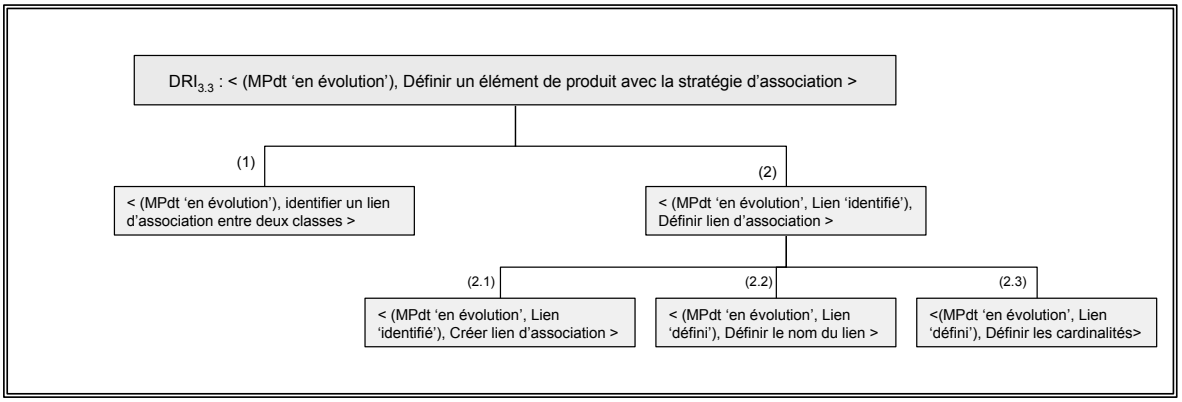


Figure 89. Structure de la $DRI_{3.3}$

4.3.1.4. Réaliser « Définir un élément de produit avec la stratégie d'agrégation » ($DRI_{3.4}$)

La $DRI_{3.4}$ est associée à la section C3.4 de carte C3. Cette directive a pour objectif de compléter la construction de la nouvelle couche du modèle de produit *To-Be* en ajoutant les liens d'agrégation entre les éléments définis dans cette couche.

La Figure 90 présente la $DRI_{3.4}$ qui est une directive tactique de type plan proposant de :

(2) *Identifier une relation d'imbrication* : Cette directive exécutable suggère à l'ingénieur de méthodes d'analyser la nouvelle couche du modèle en cours de construction dans le but d'identifier une éventuelle relation d'imbrication entre les éléments de cette couche.

(2) *Spécifier le type du lien d'imbrication* : L'application de cette directive exécutable permet de spécifier le type du lien d'imbrication : *Composition* ou *Agrégation*. Une composition étant une agrégation plus forte impliquant que : (i) un élément ne peut appartenir qu'à un seul

agrégat composite et (ii) la destruction de l'agrégat composite entraîne la destruction de tous ses éléments.

(3) *Créer le lien* : L'application de cette directive permet de créer une instance du lien d'agrégation ou de composition entre l'élément composite et l'élément composant.

(4) *Spécifier cardinalité* : Cette directive suggère à l'ingénieur de méthodes de spécifier les cardinalités associées au lien créé.

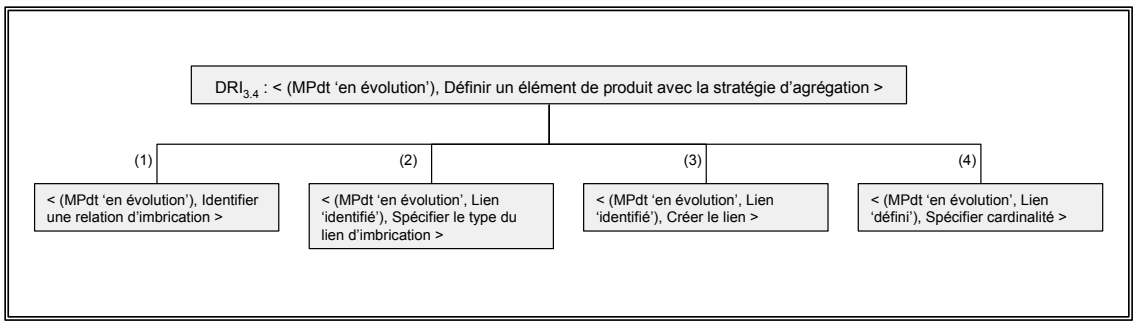


Figure 90. Structure de la $DRI_{3.4}$

4.3.1.5. Réaliser « Définir un élément de produit avec la stratégie Top-down mapping » ($DRI_{3.5}$)

La directive $DRI_{3.5}$ est associée à la section C3.5, elle est sélectionnée pour compléter le modèle de produit en cours d'évolution en ajoutant des liens d'abstraction entre les éléments des différentes couches de ce modèle.

La Figure 91 présente la $DRI_{3.5}$ qui est une directive tactique de type plan. Ce plan propose de (1) Identifier un lien d'abstraction, non modélisé, entre deux éléments n'appartenant pas à la même couche et (2) définir le lien d'abstraction identifié.

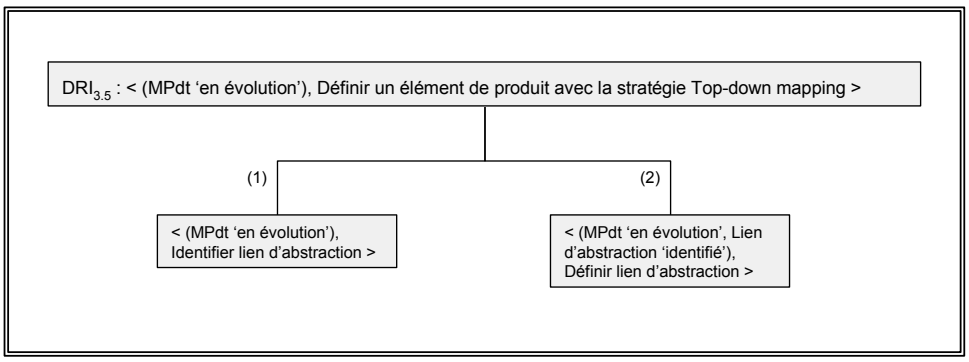


Figure 91. Structure de la $DRI_{3.5}$

4.3.1.6. Réaliser « Définir un élément de produit avec la stratégie de spécialisation » ($DRI_{3.6}$)

La $DRI_{3.6}$ est associée à la section C3.6 de la carte C3. L'objectif de cette directive est d'analyser la nouvelle couche du modèle de produit en cours de construction dans le but d'identifier un élément de

cette couche nécessitant d'être spécialisé. (i.e. une classe qui doit être spécialisée en deux ou plusieurs classes).

La Figure 92 présente la $DRI_{3.6}$ qui est une directive tactique de type plan. Ce plan propose à l'ingénieur de méthodes de (1) identifier la super-classe qui nécessite d'être spécialisée, (2) définir la ou les sous-classes et (3) créer le lien *Est-un* entre la super-classe et chaque sous-classe.

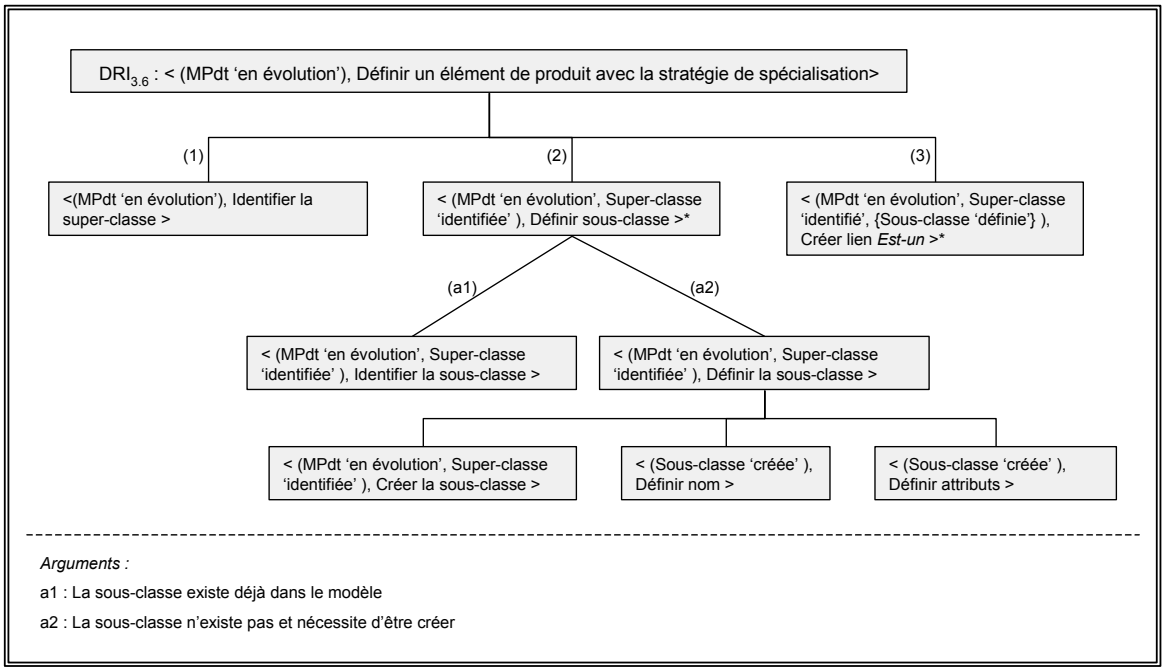


Figure 92. Structure de la $DRI_{3.6}$

Deux cas de figure peuvent se présenter pour définir une sous-classe : (i) Identifier une sous classe existante dans le modèle, (ii) Créer une nouvelle sous-classe spécialisant la super-classe.

4.3.1.7. Réaliser « Définir un élément de produit avec la stratégie de généralisation» ($DRI_{3.7}$)

La $DRI_{3.7}$ correspond à la section C3.7 de la carte C3. A l'inverse de la précédente, l'objectif de cette directive est de compléter le modèle de produit en cours d'évolution en traitant les cas de généralisation.

La Figure 93 présente la $DRI_{3.7}$ qui est une directive tactique de type plan, ce plan propose de (1) identifier les sous-classes (des classes représentant une même entité générique et ayant des attributs en commun), (2) définir la super-classe généralisant les sous-classes identifiées et (3) créer le lien d'héritage *Est-un*.

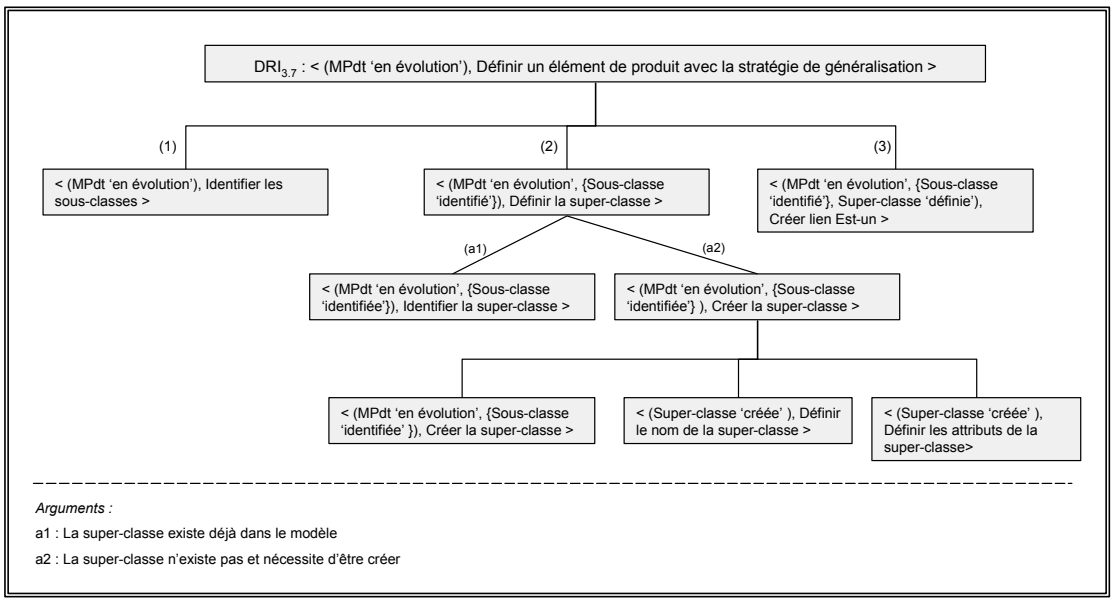


Figure 93. Structure de la $DRI_{3.7}$

4.3.1.8. Réaliser « Arrêter avec la stratégie de complétude » ($DRI_{3.8}$)

La directive $DRI_{3.9}$ est associée à la stratégie de complétude pour arrêter le processus d'évolution du modèle de produit par abstraction. Elle suggère à l'ingénieur de méthodes de vérifier la complétude du modèle de produit obtenu par évolution en s'assurant qu'il représente la totalité du produit qu'on souhaite obtenir par l'application de la méthode.

4.3.2. Carte C9 : « Faire évoluer le modèle de processus avec la stratégie orientée-stratégie » (DRI_9)

Cette section est consacrée à la présentation des Directives de Réalisation d'Intention, associées aux sections de la carte C9. Celle-ci est rappelée à la Figure 94.

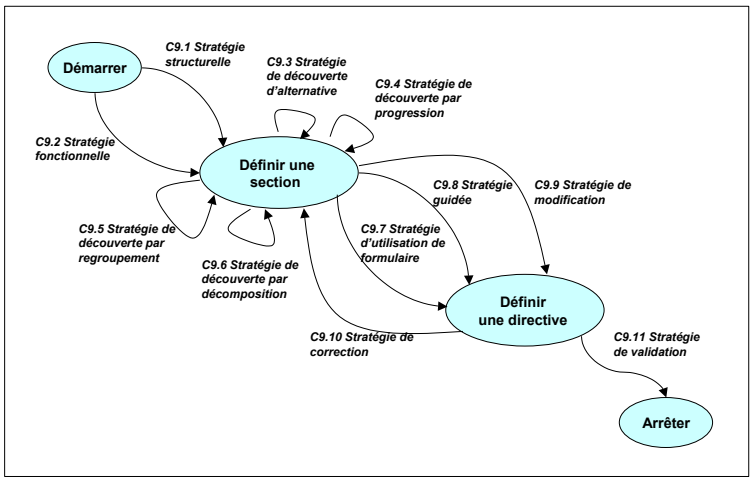


Figure 94. La carte C9

4.3.2.1. Réaliser « Définir une section avec la stratégie structurelle » (DRI$_{9.1}$)

La stratégie structurelle de définition des sections s'applique dans le cas où le modèle de processus de la méthode *As-Is* est décrit de manière informelle. La DRI$_{9.1}$ permet à l'ingénieur de méthodes de définir des sections de la carte formalisant le modèle de processus *To-Be* à partir de la description du processus et du produit de la méthode As-Is.

La définition des sections de la carte de méthode To-Be est centrée dans ce cas sur la structure du modèle de produit de la méthode. Les éléments de produit permettent de définir les intentions, les liens de précédence entre intentions et ainsi les sections de la carte de méthode.

La Figure 95 présente la structure de la DRI$_{9.1}$ qui est une directive tactique de type plan. Ce plan propose tout d'abord d'identifier la signature de la carte de méthode (1), ensuite d'identifier les intentions (2), d'identifier les stratégies pour satisfaire ces intentions (3) et finalement de définir les liens de précédence entre les intentions (4).

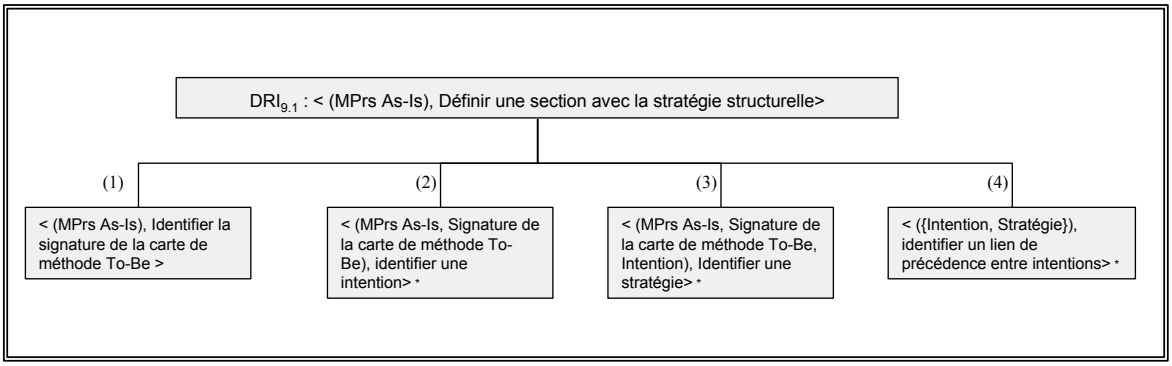


Figure 95. Structure de la DRI$_{9.1}$

(1) < *(MPrs As-Is), Identifier la signature de la carte de méthode To-Be* > : La définition des sections de la carte passe premièrement par la définition de la signature de la carte. Cette directive, présentée à la Figure 96, est une directive plan comportant deux directives exécutables < *(MPrs As-Is, Identifier la situation*> et <*(MPrs As-Is, Situation), Identifier l'intention*> permettant respectivement, de définir la situation et de définir l'intention de la carte de méthode en utilisant la structure linguistique (verbe, paramètres).

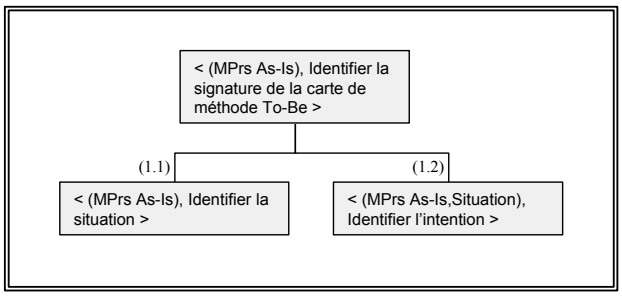


Figure 96. < (MPrs As-Is), Identifier la signature de la carte de méthode To-Be >

(2) < *(MPrs As-Is, Signature de la carte de méthode To-Be), identifier une intention* >: Cette directive offre un guidage à l'ingénieur de méthodes pour identifier une intention de la carte de la méthode *To-Be*. Cette directive, présentée à la Figure 97 est de type plan; ce plan propose quatre directives exécutables.

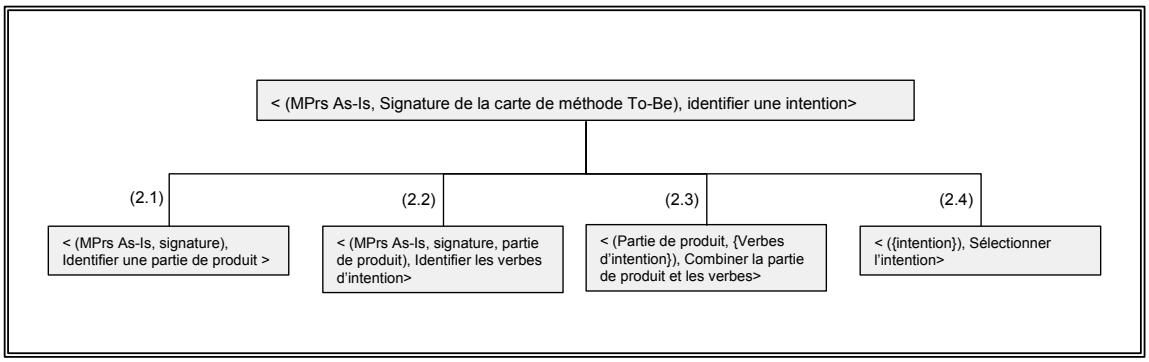


Figure 97. < (MPrs As-Is, Signature de la carte de méthode To-Be), identifier une intention >

(3) < *(MPrs As-Is, Signature de la carte de méthode To-Be, Intention), Identifier une stratégie*>: Le but de cette directive est d'identifier une stratégie à partir de la signature de la section et d'une intention déjà identifiée. Comme le montre la Figure 98 cette directive est de type plan, ce plan est composé de deux directives :

(3.1) < *(MPrs As-Is, Intention I), Identifier manières pour réaliser l'intention I*> : Cette directive informelle vise à définir une manière pour réaliser l'intention I.

(3.2) < *(Intention I, {Manière}), Agréer des manières en une stratégie* > : Cette directive est de type choix elle permet d'identifier une stratégie soit à partir d'une manière (s'il existe une seule manière), soit en agrégeant plusieurs manières en une stratégie (s'il existe plusieurs manières).

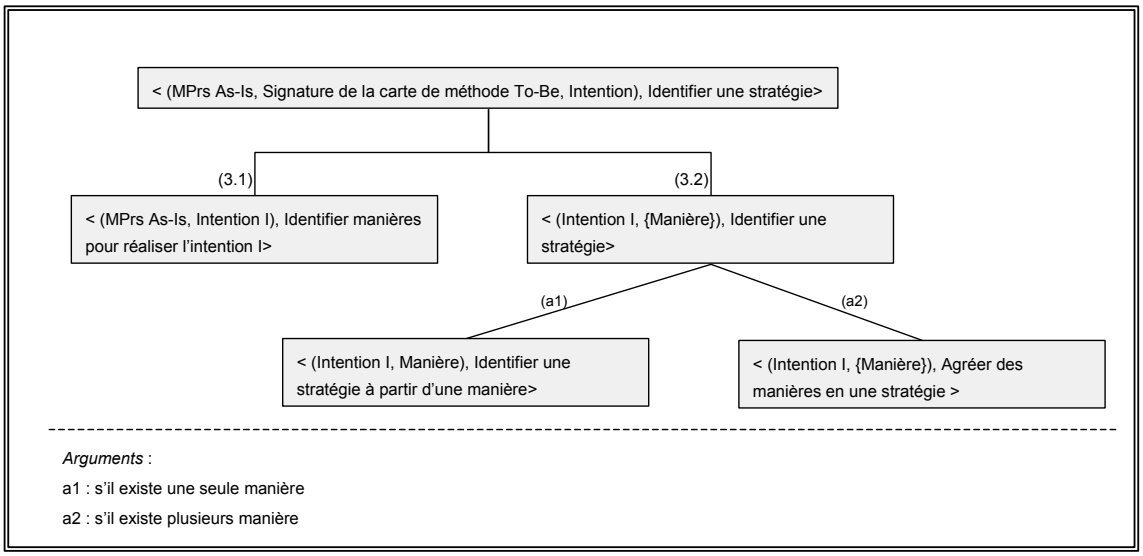


Figure 98. < (MPrs As-Is, Signature de la carte de méthode To-Be, Intention), Identifier une stratégie>

(4) *< ({Intention, Stratégie}), identifier un lien de précédence entre intentions>*: Cette directive permet l'identification des sections par définition de la situation d'application de l'intention. Cette directive est un plan qui s'exécute en séquence (Figure 99).

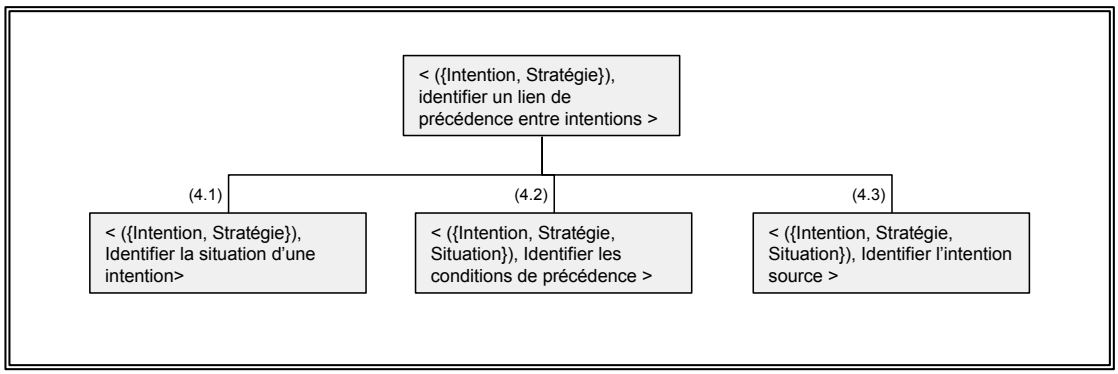


Figure 99. < ({Intention, Stratégie}), identifier un lien de précédence entre intentions >

4.3.2.2. Réaliser « Définir une section avec la stratégie fonctionnelle » ($DRI_{9.2}$)

La définition des sections avec la stratégie fonctionnelle s'applique dans le cas où le modèle de processus de la méthode *As-Is* est décrit de manière plus ou moins structurée, par des étapes et des opérations à réaliser afin d'obtenir le produit proposé par la méthode. La $DRI_{9.2}$ permet à l'ingénieur de méthodes de définir des sections de la carte formalisant le modèle de processus To-Be à partir des étapes et des opérations du modèle de processus *As-Is*. En effet, les étapes à effectuer dans la démarche de la méthode *As-Is* permettent de définir les intentions, les liens de précédence entre intentions et ainsi les sections de la carte de la méthode To-Be.

Comme le montre la Figure 100 illustrant la $DRI_{9.2}$, cette directive a la même structure que la $DRI_{9.1}$ mais le produit de départ n'est pas le même. Alors que la $DRI_{9.1}$ travaille sur la description informelle

de la méthode *As-Is*, la $DRI_{9.2}$ est basée sur l'analyse de la description plus structurée qui fait apparaître les étapes, les fonctions à réaliser et/ou les opérations à exécuter pour construire le produit proposé par la méthode.

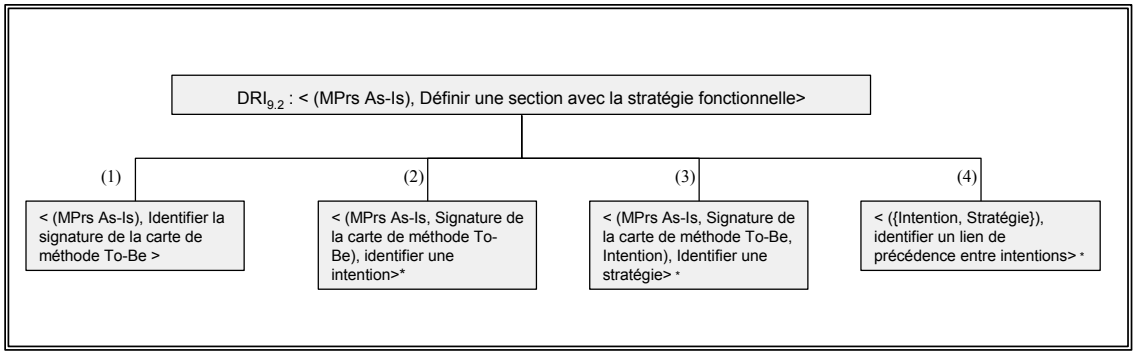


Figure 100. $DRI_{9.2}$: < (MPrs As-Is), Définir une section avec la stratégie fonctionnelle>

(1) *< (MPrs As-Is), Identifier la signature de la carte de méthode To-Be >* : Cette directive est identique à celle présentée à la Figure 96.

(2) *< (MPrs As-Is, Signature de la carte de méthode To-Be), identifier une intention>* : Cette directive est de type plan, elle vise à *identifier une intention* à partir de la signature d'une section et en partant de la description du modèle de processus *As-Is*. La Figure 101 présente cette directive, elle propose de (2.1) Identifier une étape ou fonction de processus, (2.2) Identifier une partie de produit qui est la cible de l'étape identifié, (2.3) Désigner l'étape par un ou plusieurs verbes d'intention, (2.4) Construire l'intention en combinant la partie de produit avec les verbes sélectionnés et (2.5) sélectionner l'intention la plus pertinente pour représenter l'étape de processus concerné.

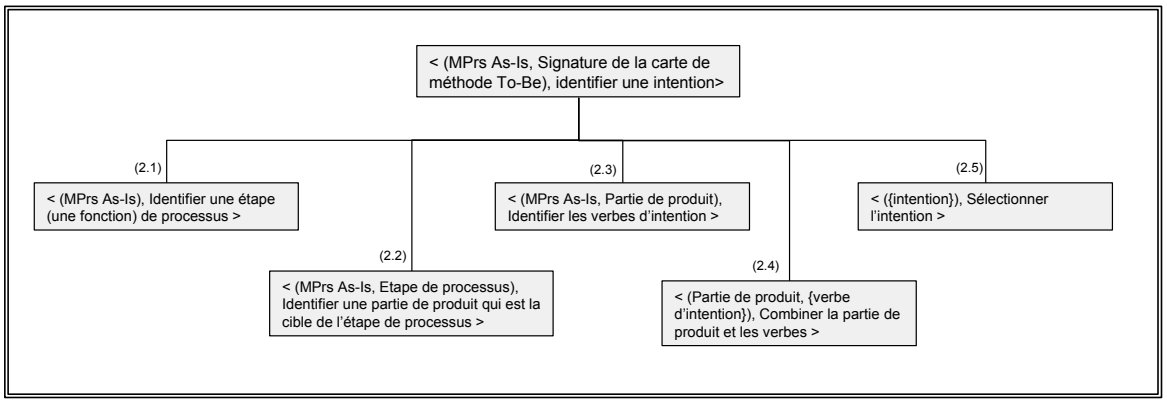


Figure 101. < (MPrs As-Is, Signature de la carte de méthode To-Be), identifier une intention>

L'identification des intentions à partir du modèle de processus As-Is structuré consiste à identifier les principales étapes de ce processus en commençant par le niveau d'abstraction le plus haut possible en sachant que chaque étape doit construire ou modifier une des parties de produit. Si à un niveau donné

les étapes proposent déjà plusieurs manières pour manipuler les parties de produit concernées, on peut s'arrêter à ce niveau d'abstraction. Sinon, il est conseillé de descendre d'un niveau, c'est-à-dire de décomposer chaque étape en sous étapes, surtout si les étapes manipulent des parties de produit complexes qui peuvent être décomposées et traitées séparément.

(3) < *(MPrs As-Is, Signature de la carte de méthode To-Be, Intention), Identifier une stratégie*> : L'identification des stratégies est également basée sur l'analyse du modèle de processus *As-Is*. L'ingénieur de méthodes est invité par cette directive à analyser chaque étape du processus et à découvrir s'il propose plusieurs manières pour satisfaire l'intention qui était identifiée dans cette étape. La Figure 102 présente cette directive qui est une directive tactique de type plan.

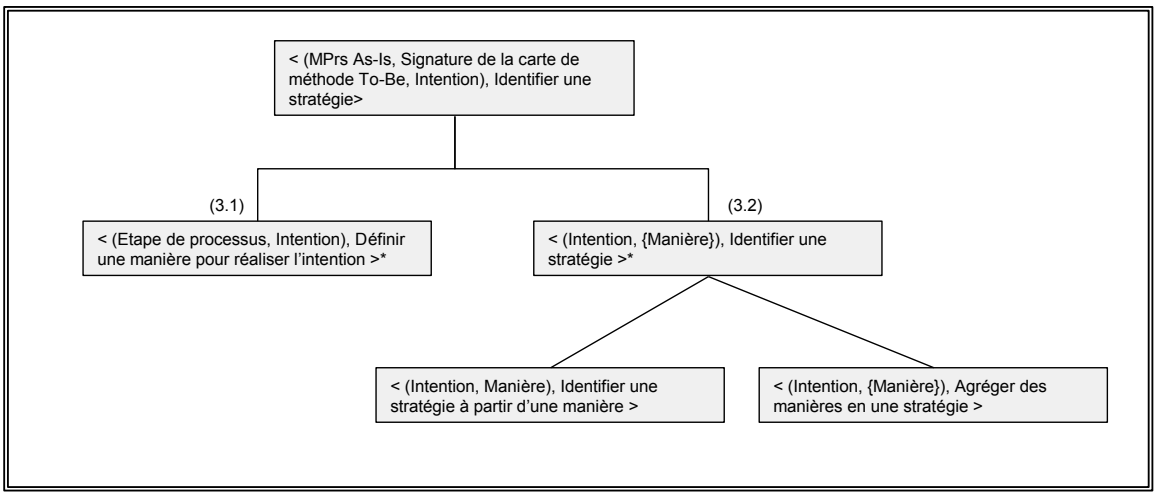


Figure 102. < (MPrs As-Is, Signature de la carte de méthode To-Be, Intention), Identifier une stratégie>

(4) < *({Intention, Stratégie}), identifier un lien de précédence entre intentions* > : Cette directive est identique à celle présenté à la Figure 99.

4.3.2.3. Réaliser « Définir une section avec la stratégie de découverte d'alternative » ($DRI_{9.3}$)

La $DRI_{9.3}$ est associée à la stratégie de découverte d'alternative pour la définition des sections. Elle aide à la découverte de sections alternatives à une section considérée, elle permet en particulier de trouver des sections ayant soit des intentions cibles alternatives, soit des stratégies alternatives.

Comme indiqué à la Figure 103, la $DRI_{9.3}$ est de type choix. Elle propose deux possibilités :

- < *({section}), Définir une section alternative en terme d'intention cible alternative* > : Cette directive peut être choisie s'il existe des alternatives à l'intention cible de la section. Elle est de type plan et composée de deux sous-directives qui visent à *Définir une intention cible alternative* et à *Définir une stratégie* respectivement.

- *< ({section}), Définir une section alternative en terme de stratégie alternative >* : Cette directive informelle vise à définir une stratégie alternative afin de définir une section alternative.

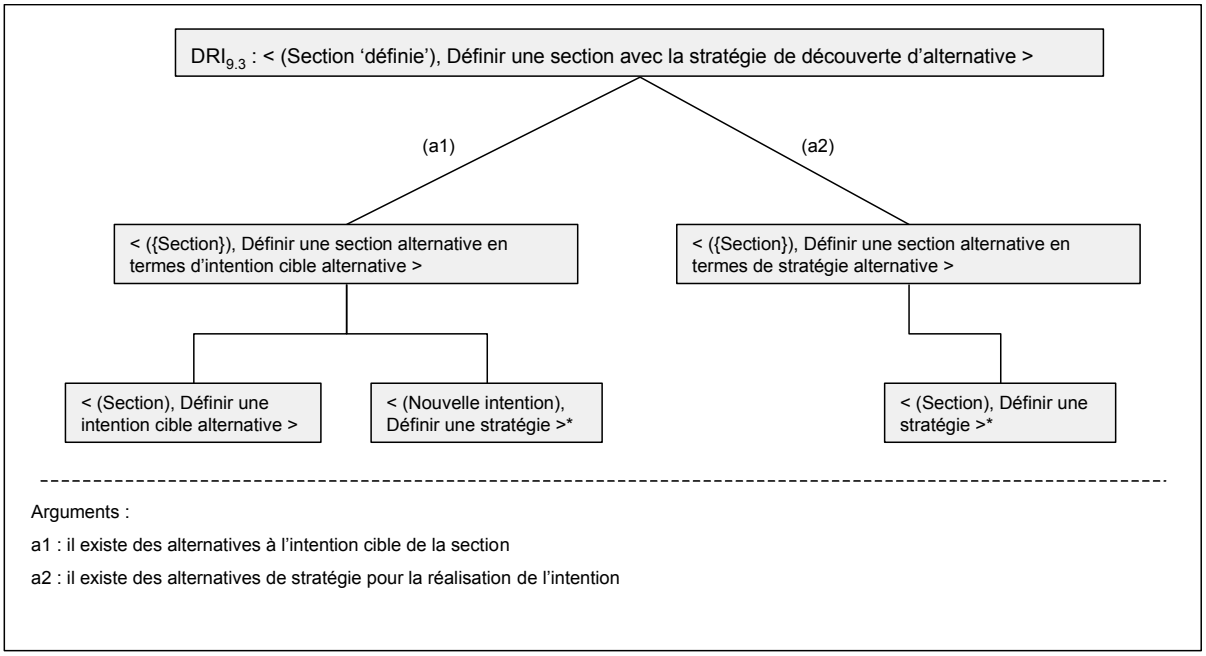


Figure 103. Structure de la $DRI_{9.3}$

4.3.2.4. Réaliser « Définir une section avec la stratégie de découverte par progression » ($DRI_{9.4}$)

La $DRI_{9.4}$ sert à la découverte des sections qui suivent une section donnée, c'est-à-dire qu'elle permet de trouver des sections permettant de progresser à partir de la situation engendrée par l'exécution de la section donnée. Cette directive est présentée à la Figure 104 ; elle est décrite sous forme d'un plan contenant trois éléments:

(1) L'analyse de l'intention cible de la section concernée,

(2) La définition d'une nouvelle intention utilisant en tant que produit source le produit obtenu en exécutant l'intention considérée et

(3) La définition de la stratégie permettant de satisfaire la nouvelle intention à partir de l'intention du départ.

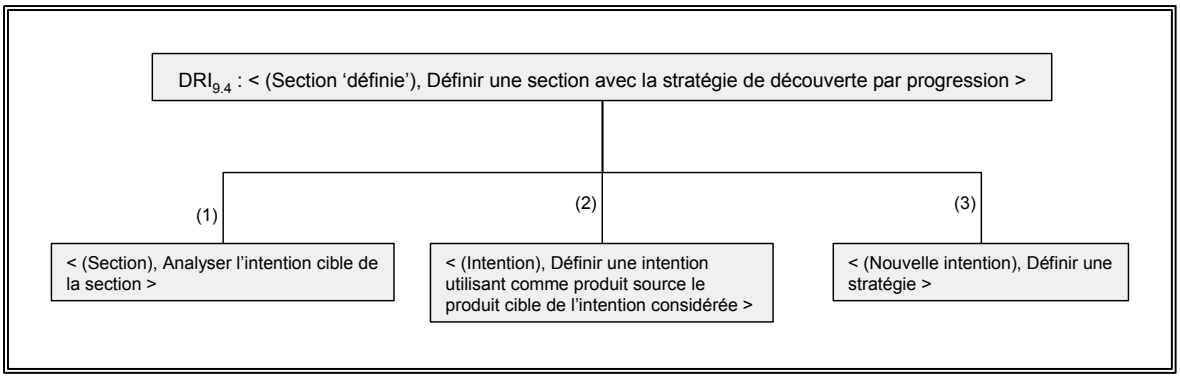


Figure 104. Structure de la $DRI_{9.4}$

4.3.2.5. Réaliser « Définir une section avec la stratégie de découverte par regroupement » ($DRI_{9.5}$)

La $DRI_{9.5}$ permet de définir une nouvelle section à partir d'un ensemble de sections. La stratégie de découverte consiste soit à regrouper les intentions des sections soit les stratégies des sections. La $DRI_{9.5}$ présentée à la Figure 105 est décrite sous forme d'un choix contenant deux alternatives : définir une nouvelle section en regroupant les stratégies ou définir une nouvelle section en regroupant les intentions.

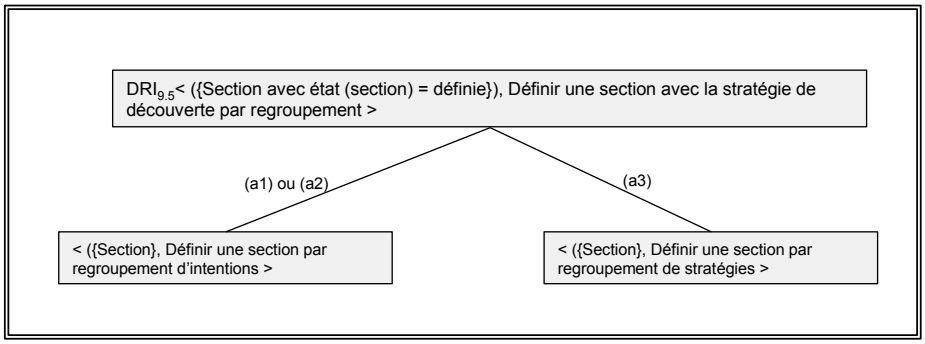


Figure 105. Structure de la $DRI_{9.5}$

La directive *<({Section}), Définir une section par regroupement d'intentions>* permet de regrouper les intentions cibles d'un ensemble de sections si l'ingénieur de méthodes juge que ces intentions sont d'un niveau d'abstraction trop bas par rapport aux intentions des autres sections de la carte de méthode (a1), ou si les sections qui se suivent n'ont pas d'alternative de stratégie et qu'elles représentent une séquence d'intentions à exécuter (a2).

La directive *<({Section}), Définir une section par regroupement de stratégies >* est une directive exécutable qui consiste à regrouper des stratégies de sections ayant les mêmes intentions sources et cibles mais des stratégies différentes qui peuvent être regroupées (a3). Cette directive consiste à supprimer les deux sections et à en définir une nouvelle ayant un nom de stratégie qui englobe les deux dernières.

4.3.2.6. Réaliser « Définir une section avec la stratégie de découverte par décomposition » ($DRI_{9.6}$)

A l'inverse de la $DRI_{9.5}$, la $DRI_{9.6}$ permet de décomposer une section en plusieurs sections. Cette décomposition peut porter sur les intentions d'une section ou sur les stratégies d'une section. Cette directive, présentée à la Figure 106, propose donc un choix contenant deux alternatives pour définir une section : *Par décomposition d'intention* ou *Par décomposition de stratégie*.

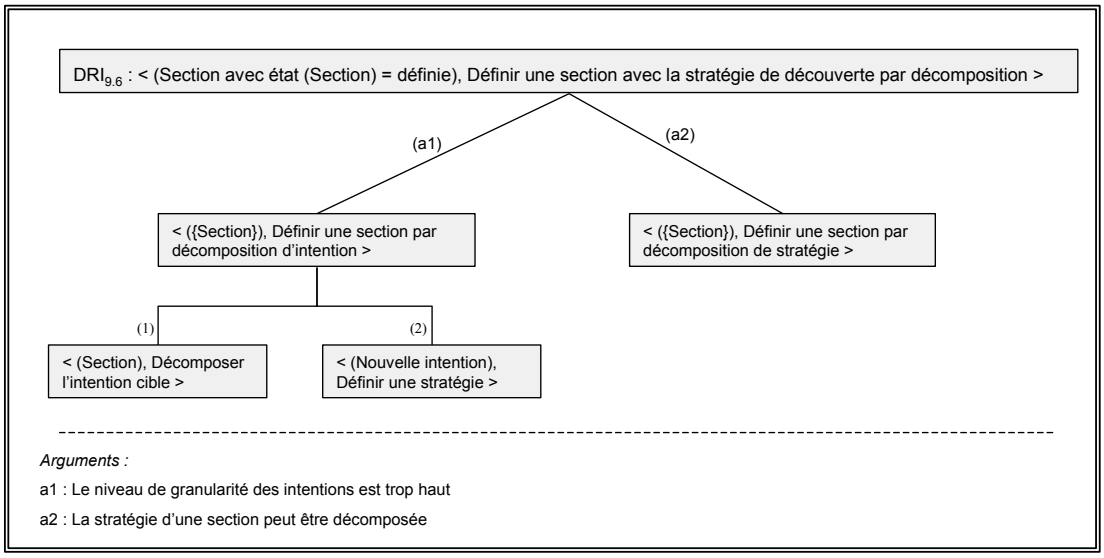


Figure 106. Structure de la $DRI_{9.6}$

Comme le montre cette Figure, la première alternative est une directive plan. Elle permet en premier lieu (1) de définir de nouvelles sections en décomposant l'intention cible de la section en plusieurs intentions si l'intention est jugée de trop haut niveau de granularité (a1). En deuxième lieu (2), elle permet de définir pour chaque couple d'intentions, la stratégie de la nouvelle section. La décomposition de l'intention utilise la formalisation des intentions (verbe + paramètres). La décomposition des paramètres utilise la décomposition des éléments de produit composants la partie de produit cible de l'intention. Les éléments de produit composants peuvent être utilisés comme nouveaux paramètres du verbe.

La deuxième alternative *<({Section}), Définir une section par décomposition de stratégie>* est une directive exécutable qui décompose la stratégie d'une section en plusieurs stratégies permettant de prendre en compte ou de générer une situation particulière.

4.3.2.7. Réaliser « Définir une directive avec la stratégie d'utilisation de formulaire » ($DRI_{9.7}$)

La $DRI_{9.7}$ permet de définir une directive associée à une ou plusieurs sections en utilisant des formulaires prédéfinis. Un des formulaires est présenté à la Figure 107.

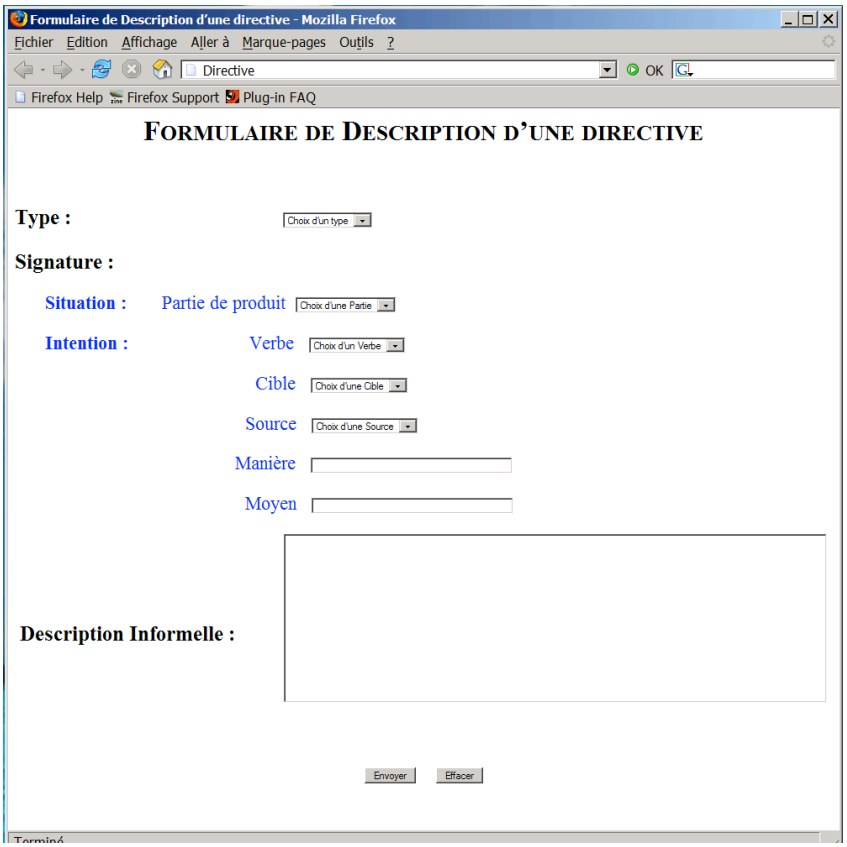


Figure 107. Formulaire de description d'une directive

Ce formulaire permet de décrire les différents éléments caractérisant une directive. Il propose à l'ingénieur de méthodes un ensemble de valeurs prédéfinies pour faciliter sa tâche.

4.3.2.8. Réaliser « Définir une directive avec la stratégie guidée » ($DRI_{9.8}$)

La définition d'une directive avec la stratégie guidée à pour objectif de guider l'ingénieur de méthodes à formaliser la directive quel que soit son type (DRI, DSI, DSS) et quel que soit son type structurel (stratégique, tactique ou simple).

La Figure 108 présente la $DRI_{9.8}$, elle est modélisée par un plan proposant : (1) d'identifier le type de la directive, (2) de définir la signature de la directive et (3) de définir le corps de la directive.

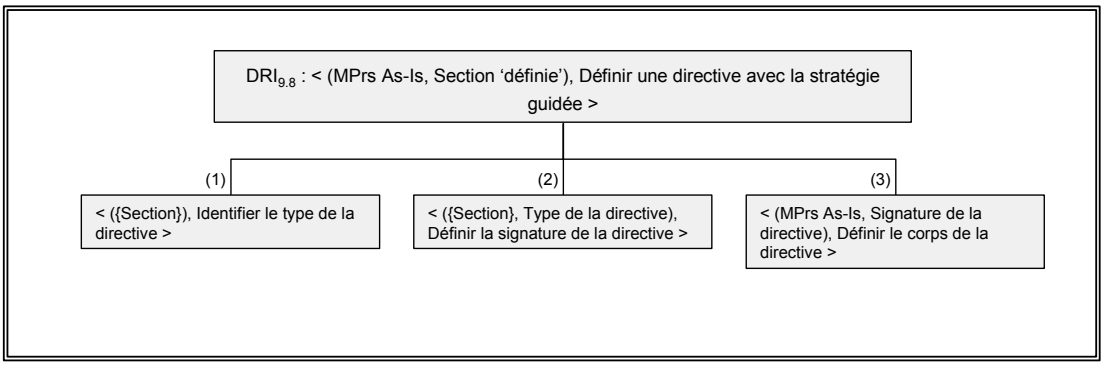


Figure 108. Structure de la $DRI_{9.8}$

(1) < *({Section}), Identifier le type de la directive* > : Cette directive de type choix vise à identifier le type de la directive à définir, elle offre trois alternatives possibles à l'ingénieur de méthode (Figure 109)

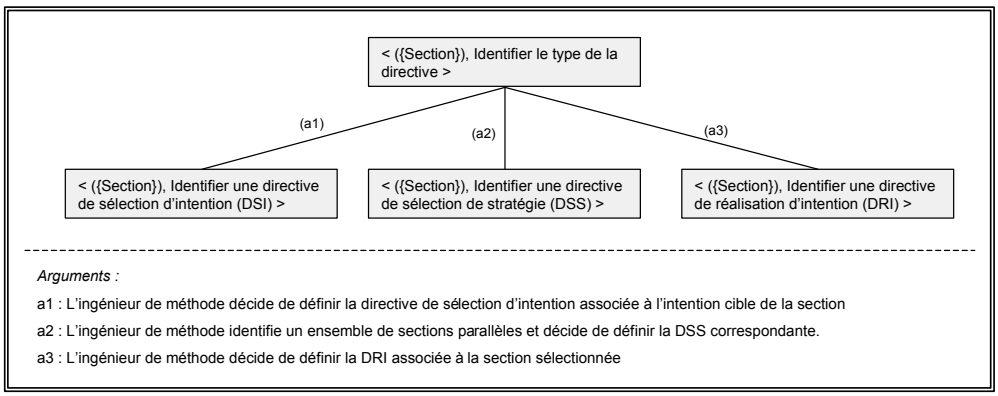


Figure 109. < ({Section}), Identifier le type de la directive >

(2) < *({Section}, Type de la directive), Définir la signature de la directive* > : Cette directive suggère à l'ingénieur de méthodes de définir la signature de la directive à définir en respectant les recommandations de formulation de signature présentées au chapitre 3. La Figure 110 présente la structure de cette directive de type plan. Ce plan propose de (1) Définir la situation de la directive et (2) Définir l'intention de la directive conformément au type de celle-ci.

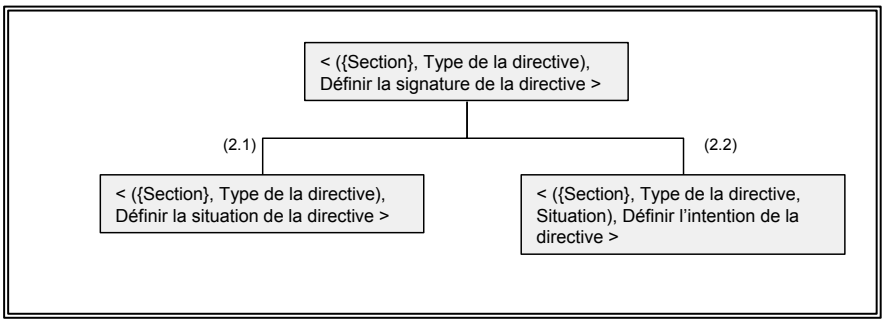


Figure 110. < ({Section}, Type de la directive), Définir la signature de la directive >

(3) < *(MPrs As-Is, Signature de la directive), Définir le corps de la directive* > : L'objectif de cette directive est d'assister l'ingénieur de méthodes dans la définition du corps de la directive à définir, elle lui suggère de (1) Identifier le type de la directive (*exécutable*, *plan* ou *choix*) et (2) définir le corps de la directive selon le type choisi et la structure présentée au chapitre 3. La Figure 111 présente cette directive.

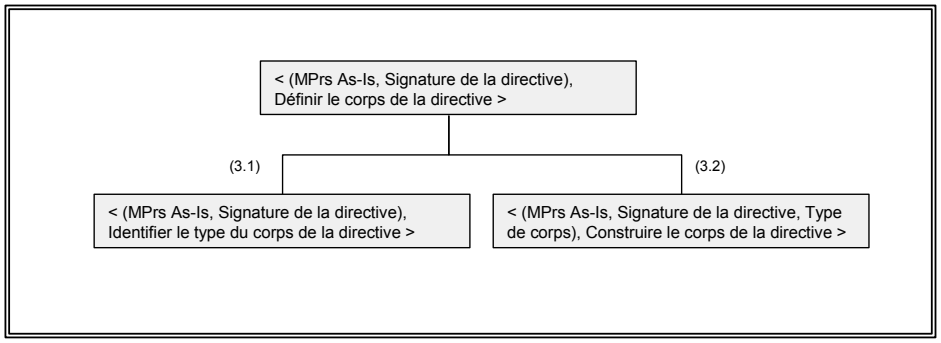


Figure 111. < (MPrs As-Is, Signature de la directive), Définir le corps de la directive >

4.3.2.9. Réaliser « Définir une directive avec la stratégie de modification » ($DRI_{9.9}$)

La modification des sections peut nécessiter la modification des directives associées déjà définies. La $DRI_{9.9}$ aide l'ingénieur de méthodes à modifier les directives en fonction de la modification effectuée sur la section traitée. La modification concerne la DRI de la section traitée, et peut concerner la DSS et la DSI associées à la section si elles existent.

La $DRI_{9.9}$ présentée à la Figure 112 est décrite sous forme d'un plan contenant quatre sous-directives, relatives à la modification des trois types de directives (DRI, DSI, DSS).

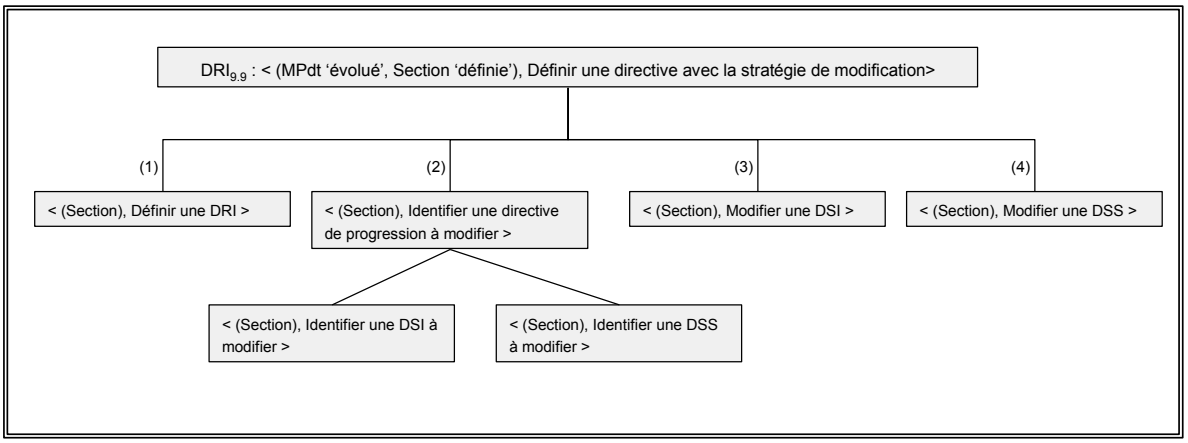


Figure 112. Structure de la $DRI_{9.9}$

4.3.2.10. Réaliser « Définir une section avec la stratégie de correction » ($DRI_{9.10}$)

La $DRI_{9.10}$ aide l'ingénieur de méthodes à corriger les sections définies préalablement. En effet, la définition d'une directive de réalisation d'intention associée à une section peut mettre en évidence le fait que la section ne soit pas correcte et qu'elle doive être décomposée en nouvelles sections.

La Figure 113 présente cette directive, elle est décrite sous la forme d'un plan proposant à l'ingénieur de méthodes de supprimer la section pour laquelle la création de sa DRI a mis en évidence la correction à apporter, de définir les nouvelles sections à partir de la description de cette DRI et finalement de supprimer cette DRI.

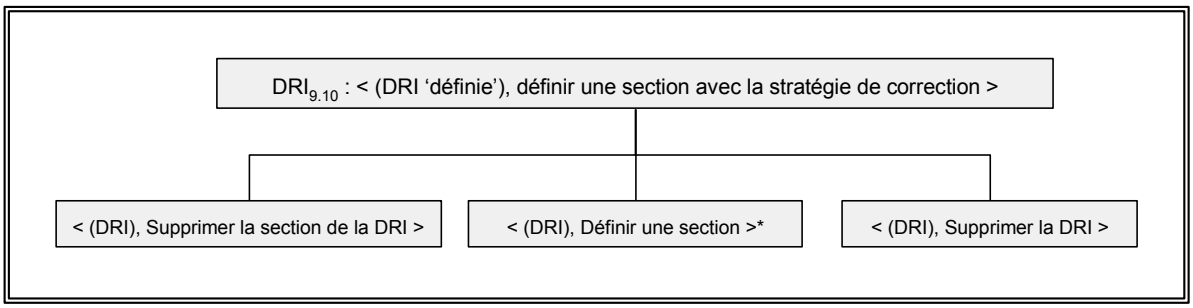


Figure 113. $DRI_{9.10}$: < (DRI 'définie'), définir une section avec la stratégie de correction >

La définition d'une nouvelle section consiste à définir soit des sections ayant des stratégies alternatives si la DRI est de type choix soit des sections successives si la DRI est de type plan.

4.3.2.11. Réaliser « Arrêter avec la stratégie de validation » ($DRI_{9.11}$)

La $DRI_{9.11}$ est une directive informelle, elle permet de terminer le processus de construction du modèle de processus *To-Be* en vérifiant tout d'abord si toutes les directives de la carte ont été définies correctement et on appliquant par la suite les règles de vérification de la carte présentées au chapitre 3. D'éventuelles actions de correction de la carte de méthode ou de ses directives peuvent alors s'avérer nécessaire.

4.3.3. Carte C11 : « Faire évoluer le modèle de processus avec la stratégie orientée patron » (DRI_{11})

Cette section est consacrée à la présentation des Directives de Réalisation d'Intention, associées aux sections de la carte C11. Celle-ci est rappelée à la Figure 114.

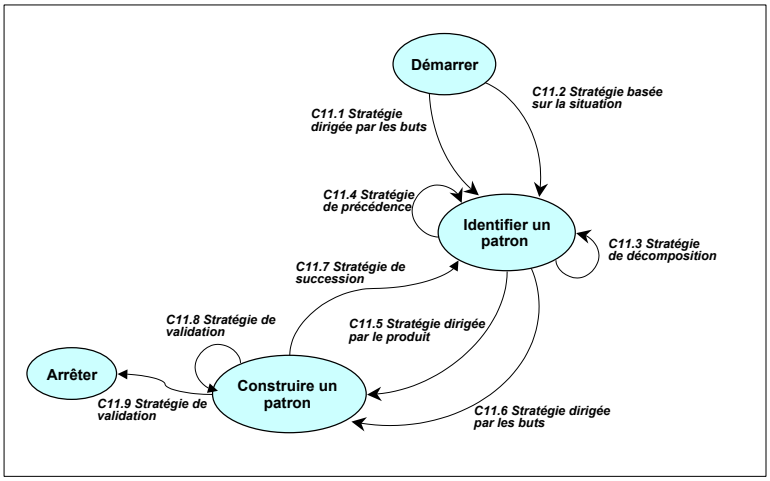


Figure 114. La carte C11

4.3.3.1. Réaliser « Identifier un patron avec la stratégie dirigée par les buts » ($DRI_{11.1}$)

La $DRI_{11.1}$ est associée à la stratégie dirigée par les buts pour l'identification de patron, elle est sélectionnée dans le cas où l'ensemble de but à satisfaire par l'application de la méthode To-Be à été spécifié.

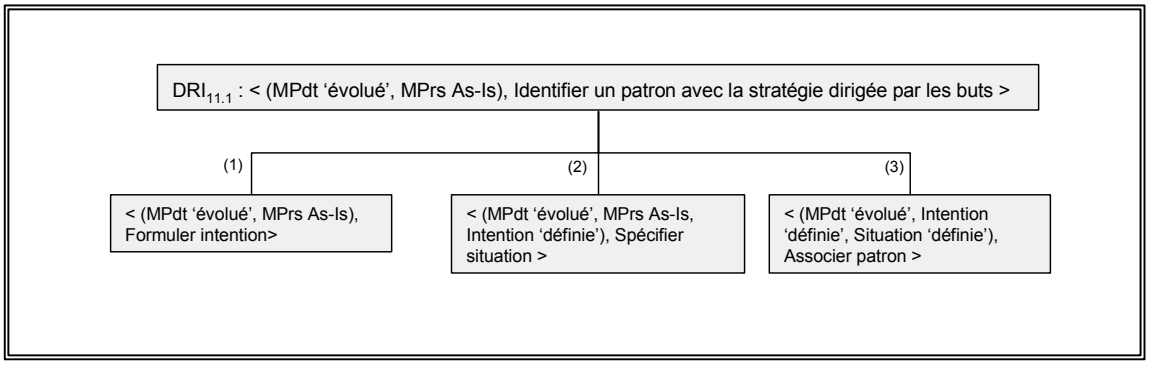


Figure 115. Structure de la $DRI_{11.1}$

Comme le montre la Figure 115, la directive $DRI_{11.1}$ est de type plan. Elle propose à l'ingénieur de méthodes de (1) Formuler l'intention du patron, (2) Spécifier la situation associée à l'intention et (3) Associer un patron au problème défini par le couple (*intention*, *situation*)

- *< (MPdt 'évolué', MPrs As-Is), Formuler intention>* : Cette directive exécutable suggère à l'ingénieur de méthodes de formuler une intention à laquelle on souhaite associer un patron.

Cette formalisation se base sur la liste des buts à satisfaire par la méthode *To-Be*. L'intention est formulée en utilisant la structure (Verbe + paramètres) présentée au chapitre 3.

- < *(MPdt 'évolué', MPrs As-Is, Intention 'définie'), Spécifier situation* > : Cette directive suggère de définir la situation associée à l'intention formulée par la directive précédente. La définition de la situation consiste à définir la partie de produit dont la transformation réalisée par le patron identifié, permet d'obtenir la partie de produit cible de l'intention.

- < *(MPdt 'évolué', Intention 'définie', Situation 'définie'), Associer Patron* > : Cette directive propose d'identifier un patron associé au couple (intention, situation) défini ci-dessus. Elle suggère à l'ingénieur de méthodes de remplir le formulaire de spécification de patron présenté à la Figure 116 pour le patron identifié. Ce formulaire reprend la structure du patron définie au chapitre 3, il est rempli au fur et à mesure de l'exécution des sections de la carteC11. L'ingénieur de méthodes est invité par cette directive à renseigner les champs : *N° du patron* (un numéro séquentiel), *Nom du patron* et *problème*.

N° du patron :
Nom du patron :
Problème : - *Situation :* - *Intention :*
Type Patron : Atomique / Composé **Patron composé :** **Patron composant :**
Solution : **1.** **2.** **….** **n.**

Figure 116. Formulaire de spécification de patron

4.3.3.2. Réaliser « Identifier un patron avec la stratégie basée sur la situation » ($DRI_{11.2}$)

La $DRI_{11.2}$ est associée à la section C11.2 de la carte C11. Elle permet à l'ingénieur de méthodes d'identifier un patron en se basant sur les situations typiques pour la construction du produit souhaité par la méthode *To-Be*. La Figure 117 présente cette directive qui est une directive tactique de type

plan. Ce plan propose à l'ingénieur de méthodes de (1) Identifier une situation, (2) Spécifier l'intention associée à la situation identifiée et (3) Associer un patron au couple (*situation*, *intention*).

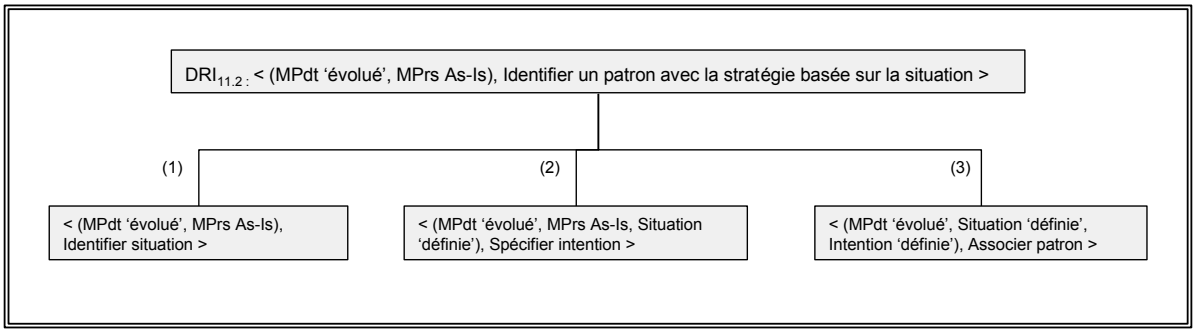


Figure 117. Structure de la $DRI_{11.2}$

- *< (MPdt 'évolué', MPrs As-Is), Identifier situation >* : Cette directive suggère à l'ingénieur de méthodes d'identifier une situation typique correspondant à un problème récurrent dans la construction du produit. La définition de cette situation revient à spécifier la partie de produit associée au problème.
- *< (MPdt 'évolué', MPrs As-Is, Situation 'définie'), Spécifier intention >* : Cette directive invite l'ingénieur de méthodes à spécifier l'intention correspondante au problème et à la situation définis par l'application de la directive précédente.
- *< (MPdt 'évolué', Situation 'définie', Intention 'définie'), Associer patron >* : Cette directive suggère d'associer un patron au couple (situation, intention) défini. L'ingénieur de méthodes est alors invité à créer un formulaire de spécification de patron et à remplir les champs (*N° du patron*, *Nom du patron* et *Problème*) pour le patron identifié.

4.3.3.3. Réaliser « Identifier un patron avec la stratégie de décomposition » ($DRI_{11.3}$)

La $DRI_{11.3}$ est associée à la section C11.3 de la carte C11. Cette directive est sélectionnée pour assister l'ingénieur de méthodes dans l'identification de patron par décomposition de patron existant. La $DRI_{11.3}$ est décrite par la directive plan présentée à la Figure 118.

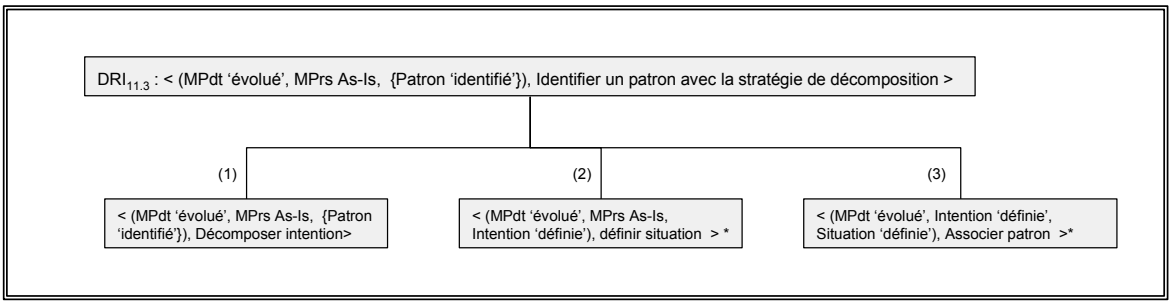


Figure 118. Structure de la $DRI_{11.3}$

- *< (MPdt 'évolué', MPrs As-Is, {Patron 'identifié'}), Décomposer intention> :* Cette directive suggère de décomposer l'intention de patron dont le problème est complexe et nécessitant une décomposition pour pouvoir lui associer une solution. La décomposition de l'intention utilise la formalisation des intentions (verbe, paramètres). La décomposition des paramètres utilise la décomposition des éléments de produit composée. Les éléments du produit composant peuvent être utilisés comme nouveau paramètre du verbe. (Par exemple l'intention *Identifier un patron* peut être décomposer en deux intentions *Identifier problème* et *Identifier solution* car l'élément de produit *patron* est composée d'un *problème* et d'une *solution*).

- *< (MPdt 'évolué', MPrs As-Is, Intention 'définie'), définir situation >* : Cette directive invite l'ingénieur de méthodes à définir la situation associée à l'intention définie par la directive précédente. La définition de cette situation s'effectue par décomposition de la partie de produit décrivant la situation du patron composite.

  - < (MPdt 'évolué', Intention 'définie', Situation 'définie'), Associer patron >. Cette directive suggère d'associer un patron au couple (*Situation*, *Intention*) découvert par décomposition. L'ingénieur de méthodes est alors invité à créer un formulaire de spécification de patron et à remplir les champs : *N° du patron*, *Nom du patron*, *Problème, Type* et *Patron composé* (Le patron qu'on a décomposé pour obtenir ce patron). Le formulaire correspondant au patron composé doit alors être mis à jour pour y inscrire les patrons composants.

4.3.3.4. Réaliser « Identifier un patron avec la stratégie de précédence » ($DRI_{11.4}$)

La $DRI_{11.4}$ est associée à la section C11.4 de la carte C11. Le but de cette directive est d'identifier un patron avec la stratégie de précédence. Cette stratégie consiste à analyser la situation d'un patron identifié pour identifier un patron dont l'application permet de créer cette situation. La $DRI_{11.4}$ est une directive tactique de type plan, La Figure 119 présente cette directive.

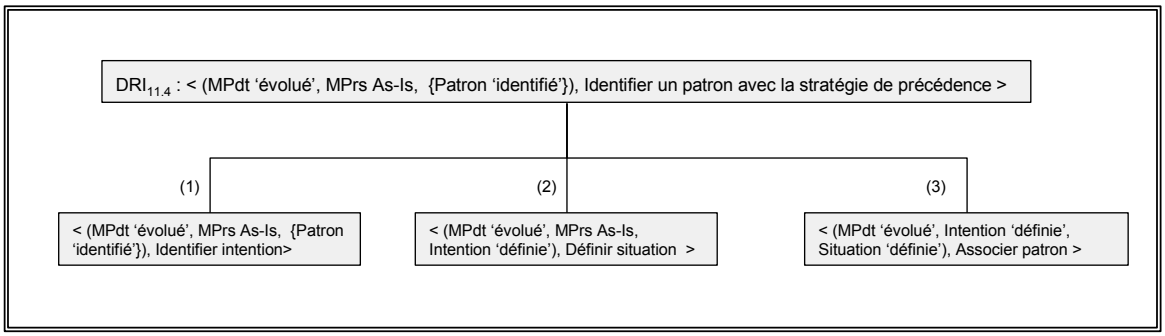


Figure 119. Structure de la $DRI_{11.4}$

- *< (MPdt 'évolué', MPrs As-Is, {Patron 'identifié'}), Identifier intention>* : Cette directive suggère à l'ingénieur de méthodes d'analyser la situation d'un patron identifié pour identifier une intention, non associée à un patron déjà identifié, dont la partie de produit cible correspond à la situation analysée.
- *< (MPdt 'évolué', MPrs As-Is, Intention 'définie'), Définir situation >* : Cette directive suggère de définir la situation associée à l'intention formulée par la directive précédente. La définition de la situation consiste à définir la partie de produit dont la transformation permet d'obtenir la partie de produit cible de l'intention.
- *< (MPdt 'évolué', Intention 'définie', Situation 'définie'), Associer patron >* : Cette directive suggère d'associer un patron au couple (intention, situation) défini. L'ingénieur de méthodes est alors invité à créer un formulaire de spécification de patron et à remplir les champs (*N° du patron*, *Nom du patron* et *Problème*) .

4.3.3.5. Réaliser « Construire un patron avec la stratégie dirigée par le produit » ($DRI_{11.5}$)

La $DRI_{11.5}$ est associée à la section C11.5. L'objectif de cette directive est de formuler la solution d'un patron identifié en formalisant le processus permettant la production de la partie de produit décrit dans son intention à partir de ceux décrit dans sa situation. La Figure 120 présente la structure de la $DRI_{11.5}$ qui est une directive tactique de type plan. Ce plan propose de (1) Définir les étapes composant la solution du patron et (2) Décomposer les étapes identifiées en actions de transformation de produit.

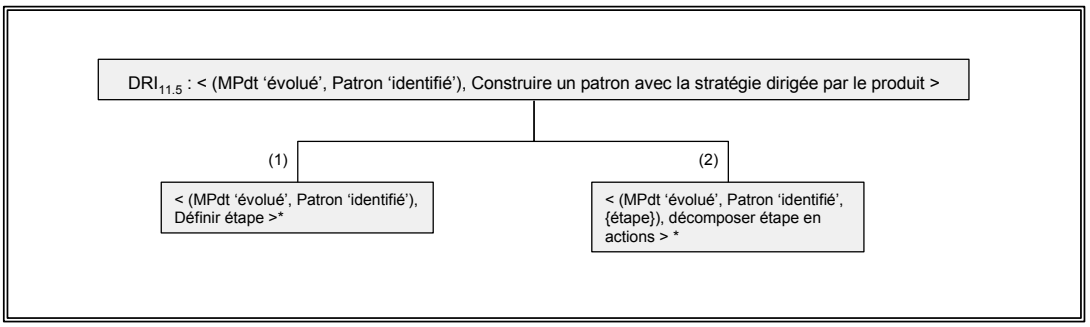


Figure 120. Structure de la $DRI_{11.5}$

- *< (MPdt 'évolué', Patron 'identifié'), Définir étape >* : Cette directive suggère à l'ingénieur de méthodes de décomposer le processus décrivant la construction de produit souhaité par l'application de patron en étapes. Chaque étape correspond à la production d'une situation intermédiaire de ce produit.
- *< (MPdt 'évolué', Patron 'identifié', {étape 'identifié'}), Décomposer étape en actions >* : Cette directive propose de décomposer les étapes identifiées en un ensemble d'action élémentaire de transformation de produit. L'ingénieur de méthodes est invité par la suite à compléter le formulaire de spécification de patron correspondant en remplissant la partie solution de celui-ci.

4.3.3.6. Réaliser « Construire un patron avec la stratégie dirigée par les buts » ($DRI_{11.6}$)

La $DRI_{11.6}$ est associée à la section C11.6. L'objectif de cette directive est de définir la solution d'un patron identifié en appliquant la stratégie dirigée par les buts. Cette stratégie permet de réduire l'intention du patron en un ensemble d'actions atomique permettant de l'opérationaliser.

La $DRI_{11.6}$ est une directive tactique de type plan. Ce plan est présenté à la Figure 121, il propose à l'ingénieur de méthodes de (1) Décomposer l'intention du patron en un ensemble de buts opérationalisable et (2) Associer à chaque but opérationalisable un ensemble d'action d'ingénierie permettant la construction du produit souhaité par l'application du patron.

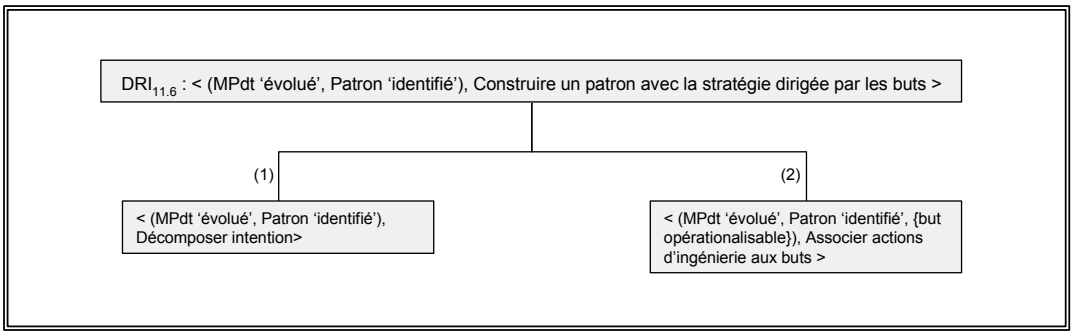


Figure 121. structure de la $DRI_{11.6}$

4.3.3.7. Réaliser « Identifier un patron avec la stratégie de succession» ($DRI_{11.7}$)

La $DRI_{11.7}$ est associée à la section < *Construire un patron, Identifier un patron, Stratégie de succession*> de la carte C11. Cette directive suggère de considérer la partie de produit obtenue par l'application d'un patron défini comme situation potentielle pour l'identification d'un nouveau patron.

La Figure 122 présente la structure de la $DRI_{11.7}$ qui est une directive tactique de type plan. Ce plan propose de (1) Définir la situation du patron, (2) Définir l'intention associée à la situation définie et (3) Associer un patron au couple (Situation, Intention) et l'inscrire avec un formulaire de spécification de patron.

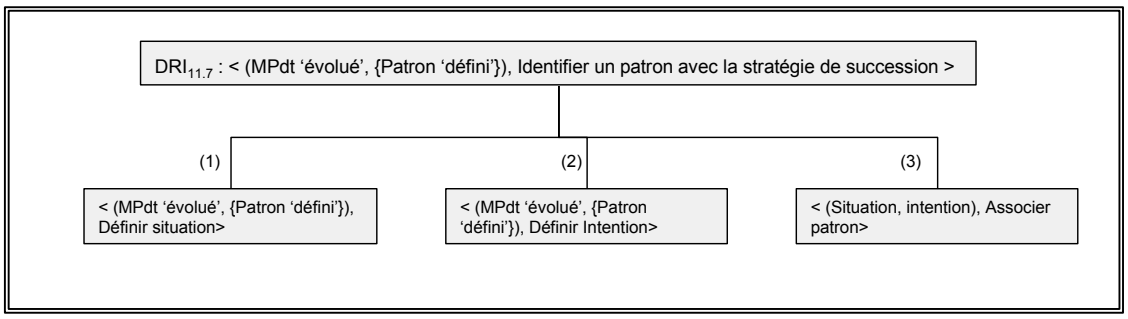


Figure 122. Structure de la $DRI_{11.7}$

4.3.3.8. Réaliser « Construire un patron avec la stratégie de validation » ($DRI_{11.8}$)

La $DRI_{11.8}$ est associée à la stratégie de validation pour atteindre l'intention *Construire un patron*. Elle est sélectionnée pour vérifier la validité des patrons construits en s'assurant qu'ils respectent les règles de construction de patron définies au chapitre 3.

4.3.3.9. Réaliser « Arrêter avec la stratégie de complétude » ($DRI_{11.9}$)

Cette directive est informelle, elle permet d'arrêter le processus d'évolution du modèle de processus en vérifiant la cohérence globale des différents patrons définis.

5. Conclusion

Dans ce chapitre nous avons présenté le modèle de processus d'évolution de méthodes. Ce modèle permet d'assister l'ingénieur de méthodes pour conduire le projet d'évolution d'une méthode existante (*la méthode As-Is*) vers une nouvelle méthode (*la méthode To-Be*) satisfaisant de nouveaux besoins d'ingénierie. Ce modèle de processus se base sur le méta-modèle de méthode proposé au chapitre 3 et peut être appliqué à toute méthode d'ingénierie de SI instance de ce méta-modèle.

L'utilisation du modèle de la carte pour la formalisation du modèle de processus de cette démarche lui confère plusieurs avantages. En effet, la démarche proposée est facilement extensible notamment par l'ajout de nouvelles stratégies ou /et de nouvelles intentions.

CHAPITRE 6 : APPLICATION A LA METHODE LYEE

1. Introduction :

Pour prouver la faisabilité de la méthode *MIME* nous l'avons appliqué pour faire évoluer la méthode de développement de logiciel Lyee. Ceci est réalisé dans le cadre du projet « *Lyee International Research Project* ».

Ce chapitre est organisé comme suit: Une présentation sommaire de la méthode Lyee constitue la première partie de ce chapitre. La deuxième partie présente, quant à elle, le processus de rétro-ingénierie de Lyee permettant de formaliser ses modèles de produit et de processus. Enfin, la troisième partie présente le processus d'évolution de la méthode Lyee conduit par la démarche d'évolution proposée au chapitre 5.

2. Présentation de la méthode Lyee

Lyee (Governementa**L** Methodolog**Y** for softwar**E** providenc**E**) est une méthode de développement de logiciels inventée par F. Negoro [Negoro 01a], [Negoro 01b]. Cette méthode a la spécificité de réduire significativement le cycle de développement de logiciels en éliminant la phase de programmation et en permettant la génération automatique de code à partir des spécifications déclaratives des besoins logiciels.

LyeeALL est l'outil *CASE* supportant la méthode Lyee. Comme le montre la Figure 123, *LyeeALL* est composé de :

- un *cadre de référence* permettant de structurer les programmes *Lyee*,
- un *moteur Lyee* pour contrôler l'exécution des programmes et,

- un *mécanisme de génération* pour générer automatiquement les programmes à partir de spécifications déclaratives des besoins logiciels dans le langage de programmation souhaité (C, Java, Cobol ou VB)

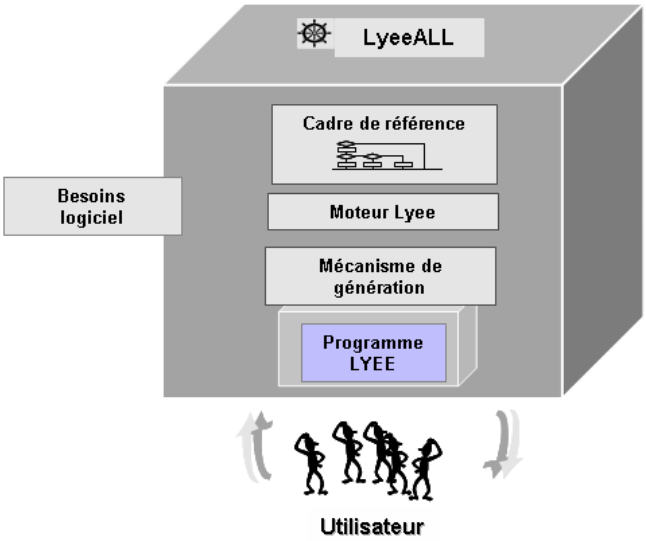


Figure 123. Architecture de LyeeALL

La description initiale de la méthode Lyee est centrée sur l'utilisation de l'outil *LyeeALL* [LyeeALLmanuel], [Poli 02], elle met l'accent sur les différentes étapes à suivre pour spécifier et saisir les besoins logiciels et générer les applications avec *LyeeALL*.

Tout programme généré par *LyeeALL* est une instance d'un modèle générique, le *Process Route Diagram (PRD)*. Le *PRD* fournit le *cadre de référence* pour structurer les programmes Lyee. Comme le montre la Figure 124, le *PRD* est un graphe dirigé dans lequel les nœuds sont des instances d'une structure de base appelée *Scenario Function (SF)*. Un *SF* est une agrégation de trois sous-structures appelées *Pallet* : *W04*, *W02* et *W03*.

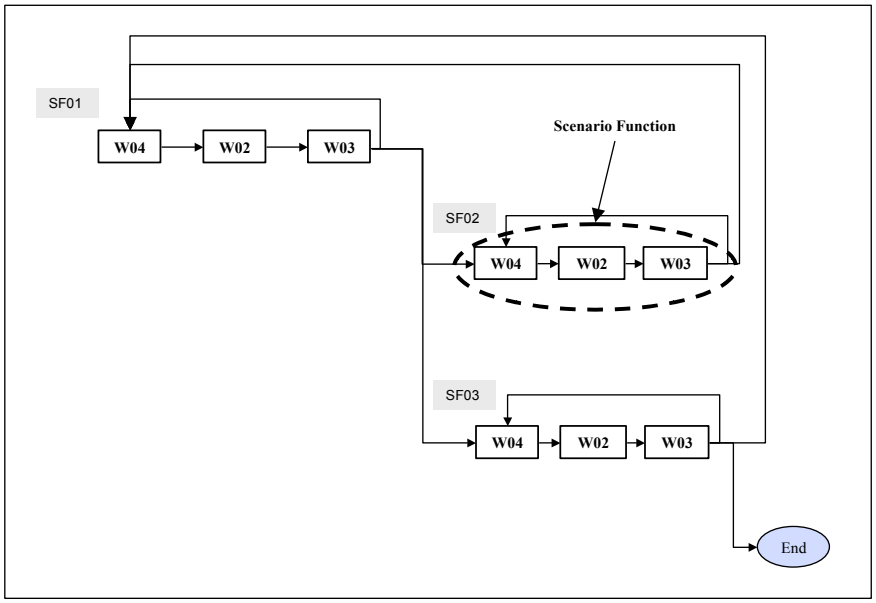


Figure 124. Structure d'un PRD

Chaque *pallet* est décomposé à son tour en plusieurs sous-structures appelées *Vecteurs*. Il existe un ensemble particulier de vecteurs attachés à chaque *pallet* (Figure 125).

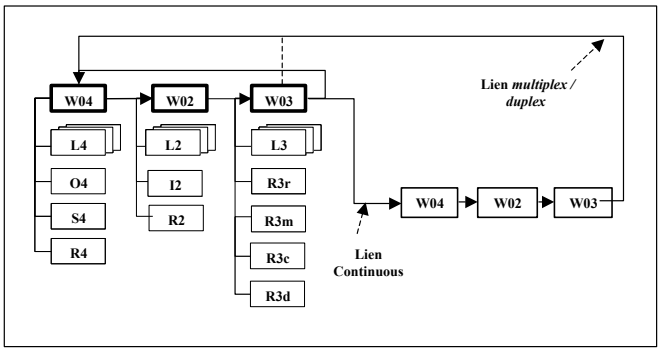


Figure 125. Structure des Pallets

- au *pallet W04* sont attachés quatre vecteurs : *L4*, *O4*, *S4*, et R4.
- au *pallet W02* sont attachés trois vecteurs : *L2*, *I2* et *R2*.
- au *pallet W03* sont attachés trois vecteurs *L3, R3r et R3k*.

Les vecteurs cités ci-dessus sont classés en trois types :

- les *Signification Vectors* (*L4*, *L2*, L3) permettent le calcul des *words* (variable programme), un vecteur *L2* est généré pour chaque *input word* (variable programme en entré), un vecteur *L3* et un vecteur *L4* sont générés pour chaque *output word* (variable programme en sortie).
- les *Routing Vectors* (*R4*, *R2*, $R3_r$, $R3_m$, $R3_c$ et $R3_d$) assurent le branchement d'un *pallet* vers un autre durant l'exécution des programmes générés par *LyeeALL*. Le vecteur R4 correspond au branchement du *pallet* W04 vers le *pallet* W02. Le vecteur R2 assure le branchement du *pallet* W02 vers le *pallet* W03. Le vecteur R3r assure la réitération dans le même SF

(branchement de W03 vers W04). Les vecteurs R3c, R3m et R3d assurent le branchement d'un SF vers un autre, il correspondent au lien *continuous (R3c)*, *multiplex (R3m)* et *duplex (R3d)* entre les SF d'un PRD (Figure 125).

- Les *Actions Vectors* (*O4*, *S4* et *I2*) contrôlent les échanges entrée/sortie dans les programmes générés par *LyeeALL*, telle que l'opération de lecture d'un champ d'une interface graphique ou l'opération d'écriture dans un fichier ou une base de données. I2 est utilisé à la capture des entrées, O4 et S4 sont utilisés pour la communication des résultats.

Les trois types de vecteurs présentés ci-dessus sont générés à partir de trois types de *words* : les *domain words* pour la génération des *signification vecteurs*, les *routing words* pour la génération des *routing vecteurs* et les *actions words* pour la génération des *actions vecteurs*.

Le *moteur Lyee* contrôle l'exécution des programmes générés par *LyeeALL* en exécutant la *Tense Control Function* φ. φ assure le contrôle des programmes à travers les *Pallets Functions*, une par pallet (*φ4* pour *W04*, *φ2* pour *W02* et *φ3* pour *W03*) . Φ est définie alors comme suit φ = φ4 +φ2 +φ3, tels que :

$\varphi 4 = (\{L4,j\}+\{O4,r_o\}+\{S4,r_s\}+R4)$

$\varphi 2 = (\{L2,i\}+\{I2,r_i\}+ R2)$

$\varphi 3 = (\{L3,j\}, R3r, R3d, R3m, R3c)$

φ4 contrôle le calcul des *output words* et leur présentation physique, φ2 assure la capture des *input words*. φ3 vérifie les conditions dans lesquelles le calcul des *output Words* est possible.

3. Présentation du projet « Lyee International Research Projet »

Le propos du projet « Lyee International Research Projet » est d'effectuer des travaux de recherche autour de la méthode *Lyee*. Les objectifs de ce projet sont multiples : il vise en premier lieu l'étude, l'évaluation de la méthode Lyee et sa comparaison avec les méthodes d'ingénierie de SI existantes. En deuxième lieu, il vise à apporter des améliorations à cette méthode pour pallier ses limites, élargir son champ d'application et la promouvoir dans le milieu industriel.

Comme on peut le constater, à travers la section précédente de ce mémoire, la méthode Lyee est comprise et présentée par ses éditeurs dans des termes opérationnels de bas niveau sans la proposition d'une vue générale de celle-ci. En effet, la méthode Lyee ne repose pas sur des modèles permettant de la formaliser ce qui rend difficile sa compréhension et son utilisation par des novices.

Un autre problème sous-jacent la méthode Lyee réside dans le fait que les besoins des utilisateurs sont exprimés dans *LyeeALL* dans des termes de bas niveau tels que le contenu des écrans ou des

accès à une base de données ce qui a posé de nombreux problèmes dans les projets utilisant cette méthode. L'écart entre les besoins tels que les utilisateurs peuvent les exprimer et les entrées requises pour *LyeeALL* est trop important.

La résolution des deux problèmes cités ci-dessus est l'objet de notre participation dans le projet Lyee. Notre participation a un double objectif : Le premier concerne la *ré-ingénierie* de la méthode *Lyee* par la formalisation de ses modèles de produit et de processus. Le deuxième concerne *l'évolution* de cette méthode pour proposer une nouvelle méthode supportant le développement de logiciels en deux étapes : l'ingénierie des besoins et la génération automatique de code. La première étape se base sur une adaptation des travaux d'ingénierie des besoins de notre groupe; la deuxième étape est assurée par *LyeeALL*.

Nous montrons dans les deux sections suivantes l'application de MIME au cas de Lyee.

4. Rétro-ingénierie de la méthode Lyee

En l'absence de modèle formalisant la méthode Lyee, une étape de rétro-ingénierie de celle-ci est nécessaire avant de procéder à son évolution. Le modèle de processus de rétro-ingénierie présenté au chapitre 4 est utilisé à cette fin. La Figure 126 rappelle la carte de rétro-ingénierie proposée.

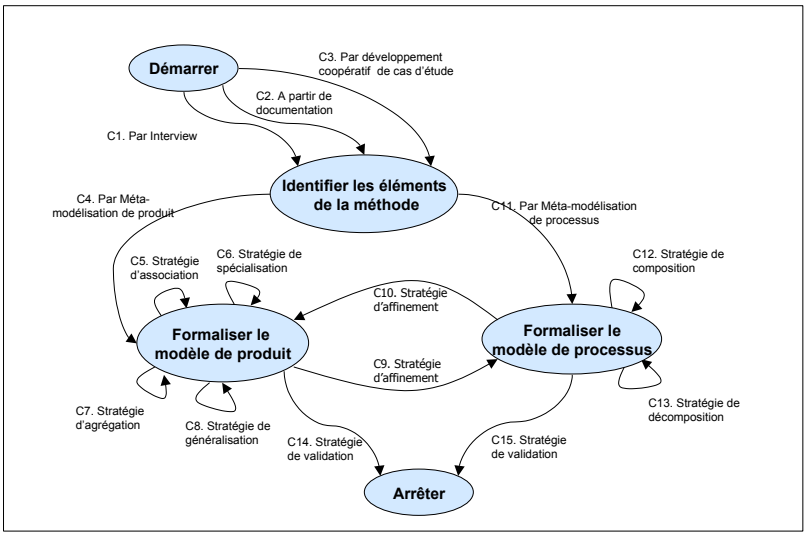


Figure 126. La carte de Rétro-ingénierie

- *Atteindre l'intention Identifier les éléments de la méthode*

La première intention à atteindre dans cette carte est *Identifier les éléments de la méthode*. Au démarrage du projet la transmission du savoir s'est faite par l'échange de documentation entre notre équipe et les éditeurs de la méthode. La stratégie *A partir de documentation* a donc été sélectionnée. L'exécution de la directive associée à cette stratégie nous a permis d'avoir une première vision de la méthode. Les documents disponibles de la méthode Lyee étant très techniques et présentant

essentiellement des exemples d'application de celle-ci et les fragments de code générés par *LyeeALL*, il nous a été impossible de dégager une vision globale de la méthode. L'exécution d'une autre section pour l'identification des éléments de la méthode s'est avérée nécessaire. La DSS_1 de la carte de rétro-ingénierie suggère d'appliquer la stratégie *Par développement coopératif de cas d'étude*.

Le plan décrivant la DRI_3 associée à cette stratégie suggère d'organiser en premier lieu une séance de travail, ceci a été réalisé par l'organisation d'un workshop d'une semaine rassemblant des experts de la méthode Lyee et des experts de notre équipe. En deuxième lieu, la directive suggère de rédiger un cas d'étude, ceci a été fait par la rédaction d'un ensemble de cas d'étude de complexité croissante. La troisième intention à atteindre dans le plan de la DRI_3 est l'identification d'éléments de méthode durant le déroulement des séances de travail. Ceci a été effectivement possible et a débouché sur l'identification des éléments de la méthode présentés au Tableau 19. Ce tableau associe à chaque élément de produit identifié l'élément de processus permettant sa production ou sa transformation.

Elément de produit	Elément de processus
PRD	Spécifier les *PRDs* d'une application
SF	Spécifier les *SFs* composant chaque *PRD*
Pallet (W03, W02, W04)	
Logical Unit	Spécifier les *logicals units* associée à chaque pallet
Domain Word	Specifier les *domain words*
Action Word (PRD1, POP1, PCL1, PCR1, PCR2, PBOX et PWT1)	Spécifier les *actions words*
Routing word (PNTN, PNTR, PNTD, PNTC, PNTA, PNTM, PNTE)	Spécifier les *routing words*
-	Définir l'interface graphique de l'utilisateur et la base de données
-	Générer les programmes par LyeeALL

Tableau 19. les éléments identifiés de la méthode Lyee

Les trois premiers éléments de produit dans le tableau (*PRD*, *SF* et *pallet*) sont présentés à la section 2. L'élément *Logical Unit* représente un ensemble cohérent de *words* utilisés dans le même processus (lecture ou écriture) et correspondant à un même support physique (base de données, fichier, écran, etc.) utilisé par un programme généré par *LyeeALL*. Les trois éléments de produit (*domain word*, *action word* et *routing word*) permettent la génération des vecteurs contrôlant l'exécution des programmes générés par *LyeeALL*. Les deux dernières lignes du tableau présente des éléments de

processus auquel on n'a pas pu associer directement d'éléments de produit puisqu'elles concernent la phase de l'implémentation et de l'intégration de l'application générée par *LyeeALL*.

Comme le montre le Tableau 19, sept types d'*action word* ont été identifiés par l'application de la stratégie *Par développement coopératif de cas d'étude* utilisée chacun pour contrôler un type particulier d'action physique d'entrée/sortie :

- PRD1 : permettant la génération du vecteur pour la lecture des *words* dans un écran ou une base de données.
- POP1 : pour ouvrir un fichier.
- PCL1 : pour fermer un fichier.
- PWT1 : permettant la génération du vecteur pour l'affichage des *words* dans un écran ou l'écriture des words dans la base de donnée.
- PCR1, PCR2 : pour vider les zones en lecture ou en écriture.
- PBOX : pour vider les tampons mémoires utilisés par l'application

De la même manière, 7 types de *routing word* ont été identifiés, chacun correspond a un type particulier de branchement entre *pallets* :

- PNTN: pour la génération des vecteurs permettant de passer le contrôle de la *pallet function* φ4 vers φ2 et de φ2 vers φ3 (R2 et R3)
- PNTR : pour la génération du vecteur permettant de passer le contrôle de la *pallet function* φ3 vers φ4 (R3r)
- PNTD : pour la génération du vecteur permettant de passer le contrôle de la *pallet function* φ3 de SF_i vers φ3 de SF_j avec un lien duplex (R3d).
- PNTC : pour la génération du vecteur permettant de passer le contrôle de la *pallet function* φ3 de SF_i vers φ4 de SF_j avec un lien continuous (R3c).
- PNTA : pour la génération du vecteur permettant d'atteindre un SF_j suivi d'un lien *duplex*
- PNTM : pour la génération du vecteur permettant de passer le contrôle de la pallet function φ3 de SF_i vers φ4 de SF_j avec un lien *multiplex* (R3m).
- PNTE : pour la génération du vecteur permettant de lier φ3 de SF_i au point *End*.

La carte de rétro-ingénierie nous suggère de réaliser la formalisation des modèles de produit et de processus de manière conjointe. Dans un souci de lisibilité nous présentons la construction du modèle de produit et celle du modèle de processus séparément.

- *Atteindre l'intention Formaliser le modèle de produit*

Pour atteindre l'intention *Formaliser le modèle de produit* une seule stratégie est proposée par la carte de rétro-ingénierie *Par méta-modélisation de produit*. La DRI_4 , rappelée à la Figure 127, est associée à cette stratégie.

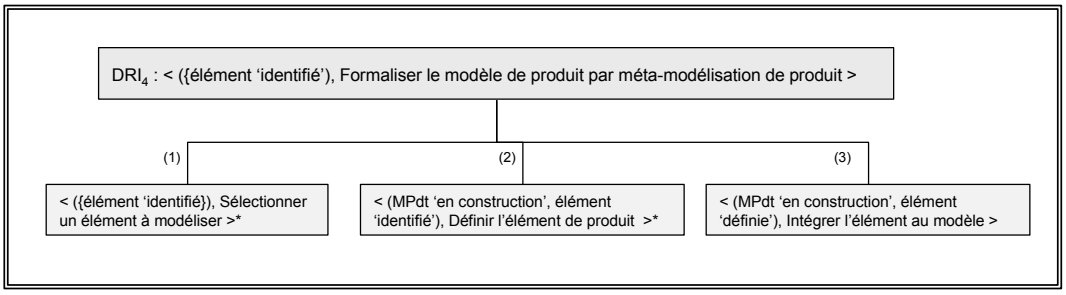


Figure 127. structure de la DRI_4

Le plan proposé par cette directive nous suggère de partir de la liste des éléments de la méthode Lyee identifiés par la réalisation de l'intention *Identifier les éléments de la méthode* (Tableau 19) pour formaliser son modèle de produit. Ce plan propose d'exécuter pour tous les éléments de produit identifiés les trois directives suivantes : (i) sélectionner un élément à modéliser, (ii) définir l'élément de produit identifié et (iii) intégrer l'élément au modèle.

Le modèle de produit de la méthode Lyee est alors construit comme suit :

- Le PRD est le premier élément identifié de la méthode Lyee, la DRI_4 nous suggère de créer une classe pour modéliser cet élément, de définir son attribut (*PRDName*) et de l'intégré au modèle de produit en cours de construction.
- Le PRD est défini comme une succession de SF, une classe modélisant l'élément *Scenario Function* est alors défini et intégré au modèle de produit en cours de construction avec un lien d'agrégation le reliant à l'élément PRD.

- Le troisième élément de produit identifié dans le Tableau 19 est *pallet*, comme nous l'avons spécifié dans la section 2, un SF est composé de trois *pallets* W04, W03 et W02. L'application de la DRI_4 permet de créér trois classes pour modéliser ces trois types de *pallets* et de les intégrer au modèle de produit en cours de construction avec des liens d'agrégation les reliant à l'élément *Scenario Function.* La *stratégie de généralisation* de la carte de rétro-ingénierie (Figure 129) est par la suite appliquée pour créer la classe modélisant l'élément *Pallet,* définir son attribut (*PalletID*) et l'intégrer au modèle de produit en cours de construction en moyennant un lien *Est-un* le reliant à W02, W03 et W04. La Figure 128 montre le modèle de produit de la méthode Lyee obtenu à se stade de processus en utilisant le formalisme UML [UML 03].

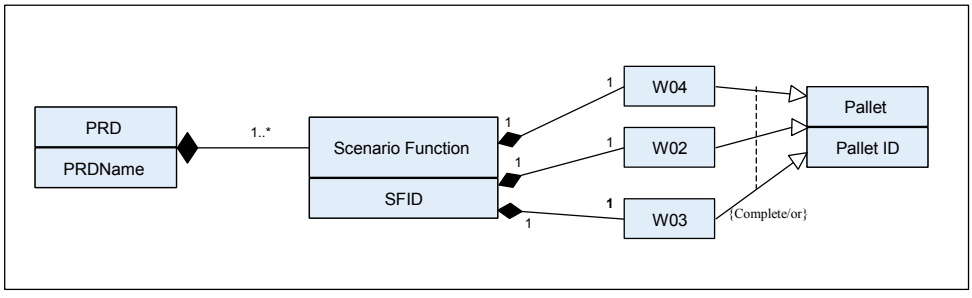


Figure 128. Le modèle de produit en cours de construction

- Le quatrième élément de produit identifié dans le Tableau 19 est *logical unit,* l'application de la DRI_4 permet de créer une classe modélisant cet élément, de définir ses attributs (*LogicalID*, *Device*) et de l'intégrer au modèle de produit en cours de construction.

- Le cinquième élément de produit identifié est *domain word*, l'application de la DRI_4 permet de créer la classe modélisant cet élément, de définir ces attributs (*L3-condition*, *L4-formula*, *Name* et *Domain*) et de l'intégrer au modèle de produit en cours de construction.

- Le sixième élément de produit identifié est a*ctions word,* d'une manière similaire aux précédents l'application de la DRI_4 permet de créer une classe modélisant cet élément et de l'intégrer au modèle de produit en cours de construction. Dans le Tableau 19 on a identifié sept types d'*action word* (*PRD1*, *POP1*, *PCL1*, *PCR1*, *PCR2*, *PBOX* et *PWT1*), la DSS2 de la carte de rétro-ingénierie suggère d'appliquer la *Stratégie de spécialisation* dans cette situation. Celle-ci est alors appliquée pour créer une classe modélisant chaque type identifié et d'intégrer les classes définies dans le modèle de produit en cours de construction en moyennant un lien d'héritage les liants à la classe *action word* (Figure 129).

- Le dernier élément de produit identifié est *routing word*, cet élément est alors défini en appliquant la DRI_4 par une classe ayant (*NextPalletID*) comme attribut. Sept types de *routing word* ont été identifiés dans le Tableau 19 (*PNTN, PNTR, PNTD, PNTC, PNTA, PNTM, PNTE*), chacun de ces words correspond à un type particulier de branchement entre *pallets*. Cependant, nous avons identifié deux cas génériques de branchement entre *pallets*: le branchement entre les *pallets* d'un même SF et le branchement entre les *pallets* d'un SF vers un autre. La *Stratégie de spécialisation* a été alors appliquée pour créer deux types de *routing word* correspondant successivement à ces deux cas de branchement : *IntraSF* et *InterSF*. Deux classes ont été alors créées pour modéliser ces deux éléments et les intégrer au modèle de produit en cours de construction avec deux liens *Est-un* les reliant à la classe *routing word* (Figure 129). La *Stratégie de spécialisation* a été par la suite appliquée pour spécialiser la classe *IntraSF* en deux sous-classes (*PNTN* et *PNTR*) et pour spécialiser la classe InterSF en cinq sous-classes (*PNTD*, *PNTC*, *PNTA*, *PNTM* et *PNTE*).

La *Stratégie d'association* est la dernière appliquée pour la formalisation du modèle de produit, elle permet de modéliser le lien entre *Domain Word*, *Logical Unit* et *Pallet* en utilisant une relation ternaire.

La Figure 129 présente le modèle de produit de la méthode Lyee obtenu par rétro-ingénierie. Ce modèle formalise l'ensemble des besoins logiciels nécessaire à *LyeeALL* pour générer les applications, nous l'appelons alors : *Modèle des Besoins Interne Lyee (MBIL).*

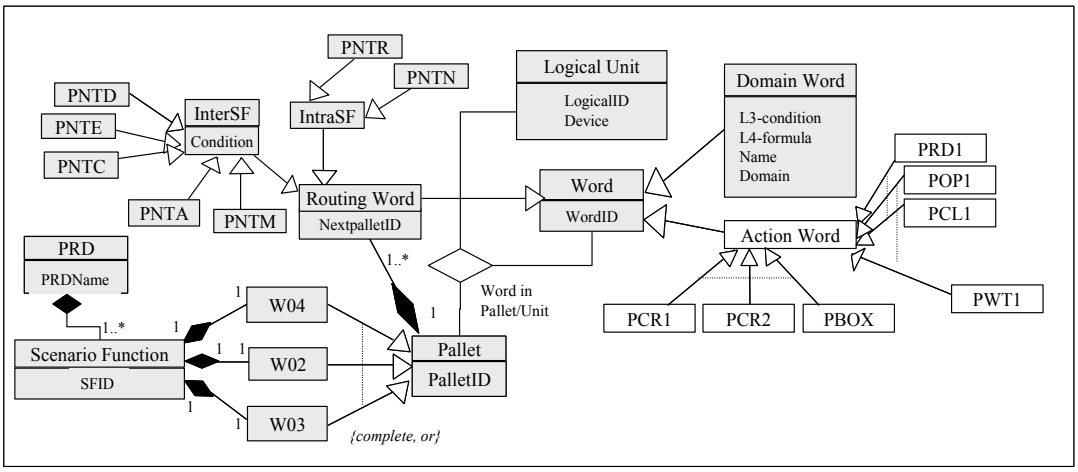


Figure 129. Modèle des Besoins Internes Lyee (MBIL)

- *Atteindre l'intention Formaliser le modèle de processus*

Pour atteindre l'intention *Formaliser le modèle de processus*, la stratégie *Par méta-modélisation de processus* suggère de construire à partir des éléments de processus identifiés par la réalisation de l'intention *Identifier les éléments de la méthode* (Tableau 19) un modèle de processus utilisant le paradigme orienté-activité.

L'exécution du plan décrivant la DRI_{11} (Figure 42) associée à cette stratégie suggère de définir en premier lieu le type d'ordonnancement du modèle de processus. Dans le cas de Lyee, nous optons pour un modèle de processus orienté-activité utilisant un ordonnancement séquentiel des activités. En deuxième lieu, cette directive suggère de modéliser en utilisant la technique de méta-modélisation les éléments de processus identifiés. Le résultat d'application de la DRI_{11} est le modèle de processus présenté à la Figure 130.

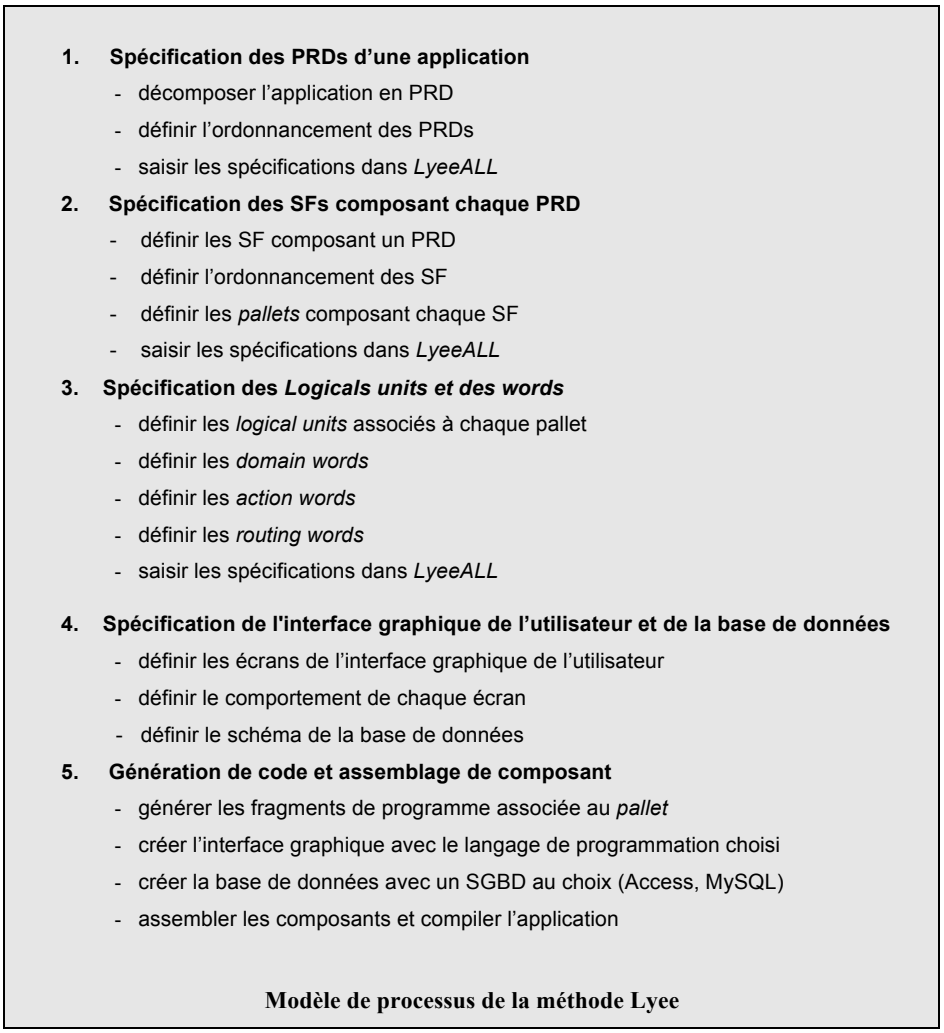


Figure 130. Modèle de processus de la méthode Lyee

1. Spécification des PRDs d'une application

Le premier élément de processus identifié au Tableau 19 est '*Spécifier les PRDs d'une application*', l'application de la DRI_{11} permet de définir cet élément comme première étape du modèle de processus en cours de construction. Afin de mieux guider l'ingénieur d'application dans l'exécution de cette étape, la *Stratégie de décomposition* suggère de la décomposer en activités plus atomiques, trois activités ont été alors identifiées (i) décomposer l'application en PRD, (ii) définir l'ordonnancement des PRD et (iii) saisir les spécifications dans LyeeALL. La Figure 131 présente le Formulaire *LyeeALL* utilisé pour l'enregistrement des PRDs.

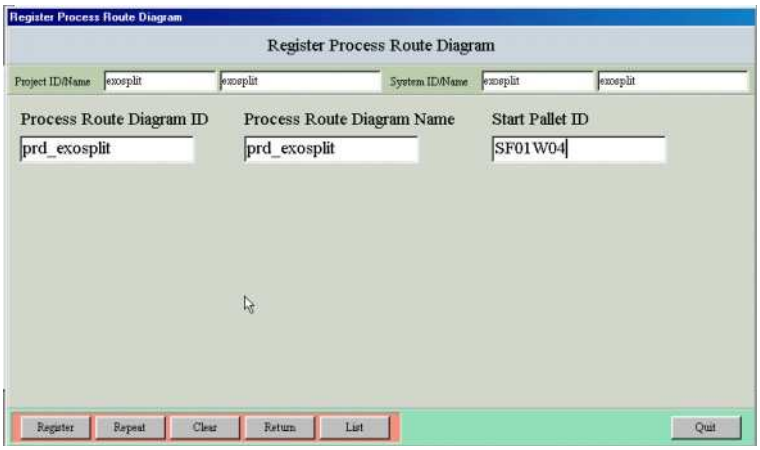


Figure 131. Formulaire LyeeALL utilisé pour enregistrer un PRD

2. Spécification des SF et des pallets composant chaque PRD

Le deuxième élément de processus identifié dans le Tableau 19 est '*Spécifier les SFs composant chaque PRD*'. L'application de la DRI_{11} permet de définir cet élément comme la deuxième étape du modèle de processus en cours de construction. La *Stratégie de décomposition* est utilisée, comme c'est le cas de la stratégie précédente, pour décomposer cette étape en quatre activités (i) définir les SF composant un PRD, (ii) définir l'ordonnacement des SF, (iii) définir les pallets composant chaque SF et (iv) saisir les spécifications dans LyeeALL. La Figure 132 présente le formulaire LyeeALL utilisé pour la saisie des *pallets* composant un PRD.

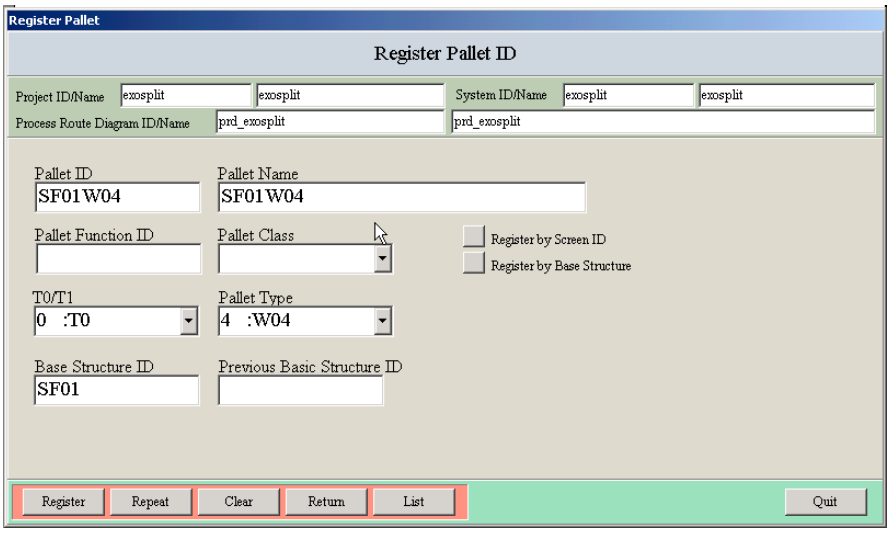


Figure 132. Formulaire LyeeALL utilisé pour enregistrer les Pallets.

3. Spécification des *logicals units* et des *words*

Le troisième élément de processus identifié dans le Tableau 19 est '*Spécifier les logical units associés à chaque pallet*'. L'application de la DRI_{11} permet de définir cet élément comme activité du modèle de processus en cours de construction.

Trois éléments de processus ont été identifiés pour la définition des *words* associés à un *pallet* et un *logical unit* : '*Spécifier les domain words*', '*Spécifier les action words*' et '*Spécifier les routing word*', la DRI_{11} associée à la stratégie Par méta-modélisation de processus est appliquée pour définir ces trois éléments de processus comme trois activités du modèle de processus en cours de construction.

Les quatre activités définis ci-dessus concernent la même partie de produit et peuvent être regrouper dans la même étape. La *Stratégie de composition* est alors utilisée pour agréer ces quatre activités en une étape appelée '*Spécification des logical units et des words*'

La dernière activité dans cette étape est la saisie des *logicals units* et des *words* dans *LyeeALL*. La Figure 133 présente le formulaire LyeeALL utilisé pour la saisie des words.

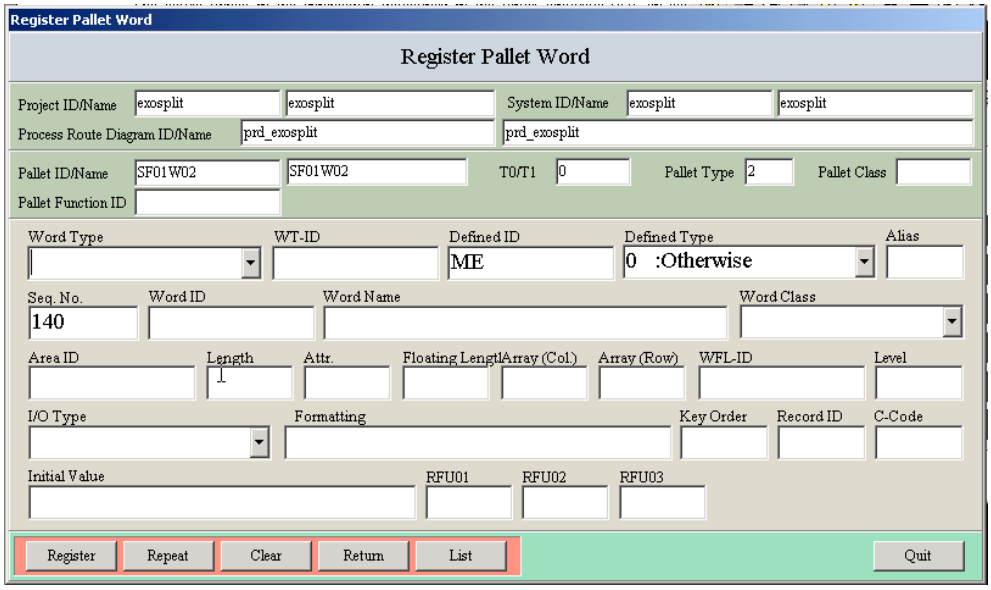


Figure 133. Formulaire LyeeALL utilisé pour la saisie des words

4. Spécification de l'interface graphique de l'utilisateur et de la base de données

L'élément de processus suivant dans la liste des éléments identifié dans le Tableau 19 est 'Définir l'interface graphique de l'utilisateur et la base de données'. L'application de la DRI_{11} permet de définir cet élément comme étape du modèle de processus en cours de construction. La *Stratégie de décomposition* est appliquée par la suite pour décomposer cette étape en trois activités : (i) définir les écrans composant l'interface graphique de l'application, (ii) définir le comportement de chaque écran et (iii) définir le schéma de la base de données.

5. Génération de code et assemblage de composant

Le dernier élément de processus identifié dans le Tableau 19 est '*Générer les programmes par LyeeALL*'. La DRI_{11} est appliquée pour définir cet élément comme la dernière étape du modèle de processus de la méthode Lyee. La *Stratégie de décomposition* est utilisée pour décomposer cette

étape en quatre activités (i) générer les fragments de programme associés au *pallet* dans le langage de programmation souhaité (C, Java, Cobol ou VB) en utilisant l'outil de génération de LyeeALL, (ii) créer l'interface graphique avec le langage de programmation choisi, (iii) créer la base de données avec un SGBD au choix (Access, MySQL) et (iv) assembler les différents composants et compiler l'application. La Figure 134 présente une image écran de l'outil de génération de programmes fourni par *LyeeALL*.

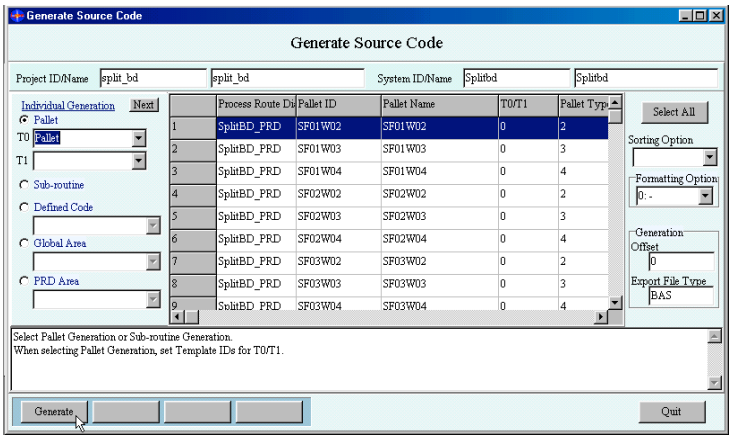


Figure 134. Génération de code avec LyeeALL

5. Evolution de la méthode Lyee

Cette section présente étape par étape le processus d'évolution de la méthode Lyee obtenue par rétro-ingénierie (considéré comme la méthode *As-Is)* vers une nouvelle méthode Lyee centrée sur les utilisateurs (la méthode *To-Be*) en appliquant la démarche d'évolution que nous proposons. La Figure 135 rappelle la carte d'évolution présentée au chapitre 5.

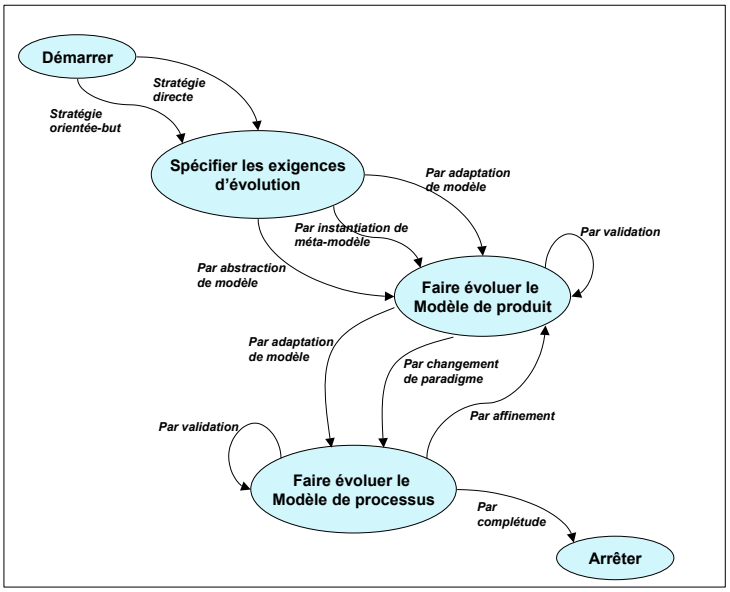


Figure 135. La carte d'évolution

- *Atteindre l'intention Spécifier les exigences d'évolution*

La première intention à atteindre dans la carte d'évolution est : *Spécifier les exigences d'évolution*. Dans notre situation la *Stratégie orientée-but* est sélectionnée, la directive associée à cette stratégie nous suggère de construire une carte des exigences formalisant les buts que doit satisfaire la méthode *To-Be*. Le résultat d'application de cette directive est la carte des exigences présentée à la Figure 136. Cette carte contient deux intentions principales *Capturer les besoins des utilisateurs* et *Spécifier les besoins Internes Lyee* ; ces deux intentions représentent les buts que doit satisfaire la méthode *To-Be*.

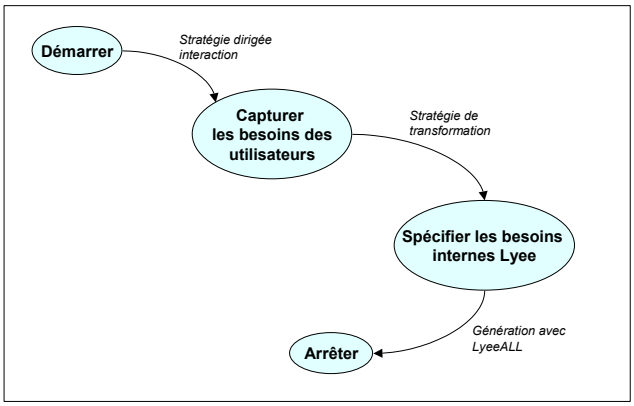


Figure 136. Carte des exigences pour l'évolution de la méthode Lyee

- *Atteindre l'intention Faire évoluer le modèle de produit*

Une fois les exigences d'évolution spécifiées, la deuxième intention à atteindre dans la carte d'évolution est *Faire évoluer le modèle de produit*. Le modèle de produit *As-Is* de la méthode Lyee est le *Modèle des Besoins Interne Lyee (MBIL)* obtenu par rétro-ingénierie (Figure 129). Vu les spécificités de ce modèle la DSS_2 de la carte d'évolution nous suggère de sélectionner la stratégie *Par abstraction de modèle* (Figure 135), l'application de la directive DRI_3 (la Carte C3) associée à cette stratégie nous permet d'abstraire à partir du *MBIL* une nouvelle couche de modèle ayant des éléments d'un niveau d'abstraction plus élevé. La Figure 137 rappelle la Carte C3.

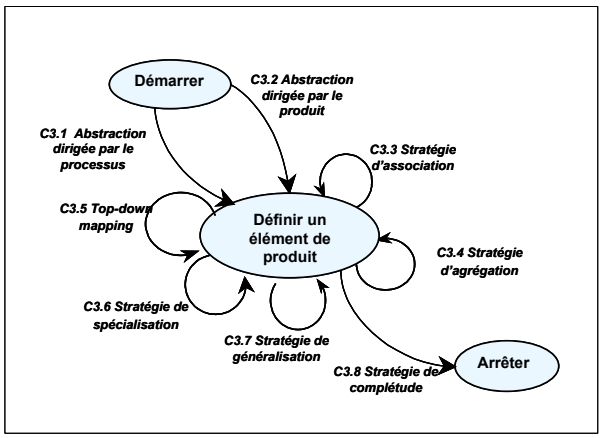


Figure 137. La Carte C3

Le résultat de l'exécution de la carte C3est le modèle de produit *To-Be* présenté à la Figure 138. Ce modèle est composé de deux niveaux, le bas niveau représente le *Modèle de Besoins Internes Lyee (MBIL),* le haut niveau représente le *Modèle de Besoins Lyee Centrés Utilisateur (MBLCU)*.

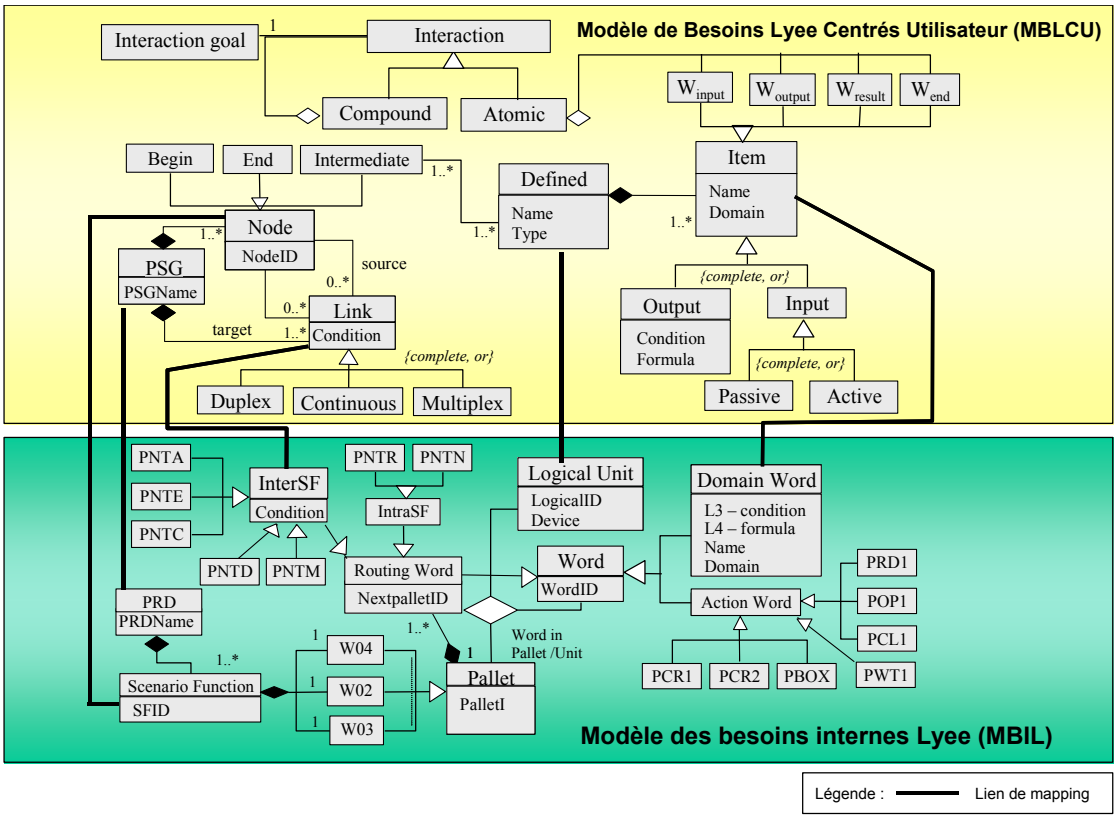


Figure 138. Modèle de produit *To-Be* de la méthode Lyee

Le modèle de produit To-Be présenté ci-dessus est construit comme suit :

En utilisant la stratégie *Abstraction dirigée par le produit* de la carte C3, le concept *Item* est obtenu par abstraction à partir du concept *Domain Word* dans le *MBIL*. La *Stratégie de spécialisation* est utilisée pour spécialiser le concept *Item* en *Input* et *Output*. Les *Outputs* représentent les résultats produits par le système par contre les *Inputs* représentent les données saisies par l'utilisateur. De la même manière le concept *Input* est spécialisé en deux concepts *Passive* et *Active*. Le premier correspond aux items déclenchant les actions du système (par exemple les boutons dans une interface graphique), le second correspond aux valeurs saisies par l'utilisateur (par exemple une zone de texte dans une interface graphique).

Ensuite, nous analysons le modèle de processus *As-Is*. Celui-ci a comme objectif la génération automatique de programmes à partir de spécifications déclaratives des besoins des utilisateurs. L'exécution des programmes générés doit satisfaire les besoins des utilisateurs, c'est-à-dire un ou plusieurs de ces buts. Pour cette raison nous devons inclure dans la nouvelle couche de modèle de

produit en cours de construction des concepts permettant de définir les buts de l'utilisateur et exprimer comment l'utilisateur interagit avec le système pour les atteindre.

La Stratégie *Abstraction dirigée par le processus* de la carte C3 nous permet de définir la notion d'interaction dans le *MBLCU*. Une interaction représente un échange entre l'utilisateur et le système. Cette interaction est orientée-but dans la mesure où l'utilisateur demande au système d'atteindre le but qu'il a en tête sans connaître comment le système va le faire. Par conséquent, le concept de but (*Interaction Goal*) est défini dans le modèle et nous l'associons à *Interaction*.

La complexité du but de l'interaction implique la complexité de l'interaction correspondante : si le but de l'interaction peut être décomposé en plusieurs buts atomiques, l'interaction correspondante peut aussi être décomposée. Pour cette raison, l'interaction est spécialisée en composée (*Compound)* et atomique (*Atomic)* en utilisant la *Stratégie de spécialisation*.

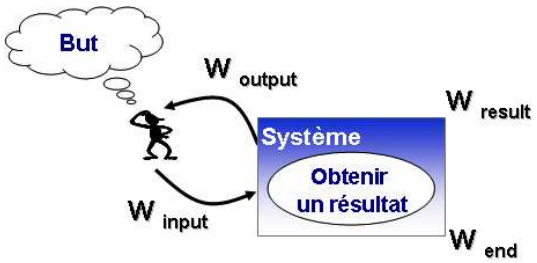


Figure 139. Vue générale d'une interaction

Une interaction atomique implique la manipulation de données en entrée et en sortie. En effet, comme le montre la Figure 139, l'utilisateur fournit un ensemble de données en input et reçoit en sortie les résultats satisfaisant son but. La *Stratégie de décomposition* nous permet de décomposer chaque interaction en quatre types d'*items* : W_{input} , W_{output} , W_{result} et W_{end}.

- W_{input} : représente les entrées fournies par l'utilisateur,
- W_{result} : représente les résultats de la réalisation du but,
- W_{output} : représente les sorties affichées à l'utilisateur
- W_{end} : représente la fin de l'interaction.

La stratégie *Abstraction dirigée par le produit* de la carte C3 est appliquée par la suite, pour abstraire le concept *Defined* à partir de celui de *Logical Unit*. Une classe modélisant l'élément *Defined* est alors créée et intégrée à la nouvelle couche du modèle de produit en cours de construction (*MBLCU)*. D'une manière similaire, le concept *PSG (Precedence Succedence Graph)* dont le *MBLCU* est obtenu par abstraction à partir du concept *PRD* dans le *MBIL*.

La stratégie *Abstraction dirigée par le produit* est appliquée ensuite pour abstraire l'élément *Node* à partir de l'élément *Scenario Function*. Une classe modélisant l'élément *Node* est ainsi créée et

intégrée au *MBLCU* en moyennant un lien d'agrégation la reliant à *PSG*. De manière semblable, l'élément *Link* est défini par abstraction de l'élément *InterSF*.

Grâce à la *Stratégie de spécialisation*, l'élément *Link* est spécialisé en trois types *Duplex*, *Continuous* et *Multiplex* correspondant aux trois types de branchement entre les *Nodes.* L'élément Node, quant à lui, est spécialisé en utilisant la même stratégie en trois types *Begin*, *End* et *Intermediate*. Trois classes correspondant à ces trois éléments sont alors définies et intégrées au *MBLCU*.

- *Atteindre l'intention Faire évoluer le modèle de processus*

Cette section est consacrée à l'illustration du processus d'évolution du modèle de processus As-Is de la méthode Lyee. En se basant sur la carte des exigences (Figure 136), le modèle de processus *To-Be* doit satisfaire les objectifs suivants:

1. systématiser la capture des besoins des utilisateurs et leur formulation conformément au *MBLCU,*

2. systématiser la génération des besoins internes Lyee à partir des besoins Lyee centrés utilisateur.

Le modèle de processus *As-Is* défini par rétro-ingénierie est formalisé en utilisant le paradigme orienté activité. Pour pouvoir satisfaire les exigences d'évolution définies ci-dessus un changement de paradigme de modélisation de processus s'avère nécessaire. La stratégie *Par changement de paradigme* de la carte d'évolution est alors sélectionnée pour atteindre l'intention *Faire évoluer le modèle de processus*.

Sur la base de notre situation d'ingénierie, la DSS_3 de la carte d'évolution nous suggère de sélectionner la stratégie *Orientée-Patron* du cluster <*Faire évoluer le modèle de produit, Faire évoluer le modèle de processus, Par changement de paradigme*>. Le modèle de processus *To-Be* est alors construit sous la forme d'un Catalogue de patrons permettant d'atteindre les objectifs énoncés.

La directive DRI_{11} (la carte C11) associée à la stratégie *Orientée-Patron* nous suggère de commencer le processus d'évolution par l'identification de patrons. Pour cela, nous appliquons la stratégie *Dirigée par les buts* de la carte C11 en se basant sur les deux buts principaux : *« Formuler les besoins des utilisateurs »* et *« Générer les besoins internes Lyee ».*

- *Formuler les besoins des utilisateurs*

Comme énoncé dans la section précédente, la démarche d'expression des besoins des utilisateurs dans la méthode Lyee *To-Be* est centrée sur le concept d'interaction. Ainsi, l'identification de patrons correspondant au but *Formuler les besoins des utilisateurs* est fondée sur ce concept et consiste à identifier les activités génériques permettant l'expression des besoins dans ce contexte. Quatre activités ont été identifiées suivant ce raisonnement :

- « Commencer une interaction » qui peut être associée au but : *Formuler les besoins 'To Start',*
- « Réaliser les traitements » qui peut être associée au but : *Formuler les besoins 'To Act',*
- « Préparer les sorties » qui peut être associée au but : *Formuler les besoins 'To Output',*
- « Terminer l'interaction » qui peut être associée au but : *Formuler les besoins 'To End'.*

Chacune de ces activités est reliée à la typologie des *Words* présentée dans le paragraphe précédent. Elles concernent la collecte et la spécification des *words*, leur groupement dans des *defined* et leur positionnement dans le *psg* correspondant à l'interaction.

- *Commencer une interaction* est en relation avec la capture des W_{input}
- *Réaliser les traitements* est en relation avec le calcul des W_{result}
- *Préparer les sorties* est en relation avec la définition des W_{output}
- *Terminer l'interaction* est en relation avec la détermination des W_{end}

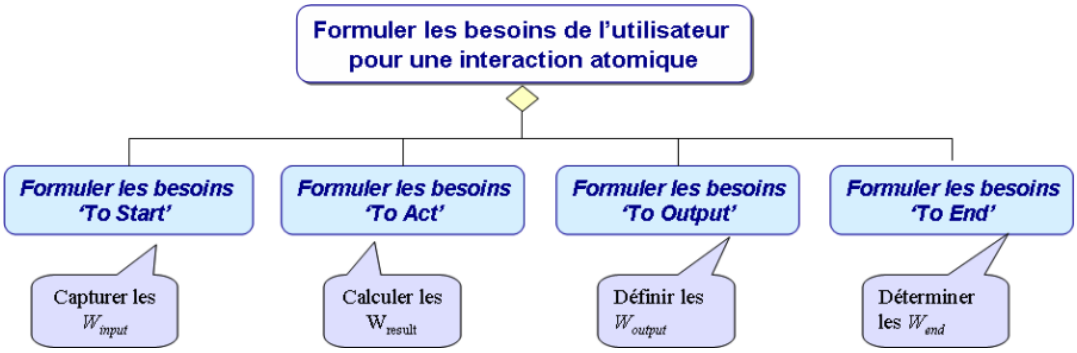


Figure 140. Activités génériques pour la capture des besoins d'une interaction atomique

Ensuite, la stratégie *Basée sur la situation* de la carte C11 est sélectionnée pour identifier les patrons correspondant aux quatre buts découverts (Figure 140). Selon cette stratégie, les différentes situations associées à chacun de ces buts sont identifiées. Par exemple, on peut distinguer deux situations différentes lors de la capture des W_{input} ; la première concerne le cas où les entrées sont directement saisies par l'utilisateur, la deuxième correspond au cas où ces valeurs sont obtenues via un accès à un fichier ou à une base de donnée. On identifie alors deux patrons ayant le même but *Formuler les besoins 'To Start'* mais ayant deux situations différentes *les Inputs capturés à partir de l'utilisateur* et *les inputs capturés à partir d'un support interne*. Ces deux patrons sont appelés *P2 Immediate Start* et *P3 Prequisite for Start.* D'une manière similaire on identifie deux situations génériques différentes pour chaque but et on associe un patron à chaque couple <*but, situation*>. Huit patrons génériques sont identifiés suivant le même principe ; ils sont récapitulés dans le Tableau 20.

But	Situation	Nom du Patron
Formuler les besoins '*To Start*'	Les W_{input} sont saisis directement par l'utilisateur	*P2 Immediate Start*

Formuler les besoins '*To Start*'	Les W_{input} sont extraits à partir d'une base de données ou un fichier	*P3 Prequisite for Start*
Formuler les besoins '*To Act*'	Les W_{result} sont calculés par une formule simple qui ne nécessite pas le calcul de *words* intermédiaires	*P1 Simple Word*
Formuler les besoins '*To Act*'	Les W_{result} sont calculés par une formule complexe qui nécessite le calcul de *words* intermédiaires et éventuellement des accès à la base de données	*P8 Complexe Word*
Formuler les besoins '*To Output*'	Pas d'obstacles dans la capture de W_{input} et dans la production des W_{result}	*P6 Single Output*
Formuler les besoins '*To Output*'	Plusieurs cas pour la production de sorties sont considérés à cause d'éventuels obstacles dans la capture de W_{input} ou la production des W_{result}	*P7 Multiple Output*
Formuler les besoins '*To End'*	L'interaction se termine normalement sans opération interne supplémentaire	*P4 Simple End*
Formuler les besoins '*To End'*	Une ou plusieurs opérations internes sont effectuées à la fin de l'interaction comme la sauvegarde de la totalité ou d'une partie des W_{output}	*P5 Compound End*
Formuler les besoins d'une interaction atomique	Le but de l'interaction est atomique	*P9 Simple Composition*
Formuler les besoins d'une interaction composée	Le but de l'interaction est composé	*P10 Complex Composition*

Tableau 20. Patrons associés au but Formuler les besoins des utilisateurs

Chacun des huit patrons (de P1 au P8) correspond à une activité de besoin atomique. La capture de l'ensemble des besoins correspondant à une interaction atomique requiert l'exécution de chacune des activités '*To Start*', '*To Act*', '*To Output*' et '*To End*'. La *Stratégie d'agrégation* est appliquée pour identifier le patron *P9 Simple composition* permettant de composer la solution correspondante à une interaction atomique comme une suite de quatre patrons atomiques.

La *Stratégie de succession* de la carte C11 nous suggère de réfléchir sur l'identification de patrons correspondant à une interaction composée. Ainsi, le patron *P10 Complex Composition* est identifié. Il est dédié à la formulation des besoins associés à une interaction composée. Il appelle le patron *P9* pour chaque interaction atomique la composant.

La deuxième intention à atteindre dans la carte C11 est la construction des patrons identifiés. Pour cela nous sélectionnons la *Stratégie dirigée par le produit*, celle-ci nous guide dans la définition de solution associée à chaque patron. Cette solution est définie par un ensemble d'activités permettant de satisfaire le but du patron. Dans ce qui suit, on présente les 10 patrons obtenus en appliquant cette stratégie.

- *Présentation du patron P2*

Le but du *patron P2* consiste à formuler les besoins '*To Start*' dans le cas où les W_{input} sont saisis directement par l'utilisateur. La solution de ce patron guide l'ingénieur Lyee pour instancier les éléments (*Defined*, *Item* et *PSG*) qui sont utilisés pour capturer les valeurs en entrées. La Figure 141 présente le patron *P2* comme suit : la première section introduit le nom du patron, la seconde section présente le problème (le but et la situation), la troisième section présente la solution proposée.

Patron P2: Immediate Start
Problème: <But : Formuler les besoins 'To Start'> <Situation : W_{input} <= $Capture_{user}$( )>
Solution : 1. Créer un *defined* S_{input} de type écran, déterminer son nom (*name*) 2. Élucider les *items* associés au W_{input} 3. Lier les items au *defined*, déterminer pour chaque *item* son nom et domaine (*domain*) 4. Typer les items comme *input* et *passive* 5. Créer un *psg* avec le *defined* comme *intermediate node* et le lier à partir du nœud *start* avec un *continuous link*.

Figure 141. Patron P2 Immediate Start

- *Présentation du patron P3*

Le but du *patron P3* est de formuler les besoins '*To Start*' dans le cas où les W_{input} sont extraits à partir d'une base de données ou un fichier, la Figure 142 présente ce patron.

Patron P3 : Prequisite for Start
Problème: <But : Formuler les besoins 'To Start'> <Situation : W_{input} <= f (W_{bd})>
Solution : 1. Appliquer la définition suivante des W_{input} pour élucider les items $W_{input} \leftarrow W_{Cmddb}$ ($W_{inputdb}$, W_{db}) W_{db} : l'ensemble des items qui vont être extrait de la base de donnée W_{Cmddb} : l'item permettant de déclencher l'extraction des items de la base de donnée $W_{inputdb}$: l'ensemble des inputs item à capturer de l'utilisateur pour formuler la requête base de donnée W_{input} : l'ensemble des items de l'écran initialiser à partir des W_{db}

2. Attacher un *name* et *domain* à chaque item

3. Créer les *defineds* suivant :

Un *defined* S_{input} de type écran

Un *defined* Db_{input} de type base de donnée

4. Attacher les items aux *defineds* :

Attacher les W_{db} au *defined* Db_{input}

Attacher W_{Cmddb} , W_{input} et $W_{inputdb}$ au *defined* S_{input}

5. Typer les items comme suit :

Typer les W_{Cmddb} et $W_{inputdb}$ comme input

Typer W_{Cmddb} comme *active* et $W_{inputdb}$ comme passive

Typer les W_{input} comme *output* et *passive input*

Typer les W_{db} comme *passive input* s'ils sont extrait à partir de la base de donnée

Typer les W_{db} comme *output* s'ils sont utiliser pour interroger la base de donnée

6. Definir la *formula*

- Pour chaque W_{db} de type *output*

Les output item des W_{db} prennent leurs valeurs à partir des $W_{inputdb}$ du S_{input}

$db_i \in W_{db}$ => formula : $db_i = S_{input}.inputdb_j$

- Pour chaque W_{db} de type *input*

Les input items des W_{db} sont extrait à partir du Db_{input}

$db_i \in W_{db}$ => formula : db_i = "database retrieve command"

- Pour chaque W_{input}

Les items W_{input} prennent leurs valeurs à partir des items extrait à partir du Db_{input}

$Input_i \in W_{input}$ => formula : $input_i = Db_{input}.db_j$ and condition : $Db_{input}.db_j$ found

7. Creer un PSG avec :

Deux Intermediate nodes S_{input} et Db_{input}

Un continuous link de S_{input} vers Db_{input}

Un continuous link de begin vers S_{input}

Un duplex link de Db_{input} vers S_{input}

Figure 142. Patron P3 Prequisite for Start

- *Présentation du patron P1*

Le but du *patron P1* est de formuler les besoins '*To Act*' dans la situation où les W_{result} sont calculés par une formule simple qui ne nécessite pas le calcul de *words* intermédiaires, la Figure 143 présente ce patron.

Patron P1: Simple Word
Problème: <But : Formuler les besoins 'To Act'> <Situation : W_{result} <= f(W_{input})>
Solution : 1. Élucider le *item* correspondant au W_{cmd}, définir son *name* et le typer comme *active input* 2. Attacher cet item au *defined* S_{input}

3. Élucider les *items* correspondant au W_{result} , définir leur attributs et les typer comme *output*
4. Créer un *defined* S_{result} de type écran (sous-defined du *defined* S_{input}), définir son *name* et lui attacher les items W_{result}
5. Définir la formula pour chaque *output item*

Figure 143. Patron P1 Simple Word

- *Présentation du patron P8 :*

La Figure 144 présente le *patron P8*, le but de ce patron est de formuler les besoins '*To Act*' dans la situation où les W_{result} sont calculés par une formule complexe qui nécessite le calcul de *words* intermédiaires et éventuellement des accès à la base de données.

Patron P8: Complex Word

Problème:

<But : Formuler les besoins 'To Act'>

<Situation : $W_{result} \leftarrow f \circ g_{Wintermediate}^{N} (W_{input})$ >

$W_{intermediate} \leftarrow g (W_{input})$

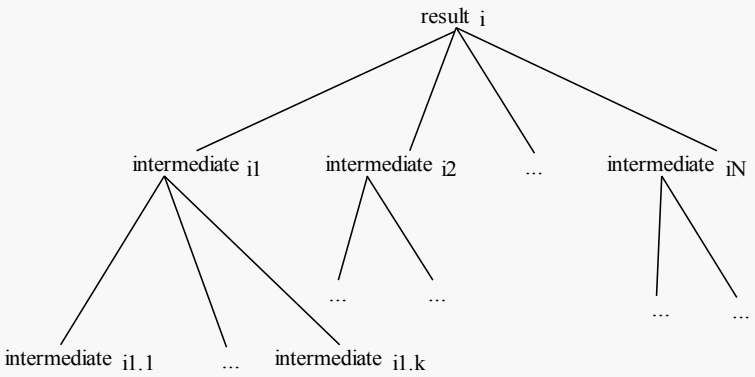
….

$W_{intermediate} \leftarrow g^{N-1} (W_{intermediate})$

$W_{result} \leftarrow f (W_{intermediate})$

Solution :

1. Élucider les *intermediate items,* pour chaque résultat ($result_i$) complexe des W_{result} construire une arbre de décomposition.



Arbre de décomposition

2. Classifier les intermediates items comme suit :

Dépendant d'écran (Scintermediate)

Dépendant de base de données (dbintermediate)

3. Créer les *defineds* suivant :

- Un *defined* S_{result} de type écran (sous-defined du *defined* S_{input}), définir ces attributs et lui attacher tous les *results* et *intermediate items*

- Un defined $Db_{intermediate}$ de type base de donnée, définir ces attributs et les lui associer les dbintermediate *items*. Typer les dbintermediate comme *output item* s'il sont utilisés pour accéder à la base de données et comme *passive input* si leurs valeurs sont extraites de la base de données.

4. Élucider l'item correspondant au W_{cmd}, définir son *name*, le typer comme *active* et l'attacher au *defined* S_{input}

(defined des W_{input}).

5. Introduire le $Db_{intermediate}$ comme *intermediate node* dans le *psg* avec :
un forward *continuous link* partant de S_{input}
un backward *duplex link* vers S_{input}

6. Définir *formula* :
- pour chaque result *item* :
$result_i = f_i (S_{input}.intermediatej)$
ou $result_i = Db_{intermediate}.input_j$
- pour chaque intermediate *item* :
$Intermediate_j = g_j (S_{input}.input_k)$
ou $Intermediate_j = Db_{intermediate}.input_k$
- pour chaque input dbIntermediate
$dbIntermediate_i$ = Select(outputdbIntermediate)
- pour chaque output dbIntermediate :
$dbIntermediate_i = S_{input}.Intermediate_i$

7. Definir *condition* :
pour chaque *item* dbdependee : $Db_{intermediate}$.intermediate found
pour chaque validity *item* : condition = $S_{input}.intermediate_{dependee}$ = true

8. La condition de validité de l'item est associée à l'input correspondant dans le S_{input}

Figure 144. Patron P8 Complex Word

- *Présentation du patron P6 :*

Le but à atteindre par l'application du *patron P6*, présenté à la Figure 145, est la formulation des besoins '*To output*' dans la situation où il n'y a pas d'obstacles dans la capture de W_{input} et dans la production des W_{result}.

Patron P6 : Single Output
Problème: <But : Formuler les besoins 'To Output' > <Situation : ∃ un seul cas: $W_{case}^1 = W_{output}$ >
Solution : 1. Dupliquer le defined associé au W_{result} en defined S_{case}^1 2. Introduire un *continuous link* sans condition au defined S_{input} 3. Définir une *formula* pour chaque *output*, la structure de la *formula* est la suivante : $output_i = S_{input}.output_i$

Figure 145. Patron P6 Single Output

- *Présentation du patron P7 :*

La Figure 146 présente le *patron P7*, le but de ce patron est de formuler les besoins '*To Output*' dans la situation ou plusieurs cas pour la production de sorties sont considérés à cause d'éventuels obstacles dans la capture de W_{input} ou la production des W_{result}.

Patron P7 : Multiple Output
Problème: < But : Formuler les besoins 'To Output' > < Situation : $\exists\ W_{case}^{i} = P(W_{output}) : f_{boolean}(W_{output})=true$ > $W_{output}= \cup_i W_{case}^{i}$
Solution : 1. Élucider $\{W_{case}^{i}\}$ correspondant aux obstacles dans : - La capture des W_{input} - Le calcul des $W_{intermediate}$ et les W_{result} 1. Décomposer le *defined* S_{result} en autant de *defined* S_{case}^{i} de type écran que de valeur de i 2. Attacher les items avec leurs attributs et leur types au defined S_{case}^{i} correspondant. 3. Définir une *formula* pour tous les *items* de chaque S_{case}^{i}. La structure de la *formula* est la suivante : $output_i \in W_{case}^{i} \Rightarrow output_i = S_{input}.output_i$ 4. Introduire les defined S_{case}^{i} découvet dans le *psg* avec un *continuous link* avec le defined S_{input} - transformer chaque $f_{boolean}(W_{output})$ en une condition Ci associée au *link* ayant pour cible le defined S_{case}^{i} - pour les obstacles de base de données typique formuler les conditions comme suit : W_{input} obstacle : Ci = Db_{input}.dbj not found W_{result} obstacle : Ci = $Db_{intermediate}$.intermediate not found

Figure 146. Patron P7 Multiple Output

- *Présentation du patron P4*

La Figure 147 présente le *patron P4*, Le but de ce patron est de formuler les besoins '*To End*' dans la situation où l'interaction se termine normalement sans opération interne supplémentaire.

Patron P4: Simple End
Problème: <But : Formuler les besoins 'To End'> < Situation : Pas d'opérations internes supplémentaires pour terminer l'interaction >
Solution : 1. Ajouter les boutons Stop et Return pour chaque defined S_{case}^{i} de type écran 2. Ajouter le bouton Cancel pour le defined S_{input} 3. Ajouter le nœud end dans le psg

4. Completer le psg avec

- un *continuous link* partant du nœud associé à S_{input} vers le nœud end avec "S_{input}.CmdCancel = true" comme condition

- un *continuous link* partant de chaque nœud associé à un S_{case}^{i} vers le nœud end avec "S_{case}^{i}.CmdStop = true" comme condition

- un *multiplex link* partant de chaque nœud associé à un S_{case}^{i} vers le nœud associé au S_{input} avec "S_{case}^{i}.CmdReturn = true" comme condition

Figure 147. Patron P4 Simple End

- *Présentation du patron P5 :*

Le but du *patron P5* est de formuler les besoins '*To End*' dans la situation où une ou plusieurs opérations internes sont effectuées à la fin de l'interaction comme la sauvegarde de la totalité ou d'une partie des W_{output}.

Patron P5 : Compound End	
Problème: <But : Formuler les besoins 'To End'> < Situation : W_{indb} ← Store (W_{output})>	
Solution :	
1.	Appliquer la définitions suivante du W_{indb} pour élucider les items W_{indb} ← W_{Cmddb} (W_{dbkey}, W_{db}) W_{Cmddb}: item pour commander le stockage des W_{output} W_{db} : les items de la base de donnée correspondant au W_{output} à stocker dans la base W_{dbkey} : le ou les items identifiant le tuple de données à stocker dans la base
2.	Définir le *domain* et *name* de chacun de ces *items*
3.	Créer un defined Db_{output} de type base de données et lui attacher les W_{db} et W_{dbkey}
4.	Attacher les W_{cmddb} au defined S_{case}^{i} correspondant
5.	typer les items :
- typer W_{Cmddb} comme active input	
- typer W_{dbkey} comme output	
- typer W_{indb} comme output	
6.	définir formula pour tout W_{indb}

Figure 148. Patron P5 Compound End

- *Présentation du patron P9 :*

La Figure 149 présente le *patron P9*, le but de ce patron est d'assister l'ingénieur Lyee dans la formulation des besoins d'une interaction atomique. Un patron atomique est alors sélectionné pour chaque activité de besoin atomique (*Formuler les besoins 'To Start'*, *Formuler les besoins 'To Act'*, *Formuler les besoins 'To Output'* et *Formuler les besoins 'To End'*).

Patron P9 : Simple Composition

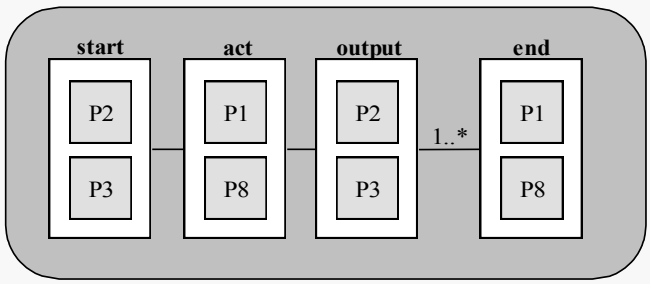
Problème:

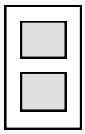
< But : Formuler les besoins d'une interaction atomique >

< Situation : Le but de l'interaction est atomique >

Solution :

1. Appliquer les patrons dans l'ordre qui suit de gauche à droite





Choix multiple ___ séquence 1..* multiplicité

2. Dans le cas de cas de choix multiple sélectionner le patron adéquat en se basant sur la situation d'ingénierie.

Figure 149. Patron P9 Simple Composition

- *Présentation du Patron P10 :*

La Figure 150 présente la patron P10, le but de ce patron est de formuler les besoins d'une interaction composé. Il est appliqué dans la situation ou le but de l'interaction est composé. Ce patron propose de décomposer une interaction composée en plusieurs interactions atomique. Il propose par la suit d'associer un patron atomique (P1 → P8) à chaque interaction atomique identifiée.

Patron P10 : Complex Composition
Problème: < But : Formuler les besoins d'une interaction composée> < Situation : Le but de l'interaction est composé>
Solution : 1. Elucider un ensemble ordonné d'interaction atomique en appliquant : alternative 1 : le mécanisme de décomposition des buts alternative 2 : la décomposition de l'interaction en se basant sur les Wcdmi 2. Appliquer le patron P9 pour formuler les besoins de chaque interaction atomique élucidé 3. Appliquer ∀ (Ii, Ii+1) la règle suivante pour établir les liens entre psgi and psgi+1 Fusionner end node du psgi avec le start node psgi+1 (Chaque link du psgi partant vers le end node et permettant d'atteindre le but d'une interaction atomique doit être attaché avec le premier intermediate node du psgi+1)

Figure 150. Patron P10 Complex Composition

- *Générer les besoins internes Lyee*

La réalisation du but *Générer les besoins internes Lyee* permet la transformation de l'ensemble des besoins des utilisateurs capturés en appliquant les 10 patrons présentés ci-dessus en un ensemble de besoins internes Lyee (Génération vers LyeeALL). En d'autres termes, ceci permet de générer une instance de PRD à partir d'une instance de PSG. Cette génération est réalisée sur la base des liens de mapping établis entre les deux couches du modèle de produit (Figure 138).

La *Stratégie basée sur la situation* pour atteindre l'intention *Identifier un patron* de la carte C11 nous suggère d'analyser les situations possibles associées à ce but. Une seule situation est identifiée : *Les besoins des utilisateurs sont capturés correctement*. De ce fait, un seul patron est identifié. Ce patron est appelé : *P11 Generate LSR*.

Pour construire le patron identifié, nous sélectionnons la *Stratégie dirigée par le produit* de la carte C11. Cette stratégie nous suggère d'identifier les activités permettant de générer une instance du *MBIL* à partir d'une instance du *MBLCU*. Deux stratégies alt ernatives pour effectuer cette génération ont été découvertes : *One to One Mapping* et *N to M Mapping*. La stratégie *One to One mapping* correspond au cas simple où chaque instance de *node* est transformée en un *scenario function.* La stratégie N to M mapping correspond au cas plus complexe où N *nodes* sont transformés en M *scenario function* . Le nombre de possibilité de mapping avec cette stratégie est multiple et dépend de la situation d'ingénierie de la méthode. La première stratégie est entièrement automatisable par contre la seconde stratégie nécessite l'intervention de l'ingénieur Lyee pour statuer sur les différentes décisions concernant la génération. La Figure 151 illustre ces deux cas de génération.

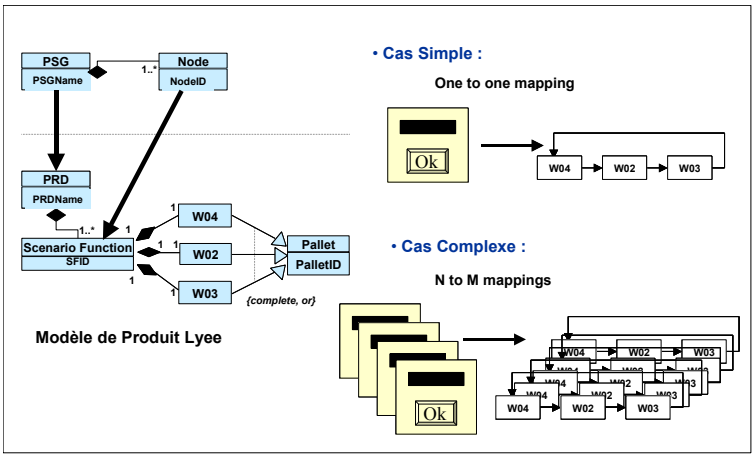


Figure 151. Les cas de génération

La solution de ce patron propose alors le choix entre ces deux stratégies. Chaque stratégie est composée d'un ensemble de règles de génération (R0, $R1_{S11}$, $R2_{S11}$, etc.), le *patron P11* est présenté à la Figure 152 ci-dessous.

Patron P11 : Generate LSR
Problème : < But : Générer les besoins Internes Lyee à partir des besoins Lyee centrés utilisateur < Situation : Les besoins des utilisateurs sont capturés correctement>
Solution : Appliquer une des deux stratégies de génération ➢ **Stratégie *One to One Mapping*** 1. Appliquer R0 : Transformer *PSG* 2. Appliquer $R1_{S11}$: Transformer *intermediate nodes* 3. Appliquer $R2_{S11}$: Transformer *defineds* et *items* 4. Appliquer $R3_{S11}$: Transformer *links* 5. Appliquer R4 : Completer le mapping pour les éléments internes Lyee additionnels ➢ **Stratégie *N to M Mapping*** 1. Appliquer R0 : Transformer *PSG* 2. Appliquer $R1_{T\_FC}$: Transformer *intermediate nodes* avec la tactique *Father-Child merge* ou Appliquer $R1_{T\_B}$: Transformer *intermediate nodes* avec la tactique *Brotherhood merge* 3. Appliquer $R2_{T\_FC}$: Transformer *defineds* et *items* avec la tactique *Father-Child merge* ou Appliquer $R2_{T\_B}$: Transformer *defineds* et *items* avec la tactique *Brotherhood merge* 4. Appliquer $R3_{T\_FC}$: Transformer *links* avec la tactique *Father-Child merge* ou Appliquer $R3_{T\_B}$: Transformer *links* avec la tactique *Brotherhood merge* 5. Appliquer R4 : Completer le mapping pour les éléments internes Lyee additionnels

Figure 152. Patron P11 Generate LSR

Les règles de génération décrivant chaque stratégie de génération présentée ci-dessus sont détaillées dans [Ben Ayed 03b].

Pour arrêter le processus d'évolution de la méthode Lyee, la carte d'évolution nous suggère d'appliquer la stratégie de validation pour s'assurer de la cohérence des modèles de la méthode *To-Be* et du respect des exigences d'évolutions exprimées au début du projet d'ingénierie. La validation de la nouvelle méthode Lyee centrée sur les utilisateurs a été réalisée à travers son application dans plusieurs cas d'étude [Ben Ayed 03a], [Rolland 02d].

6. Conclusion

Dans ce chapitre nous avons présenté le cas d'application de la méthode *MIME* proposé dans ce travail pour l'évolution de la méthode de développement *Lyee*. Cette évolution s'est déroulée en deux étapes : une étape de rétro-ingénierie et une étape d'évolution. Le résultat de cette évolution est une nouvelle méthode Lyee permettant la capture des besoins des utilisateurs et la génération automatique de code à partir de ces besoins.

CHAPITRE 7 : CONCLUSION

Cette thèse s'insère dans le domaine de l'ingénierie de méthodes, elle s'intéresse particulièrement à la problématique de l'évolution des méthodes de développement de SI. Pour résoudre cette problématique nous avons proposé au cours de ce mémoire une *Méthode pour l'Ingénierie de Méthode par Evolution (MIME)* dont l'objectif est d'offrir un guidage à l'ingénieur de méthodes dans les différentes étapes du projet d'évolution d'une méthode.

1. Bilan et Contribution

La méthode *MIME* proposée dans cette thèse constitue un pas en avant vers la compréhension de la manière avec laquelle les méthodes peuvent être construites et la proposition d'une manière systématique de le faire. La méthode *MIME* se base sur un modèle de produit consistant en un méta-modèle de méthodes et deux modèles de processus (un *modèle de processus de rétro-ingénierie* et un *modèle de processus d'évolution*) s'exécutant de manière séquentielle.

- *le méta-modèle de méthodes* recense l'ensemble des concepts qui représentent les caractéristiques communes à toute méthode d'ingénierie de SI. Toute méthode à faire évoluer par *MIME* doit être une instance de ce méta-modèle, la méthode *MIME* est également une instance de ce méta-modèle.

- *le modèle de processus de rétro-ingénierie de méthodes* offre un guidage à l'ingénieur de méthodes pour la formalisation des modèles de produit et de processus de méthode informelle ou mal définie.

- *le modèle de processus d'évolution de méthodes* assiste l'ingénieur de méthodes dans les différentes étapes du projet d'évolution d'une méthode existante (*la méthode As-Is*) vers une nouvelle méthode (*la méthode To-Be*) satisfaisant les objectifs d'évolution.

Chaque modèle de processus de *MIME* est défini avec une *carte* et un ensemble de *directives*. L'utilisation de ce formalisme de représentation, le *MAP*, offre une grande richesse et une grande puissance d'expression à la méthode *MIME* proposant ainsi des modèles de processus *contextuel*, *intentionnel* et *multi-démarches*. En effet, *MIME* offre la possibilité à l'ingénieur de méthodes de

sélectionner de manière dynamique durant le déroulement du projet d'ingénierie la démarche d'évolution adéquate à sa situation d'ingénierie parmi les différentes démarches proposées. *MIME* permet ainsi de tenir compte des spécificités de chaque méthode et des exigences d'évolution. Les directives associées à la *carte d'évolution* et la *carte de rétro-ingénierie* permettent de guider l'ingénieur de méthodes dans l'exécution de la démarche sélectionnée.

L'avantage de l'approche proposée (*MIME*), par rapport aux autres approches d'évolution de méthodes, est de s'attaquer au problème d'évolution de méthodes sous tous ses angles et dans les différentes situations de projets. La proposition de cette thèse met l'accent sur le guidage et l'identification de différentes manières de procéder à l'évolution d'une méthode. En effet, nous avons défini quatre techniques pour réaliser l'évolution des modèles d'une méthode. *Par adaptation de modèle*, *Par évolution de paradigme, Par abstraction de modèle* ou *Par instanciation de méta-modèle*.

La méthode *MIME* se positionne dans le contexte général d'ingénierie de méthodes par le moyen d'un *modèle de processus générique d'ingénierie de méthode* permettant de capturer les intentions génériques de tout projet d'ingénierie de méthodes et d'assister l'ingénieur de méthodes dans la sélection de l'approche de construction la plus appropriée à sa situation d'ingénierie. *MIME* constitue une alternative parmi d'autres dans ce modèle.

La faisabilité de la méthode *MIME* a été prouvée par son application pour l'évolution de la méthode de développement de logiciel *Lyee*. La méthode Lyee complètement informelle au début du projet a été ainsi évoluée vers une nouvelle méthode Lyee centrée sur les utilisateurs. Le modèle de produit de la nouvelle méthode est défini en deux couches de modèle ayant un niveau d'abstraction différent : le *Modèle des Besoins Internes Lyee (MBIL)* et *Modèle des Besoins Lyee Centré Utilisateur (MBLCU)*. Le modèle de processus est défini, quant à lui, sous la forme d'un catalogue de patron permettant la capture des besoins des utilisateurs et la génération des besoins interne Lyee.

2. Perspectives

Le travail présenté dans cette thèse peut être poursuivi dans plusieurs directions :

1. Amélioration de la méthode *MIME* proposée, cette amélioration peut concerner plusieurs aspects :

- Enrichissement du méta-modèle de méthodes : ce méta-modèle peut être évolué pour tenir compte d'autres formalismes de modélisation de produit ou de processus des méthodes. Une autre évolution possible consiste à étendre ce méta-modèle afin de proposer une représentation modulaire des méthodes
- Evolution des modèles de processus de *MIME* : cette évolution peut concerner plusieurs point :

- Extension de la carte de rétro-ingénierie par l'ajout de nouvelles stratégies permettant de formaliser les modèles d'une méthode en utilisant d'autres paradigmes de modélisation de produit ou de processus que ceux proposés.
- Extension de la carte d'évolution par l'ajout de nouvelles stratégies pour faire évoluer les modèles de produit et de processus d'une méthode.
- Affinement des directives associées à la carte de rétro-ingénierie et la carte d'évolution jusqu'au niveau d'actions atomiques exécutables ce qui constitue une étape vers l'automatisation de la méthode *MIME*
- La proposition d'un méta-modèle modulaire nous suggère de réfléchir à faire évoluer la méthode *MIME* pour permettre l'évolution de fragments/composants de méthodes stockés dans une base de méthodes. Les fragments obtenus peuvent être ensuite utilisés par les approches de construction de méthodes par assemblage.

2. Une poursuite de ce travail sur le plan pratique peut consister à développer un outil CAME permettant de systématiser la méthode *MIME* proposée.

BIBLIOGRAPHIE

[ACM 96] *Software Patterns,* Communications of the ACM, Volume 39, Numéro 10, Octobre 1996.

[Agerfalk 03] P.J. Agerfalk, K. Wistrand, F. Karlson, G. Börjesson, M. Elmberg, K. Möler, *Flexible Process and Method Configuration : Outline of a Joint Industry-Academia Research Project*, Proceeding of the 5th International Conference on Enterprise Information Systems (ICEIS 03), 2003.

[Alexander 77] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, S. Angel, *A Pattern Language*, Oxford University Press, New York 1977.

[Banerjee 87] Banerjee, J., Kim, W., Kim, H.-J., Korth, H. F.: Semantics and Implementation of Schema Evolution in Object Oriented Databases In Proc. of the ACM-SIGMOD Annual Conference, pages 311--322, San Francisco, CA, May 1987.

[Ben Ayed 02] M. Ben Ayed, *Lyee Program Execution Patterns*, Proceeding of Lyee-W02: New Trends in Software Methodologies, Tools and Techniques, pp 212-224, Paris, 2002.

[Ben Ayed 03a] M. Ben Ayed, S. Nurcan, *Technical Report TR2-1*, Lyee International Research Project, Université Paris 1, 2003.

[Ben Ayed 03b] M. Ben Ayed, S. Nurcan, *Technical Report TR2-2*, Lyee International Research Project, Université Paris 1, 2003.

[Ben Ayed 04] M. Ben Ayed, J. Ralyté, C. Rolland, *Constructing the Lyee Method with a Method Engineering Approach*, Knowledge Based Systems Journal (17), pp. 239-248, Elsevier, 2004.

[Benali 89] K. Benali, N. Boudjlida, F. Charoy, J. C. Derniame, C. Godart, Ph. Griffiths, V. Gruhn, Ph. Jamart, D. Oldfield, F. Oquendo, *Presentation of the ALF project*, Proceedings of the International Conference on System Development Environments and Factories, 1989.

[Benjamen 99] A. Benjamen, *Une Approche Multi-démarches pour la modélisation des démarches méthodologiquesi.* Thèse de doctorat en informatique de l'Université Paris 1, 6 octobre 1999.

[Boehm 87] B. Boehm, *Improving Software productivity,* IEEE Computer, September, pp 43-57, 1987.

[Boehm 88] B. Boehm, *A Spiral Model of Software Development and Enhancement*, IEEE Computer 21, 5,1988.

[Booch 91] G. Booch, *Object Oriented Analysis and Design with Applications*, Benjamin/Cummings, 1991.

[Booch 98] G. Booch, I. Jacobson et J. Rumbaugh, *Unified Modeling Language User Guide*, Addison Wesley, 1998.

[Boyd 04] M. Boyd, P. McBrien, *Towards a Semi-Automated Approach to Intermodel Transformation*, Proceeding of EMMSAD'04 Workshop, Riga, Latvia, 2004.

[Brinkkemper 01] S. Brinkkemper, M. Saeki, F. Harmsen, *A Method Engineering Language for the Description of Systems Development Methods*, Proceeding of CAiSE'01, pp. 473-476, 2001.

[Brinkkemper 90] S. Brinkkemper, *Formalisation of information systems modelling*, Phd. Thesis, University of Nijmegen, Thesis Publishers, Amsterdam, 1990.

[Brinkkemper 95] S. Brinkkemper, *Method Engineering : Engineering of Information Systems Developments Methods and Tools*, Information & Software Technology, 37 (11) pp. 1-6, 1995.

[Brinkkemper 96] S. Brinkkemper, *Method engineering : engineering of information systems development methods and tools*, Information and Software Technology, Vol. 38, No.4, pp.275-280, 1996.

[Brinkkemper 98] S. Brinkkemper, M. Saeki, F. Harmsen, *Assembly Techniques for Method Engineering*, Proceedings of the 10th Conference on Advanced Information Systems Engineering, CAiSE'98. Pisa Italy, 8-12 June, 1998.

[Brinkkemper 99] S. Brinkkemper, M. Saeki, F. Harmsen, *Meta-Modelling Based Assembly Techniques for Situational Method Engineering,* Information System 24(3): 209-228 (1999).

[Charrette 86] R.N. Charrette, *Software Engineering Environments - Concepts and Technology*

Intertext / McGraw-Hill, 1986

[Coad 91] P. Coad, E. Yourdon, *Object-Oriented Analysis*. Yourdon Press, 1991.

[Coad 92] D. Coad, *Object oriented patterns*. Communications of the ACM, Vol. 35, No. 9, pp. 152-159, 1992.

[Coplien 00] James O. Coplien, "Software Patterns", Lucent Technologies, Bell Labs Innovations, SIGS Books & Multimidia, New York, 2000.

[Coplien 95] J.O. Coplien, D.O. Schmidt (ed.) *Patron Languages of Program Design*

Addison-Wesley, Reading, MA (1995)

[Curtis 88] B. Curtis, M. Kellner, J. Over, *A Field Study of the Software Design Process for Large Systems*, Com. ACM, Vol. 31, No. 11, 1988.

[Curtis 92] B. Curtis, M. I. Kellner, J. Over : *Process Modelling*. Communications of the ACM, Vol. 35, N° 9, pp 75-90, 1992.

[Dardenne 91] A. Dardenne, S. Fickas, A. van Lamsweerde, *Goal – directed concept acquisition in requirements elicitation* . Proceedings of the 6th IEEE Workshop System Specification and Design, Como, italy, 14-21, 1991.

[Debenhan 03] J. Debenhan, B. Henderson-Sellers, *Designing Agent-based Process Systems – Extending the OPEN Process Framework*, Chapter VIII in Intelligent Agent Software Engineering (ed. V. Plekhanova), Idea Group Publishing, pp. 160-190, 2003.

[Deneckere 01] R. Deneckere, *Approche d'extension de méthodes fondée sur l'utilisation de composants génériques*, Thèse de doctorat, Université Paris 1, 2001.

[Deneckere 02] R. Deneckere, *Using Meta-patterns to Construct Patterns, Proceeding of the Conference on Object-Oriented Information Systems*, OOIS'2002, Springer, France, 2002.

[Dowson 88] M. Dowson, *Iteration in the Software Process*, Proceedings of the 9th International Conference on Software Engineering, 1988.

[Emmerich 91] W. Emmerich, G. Junkermann, B. Peuschel, W. Schäfer, S. Wolf, *MERLIN: Knowledge based process modelling.* 1st European Workshop on Software Process Modeling. A. Fuggetta, R. Conradi, V. Ambriola (Eds), Mila, Italy, May 1991.

[Finkelstein 90] A. Finkelstein, J. Kramer, M. Goedicke, *ViewPoint Oriented Software Development*, Actes de Conférence "Le Génie Logiciel et ses Applications", Toulouse, p 337-351, 1990.

[Firesmith 02] D.G. Firesmith, B. Henderson-Sellers, *The OPEN Process Framework. An Introduction*, Addison-Wesley, Harlow, Herts, UK, 2002.

[Flood 90] R.L. Flood, E.R. Carson, *Dealing with Complexity : An introduction to the theory and Application of Systems Science,*New York : Plenum Press, 1990

[Fowler 97a] M. Fowler, *Analysis Patterns : reusable objects models*, Addison-Wesley, 1997.

[Fowler 97b] M. Fowler, *UML distillated: applying the standard object modeling notation*, Addison-Wesley, Reading, MA, 1997.

[Franckson 91] M. Franckson, C. Peugeot, *Specification of the object and process modeling language,* ESF Report n° D122-OPML-1.0, 1991.

[Gamma 93] E. Gamma, Richard Helm, Ralph Johnson, Jhon Vlissides, *Design Patterns: Abstraction and Reuse of Object-Oriented Design*, 7th European Conference on Object Oriented Programming ECOOP'93, Spring Verlag, 1993.

[Gamma 94] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns : Elements of Reusable Object-Oriented Software*, Addisson-Wesley, 1994

[GOF 97] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns : Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional Computing Series, 1997.

[Grosz 97] G. Grosz, C. Rolland, S. Schwer, C. Souveyet, V. Plihon, S. Si-Said, C. Ben Achour, C. Gnaho, *Modelling and Engineering the Requirements Engineering Process : an overview of the NATURE approach*. Requirements Engineering Journal 2, pp. 115-131, 1997.

[Grundy 96] J.C. Grundy, J.R.Venable, *Towards an integrated environment for method engineering*, Proceeding of IFIP WG 8.1 Conference on method Engineering, Chapman and Hall, pp 45-62, 1996.

[Gupta 01] D. Gupta, N. Prakash, *Engineering Methods from Method Requirements Specifications*, Requirements Engineering Journal, Vol.6, pp.135-160, 2001.

[Harmsen 94] A.F. Harmsen, S. Brinkkemper, H. Oei, *Situational Method Engineering for Information System Projects*. In Olle T. W. and A. A. Verrijn Stuart (Eds.), Methods and Associated Tools for the Information Systems Life Cycle, Proceedings of the IFIP WG8.1 Working Conference CRIS'94, pp. 169-194, North-Holland, Amsterdam, 1994.

[Harmsen 95a] F. Harmsen, S. Brinkkemper, *Description and Manipulation of Method Fragments for Situational Method Assembly,* Proceedings of the Workshop on Management of Software Projects, Pergamon Press, London, 1995.

[Harmsen 95b] F. Harmsen, S. Brinkkemper, *Design and Implementation of a Method Base Management System for Situational CASE Environment*, Proceeding of 2nd APSEC Conference, IEEE Computer Society Press, pp 430-438, 1995.

[Harmsen 96] F. Harmsen, M. Saeki*, Comparison of four method engineering languages*, IFIP 8.1 Conference on Method Engineering, 1996, Chapman and Hall, pp 209-231, 1996.

[Hay 96] D. Hay, *Data Model Patrons : Conventions of Thought,* Dorset House, NY, 1996.

[Henderson 00] B. Henderson-Sellers, *The OPEN Framework for Enhancing Productivity*, IEEE Software 17(2): (2000).

[Henderson 04] B. Henderson-Sellers, J. Debenham, Q. –N.N. Tran, *Adding Agent-Oriented Concepts Derived from Gaia to Agent Open*, Proceedings of the 16th Conference on Advanced Information Systems Engineering, (CAISE'04), Springer Verlag (pub), pp 98-111, 2004.

[Henderson 90] B. Henderson-Sellers, J.M. Edwards, *The Object-oriented Systems Life-Cycle*, Communications of the ACM (9), 1990.

[Henderson 98] B. Henderson-Sellers, A.J.H. Simons, H. Younessi, *The OPEN Toolbox of Techniques*, Addison-Wesley, UK, 1998.

[Heym 92] M. Heym, H. Österle, *A Reference Model of Information Systems Development*, The Impact of Computer Supported Technologies on Information Systems Development, K.E. Kendall, K. Lyytinen and J.I. DeGross (Eds.), Amsterdam, North-Holland, pp. 215-240, 1992.

[Hofstede 93] A.H.M. ter Hofstede, *Information Modelling in Data Intensive Domains,* Dissertation, University of Nijimegen, the Netherlands, 1993.

[Hong 93] S. van der Hong, G. Goor, S. Brinkkemper, *A Formal approach to the comparaison of Object-Oriented Analysis et Design Methodologies*, Proceeding of the 26th Hawaii

international Conference on System Science, J. Numamaker, R. Sprague (Eds.), 4, IEEE Computer Society Press.

[Jarke 92] M. Jarke, K. Pohl, *Information systems quality and quality information systems*. Proceedings of the IFIP 8.2 Working Conference on the impact of computer-supported techniques on information systems development, Minneapolis, NM, 1992.

[Jarke 99] M. Jarke, C. Rolland, A. Sutcliffe, R. Domges, *The NATURE requirements Engineering*, Shaker Verlag, Aachen 1999.

[Karlsson 02] F. Karlsson, *Meta-Method for Method Configuration : A Rationale Unified Process Case*, Licentiate Thesis, Linköping University, Linköping (2002).

[Katamaya 89] T. Katayama, *A hierarchical and functional software process description and its enaction*, Proceedings of the 11th International Conference on Software Engineering, pp. 343-352, 1989.

[Kelly 96] S. Kelly, K. Lyyttinen, M. Rossi, *Meta Edit+: A fully configurable, multi-user and multi-tool CASE and CAME environment*, Proceedings of the CAiSE'96 Conference, 20-24 may, Heraklion, Crete, Greece, Springer Verlag, 1996.

[Kronlof 93] K. Kronlof, *Method Integration, Concepts and Case studies* , Wiley series in software based systems, John Wiley and sons Ltd., 1993.

[Kumar 92] K. Kumar, R.J. Welke, *Methodology Engineering : A Proposal for Situation-Specific Methodologies Construction,* In Challenges and Strategies for Research in System Development, Cotterman, W.W. and Senn, J.A. eds. , Wiley & Sons Ltd., 1992.

[Longworth 92] G. Longworth, *Introducing SSADM Version 4,* NCC Blackwell, Oxford, 1992.

[LyeeALLmanuel] "*Introduction to Lyee, Revolution by Automatic Software Development*", Fascicle, booklet produced by The institute of Computer Based Software Methodology and Technology, 2001.

[Marttiin 95] P. Marttiin, K. Lyytinen, M. Rossi, V-P. Tahvanainen, J-P. Tolvanen, *Modeling requirements for future CASE : issues and implementation considerations*. Information Resources Management Journal 8 (1), pp. 15-25, 1995.

[MetaModel.com] Web site, http://www.MetaModel.com.

[MOF 02] *Meta-Object Facility (MOF™) Specification*, version 1.4, http://www.omg.org/technology/documents/formal/mof.htm.

[Muller 97] P. Muller, *Modélisation objet avec UML*, Eyrolles, 1997.

[Mylopoulos 90] J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis , *Telos : Representing Knowledge About Information Systems*. ACM Transactions on Information Systems, 8, 4, pp. 325-362, 1990.

[Negoro 00] F. Negoro, Principle of Lyee Software, Proceedings of the International Conference on Information Society in the 21st Century, 2000.

[Negoro 01a] F. Negoro, '*Methodology to Determine Software in a Deterministic Manner*', Proceeding of ICII, Beijing, China, 2001.

[Negoro 01b] F. Negoro, '*A proposal for Requirement Engineering*', Proceeding of ADBIS, Vilnius, Lithuania, 2001.

[Negoro 01c] F. Negoro, '*The Predicate Structure to Represent the Intention for Software*', Software Engineering, Artificial Inteligence, Networking & Parallel / Distributed Computing (SNPD'01), 2001.

[Negoro 01d] F. Negoro, I. H. Fujita, *A Proposal for Intention Engineering*, International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, 2001.

[Nurcan 02] S. Nurcan, M. Ben Ayed, C. Rolland, *Scientific Report SC1*, Lyee International Research Project, Université Paris 1, 2002.

[Nurcan 03a] S. Nurcan, J. Barrios, *Entreprise Knowledge and Information System Modelling in an Evolving Environment*, Workshop on Engineering Methods to Support Information Systems Evolution (EMSISE'03), Geneva, Switzerland, 2003.

[Nurcan 03b] S. Nurcan, M. Ben Ayed, C. Rolland, *Process of Mapping between User Centric Concepts and Lyee Internal Concepts,* Proceeding of Lyee-W03 New Trends in Software Methodologies, Tools and Techniques, pp 320-336, Stockholm, 2003.

[Odell 96] J.J. Odell, *A Primer to Method Engineering*, in Method Engineering : Principle of method construction and tool support (IFIP TC8, WG8.7/8.2 Working conference on method engineering), Atlanta, USA, 1996.

[Oei 92] J.L.H. Oei, L.J.G.T. van Hemmen, E.D. Falkenberg, S. Brinkkemper, T*he Meta Model Hierarchy : A Framework for Information Systems Concepts and Techniques*, University of Nijmegen, 1992.

[Oei 94] J.L.H. Oei, E.D. Falkenberg, *Harmonisation of information systems modelling and specification techniques*. In Methods and Associated Tools for the Information Systems Life Cycle, pp. 151-168, A.A. Verrijn-Stuart and T.W. Olle (Eds.), No. A-55, Elsevier Science publishers, 1994.

[Oei 95] J.L.H. Oei, *A meta model transformation approach towards harmonisation in information system modeling.* In Information Systems Concepts – towards a consolidation of views, pp. 106-127, E.D. Falkenberg, W. Hesse and A. Olivé (Eds.), Chapman & Hall, London, 1995.

[Olle 88] T. W. Olle, J. Hagelstein, I. MacDonald, C. Rolland, F. Van Assche, A. A. Verrijn-Stuart, *Information Systems Methodologies : A Framework for Understanding*, Addison Wesley (Pub.), 1988.

[Plihon 94] V. Plihon, *The OMT Methodology, The OOA Methodology (Coad/yourdon), The SA/SD Methodology, The Entity Relationship Methodology, The O* Methodology, The OODMethodology,* NATURE Deliverable DP2, 1994.

[Plihon 95] V. Plihon, C. Rolland, *Modelling Ways-of-Working*, Proceeding of the 7th International Conference on Advanced Information Systems Engineering, CAISE'95, Springer Verlag (Pub.), 1995.

[Plihon 96] V. Plihon, *Un environnement pour l'ingénierie des méthodes*, Thèse de doctorat de l'Université Paris 1, janvier 1996.

[Poli 02] R. Poli, I. H. Fujita, R. IJIOUI, *The World of Lyee*, The 1st International workshop on New Software Methodologies, Tools and Techniques, SoMet'02, Paris, 2002.

[Potts 89] C. Potts, *A Generic Model for Representing Design Methods*. Proceedings of the 11th International Conference on Software Engineering, 1989.

[Potts 94] C. Potts et al., *Inquiry-based Requirements Analysis* , IEEE Software, Mars 1994.

[Prakash 02] N. Prakash, M. P. S. Bhatia, *Generic Models for Engineering Methods of Diverse Domains*, *Proceedings of CAISE'02,* Toronto, Canada, LNCS 2348, pp. 612, 2002.

[Prakash 94] N. Prakash, *A Process View of Methodologies* , 6th International Conférence on Advanced Information Systems Engineering CAISE'94, Springer Verlag, 1994.

[Prakash 99] N. Prakash, *On Method Statics and Dynamics,* Information Systems. Vol.34 (8), pp 613-637, 1999.

[Prat 97] N. Prat, *Goal formalisation and classification for requirements engineering*. Proceedings of the third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ'97, Barcelona, pp. 145-156, 1997.

[Prat 99] N. Prat, *Réutilisation de la trace par apprentissage dans un environnement pour l'ingénierie des processus*, Thèse de doctorat en informatique de l'université Paris 1, 1999.

[Punter 96] H.T. Punter, K. Lemmen, *The MEMA model : Towards a new approach for Method Engineering*, Information and Software Technology, Vol. 38, No.4, pp.295-305, 1996.

[Ralyté 01a] J. Ralyté, C. Rolland, *An Assembly Process Model for Method Engineering,* Proceedings of the 13th International Conference on Advanced Information Systems Engineering CAISE'01, Interlaken, Switzerland, 2001.

[Ralyté 01b] J. Ralyté, *Ingénierie des méthodes à base de composants*, Thèse de doctorat en informatique, Université paris 1, 2001.

[Ralyté 03a] J. Ralyté, C. Rolland, M. Ben Ayed, *An Approach for Evolution-Driven Method Engineering,* Proceeding of EMMSAD, pp 208- 217, Austria, 2003.

[Ralyté 03b] J. Ralyté, R. Deneckère, C. Rolland, *Towards a Generic Model for Situational Method Engineering*, Proceedings of the 15th International Conference CAISE'03, Klagenfurt/Velden, Austria, LNCS 2681, Springer, pp. 95-110, 2003.

[Ralyté 04] J. Ralyté, C. Rolland, R. Deneckère, *Towards a Meta-tool for Change-centric Method Engineering: a Typology of Generic Operators*, Proceedings of the 16th Conference on Advanced Information Systems Engineering CAISE'04, 2004.

[Ralyté 05] J. Ralyté, C. Rolland, M. Ben Ayed, *An Approach for Evolution-Driven Method Engineering*, Chapitre 5 dans le livre "Information Modeling Methods and Methodologies", IDEA Group (pub), pp. 80-100, 2005.

[Robert 86] P. Robert, *Le petit robert 1 : Dictionnaire alphabétique et analogique de la langue française,* Dictionnaires Le Robert, 1986

[Rolland 01] C. Rolland, N. Prakash, *Matching ERP System Functionality to Customer Requirements*, Proceedings of the 5th International Symposium on Requirements Engineering (RE'01), Toronto, Canada, pp. 66-75, 2001.

[Rolland 02a] C. Rolland, M. Ben Ayed, *Understanding the Lyee Methodology through Meta Modelling*, Proceeding of EMMSAD, pp 95-103, Toronto, 2002.

[Rolland 02b] C. Rolland, *A User Centric View of Lyee Requirements*, Proceeding of Lyee-W02: New Trends in Software Methodologies, Tools and Techniques, Paris, 2002.

[Rolland 02c] C. Rolland, C. Souveyet, M. Ben Ayed, *Users Requirements Elicitation in the Lyee Software Factory*, Proceeding of SCI, Orlando, 2002.

[Rolland 02d] C. Rolland, *Lyee Project Technical Report TR1-2*, Lyee International Research Project, Université Paris 1, 2002.

[Rolland 03] C. Rolland, C. Souveyet, M. Ben Ayed, *Guiding Lyee user requirements capture*, Knowledge Based System Journal, Volume 16, Issues 7-8, Intention and Software Process, pp 351-359, 2003.

[Rolland 04] C. Rolland, C. Salinesi, A. Etien, *Eliciting Gaps in Requirements Change, Requirements Engineering Journal (REJ)*, 9:1, pp. 1 – 15, 2004.

[Rolland 88] C. Rolland, O. Foucault, G. Benci, *Conception des systèmes d'information: La méthode REMORA*, Eyrolles, 1988.

[Rolland 94] C. Rolland, N. Prakash, *A Contextual Approach to modeling the Requirements Engineering Process*, SEKE'94, 6th International Conference on Software Engineering and Knowledge Engineering, Vilnius, Lithuania, 1994.

[Rolland 95] C. Rolland, C. Souveyet, M. Moreno. *An Approach for Defining Ways-Of-Working*, in the Information Systems Journal, 1995.

[Rolland 96a] C. Rolland, V. Plihon, *Using Generic Chunks to Generate Process Models Fragments,* Proceeding of 2nd IEEE International Conference on Requirements Engineering", ICRE'96, Colorado Spring, 1996.

[Rolland 96b] C. Rolland, N. Prakash, *A Proposal for Context-Specific Method Engineering*, Proceedings of the IFIP WG 8.1 Conference on Method Engineering, Chapman and Hall, pp 191-208, Atlanta, Gerorgie, USA, 1996.

[Rolland 98] C. Rolland, G. Grosz, S. Nurcan, W. Yue, C. Gnaho, *An electronic handbook for acessing domain sprecific generic patterns*, Working Conference on Information Systems in the www Environment, beijing, China, 15-17 July 1998.

[Rolland 98c] C. Rolland, *A Comprehensive View of Process Engineering*, Proceedings of the 10th International Conference CAISE'98, B. Lecture Notes in Computer Science 1413, Springer Verlag, Pernici, C. Thanos (Eds), Pisa, ITALY, June 1998.

[Rolland 99] C. Rolland, N. Prakash, A. Benjamen, *A Multi-Model View of Process Modelling*, Requirements Engineering Journal, Vol 4, No 4, 169-187, 1999.

[Rossi 96] M. Rossi, S. Brinkkemper, *Complexity Metrix for Systems Development Methods and Techniques*, Information Systems, 21(2), pp. 209-227, 1996.

[Rossi 99] M. Rossi, S. Kelly, *Construction of a CASE tool: the case for MetaEdit+,* in Proceeding of The 1st International Symposium on Constructing Software Engineering Tools (CoSET'99), USA, 1999.

[Royce 70] W. W. Royce, *Managing the development of large software systems*; Proceedings of the IEEE WESCON, August, 1970.

[Rumbaugh 91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-Oriented Modeling and Design*. Prentice Hall, 1991.

[Rumbaugh 95] J. Rumbaugh, M. Blaha, F. Eddy, W. Premerlani, and W. Lorensen, *OMT Modélisation et conception orientées objet*. Masson Paris and Prentice Hall London, 1995.

[Saeki 02] M. Saeki, *Role of Model Transformation in Method Engineering*, Proceeding of CAISE'02, Toronto, Canada, pp 626-642, 2002.

[Saeki 03a] M. Saeki, *Embeding Metrics into Information Systems Development Methods: An Application of Method Engineering Technique*, Proceedings of the 15th International Conference CAISE'03, Velden, Austria, LNCS 2681, Springer, pp. 374-389, 2003.

[Saeki 03b] M. Saeki, *Toward Automated Method Engineering : Supporting Method Assembly in CAME*, Invited talk in the International Workshop on Engineering Methods to Support

Information Systems Evolution (EMSISE'03), Geneva, Switzerland, 2003.

[Saeki 91] M. Saeki, T. Kaneko, M. Sakamoto, *A Method for Software Process Modelling and Description using LOTOS*, Proceeding of the 1st Intl. Conference on the Software Process, IEEE Computer Society Press, Los Alamitos, CA, USA, pp 90-104, 1991.

[Saeki 93] M. Saeki, K. Iguchi, K Wen-yin, M Shinohara, *A meta-model for representing software specification & design methods*, Proceeding of the IFIP¨WG8.1 Conference on Information Systems Development Process, pp 149-166, 1993.

[Saeki 94] M. Saeki, K. Wen-yin, *Specifying Software Specification and Design Methods,* Proceedings of Conference on Advanced Information Systems Engineering, CAISE'94, Lecture Notes in Computer Science 811, Springer Verlag, pp. 353-366, Berlin, 1994.

[Salinesi 02] C. Salinesi, M. Ben Ayed, S. Nurcan, *Development Using Lyee A Case Study with LyeeALL*, Technical Report TR1-2, Lyee Industrial Project, University Paris1, 2001.

[Shlaer 88] S. Shlear, S. J. Mellor, *Object Oriented Systems Analysis,* Prentice-Hall, 1988.

[Shlaer 92] S. Shlear, S. J. Mellor, *Object Lifecycles. Modeling the World in States*. Prentice-Hall, 1992.

[Si-Said 96] S. Si-said, G. Grosz, C. Rolland, *Mentor, A computer Aided Requirements Engineering Environment*, Proceedings of the 8th CAISE Conf. Chalenges In Modern Information Systems, Heraklion, Crete, Greece, May 1996.

[Si-Said 99] S. Si-Said, *Proposition pour la modélisation et le guidage des processus d'analyse des systèmes d'information*, Thèse de doctorat en informatique de l'université Paris 1. Février 1999.

[Smolander 91] K. Smolander, K. Lyytinen, V. Tahvanainen, P. Marttiin, *MetaEdit – A Flexible Graphical Environment for Methodology Modelling*, Proceedings of the 3th International Conference in Advanced Information Systems Engineering CAISE'91, Trondheim, Norway, May 1991.

[Smolander 92] K. Smolander, *OPRR – A Model for Methodology Modeling*, Next Generation of Communication Systems, IOS press, pp. 224-239, 1992.

[Song 92] X. Song, L.J. Osterweil, *Towards objective, systematic, design-method comparison*, IEEE Software, Vol. 34, No.5, May, pp. 43-53, 1992.

[Song 97] X. Song, *Systematic Integration of Design Methods*, IEEE Software, 1997.

[Sorenson 88] P.G. Sorenson, J.P. Tremblay, A.J. McAllister, *The Metaview System for Many Specificatio Environments*, IEEE Software, pp. 30-38, 1988.

[Sowa 92] J.F. Sowa, J.A. Zachmen, *Extending and Formalising the Framework for Information Systems Architecture*, IBM Systems Journal, Vol. 31, No. 3, pp. 590-616, 1992.

[Tardieu 83] H. Tardieu, A. Rochfeld, R. Coletti, *La méthode Merise : Tome 1, Principe et outils,* Editions d'organisation, 1983.

[Tawbi 01] M. Tawbi, *CREWS-L'Ecritoire: un guidage outillé du processus d'ingénièrie des besoins*, Thèse de doctorat de l'université Paris 1, 2001.

[Terrase 03] M. N. Terrasse, M. Savonnet, G. Becker, E. Leclercq, *UML-based Metamodeling for Information System Engineering and Evolution*, Proceeding of the 9th International Conference on Object-Oriented Information Systems OOIS'03, Switzerland 03.

[Tolvanen 03] J.-P. Tolvanen, M. Rossi, *MetaEdit+: Defining and Using Domain-Specific Modeling Languages and Code Generators*, Proceeding of OOPSLA, USA, pp. 92-93, 2003.

[Tolvanen 93] J.-P. Tolvanen, K. Lyytinen, *Flexible method adaptation in CASE environements – The metamodeling approach*, Skandinavian Journal on Information System, Vol. 5(1), pp. 51-77, 1993.

[Tolvanen 98] J.-P. Tolvanen, *Incremental Method Engineering with Modeling Tools: Theoretical Principles and Empirical Evidence*, PhD Dissertation, University of Jyväskylä, Finland, 1998.

[Turner 87] J.A. Tuerner, *Understanding the Elements of system design,* R.J. Boland jr. R.A. Hirshheimù (eds) Critical issues i information systems research, Chichester : Wiley, pp 97-112, 1987.

[UML 03] *Unified Modeling Language (UML) specification, version 1.5, March 2003,* http://www.omg.org/technology/documents/formal/uml.htm.

[Van Slooten 96] K. van Slooten, B. Hodes, *Characterising IS development project*, IFIP WG 8.1 Conference on Method Engineering, Chapman and Hall, pp 29-44, 1996.

[Verhoef 95] T.F. Verhoef, A.H.M. ter Hofstede, *Feasibility of Flexible Information Modelling Support*, Advanced Informtion Systems Engineering, J. Iivari, K. Lyytinen and M. Rossi (Eds.), Springer-Verlag, pp. 168-185, 1995.

[Vlissides 96] J.M. Vlissides, J.O. Coplien, N.L. Kerth (ed.), *Pattern Languages of Program Design 2,* Addison-Wesley (1996)

[Wijers 91] G. M. Wijers, *Modeling Support in Information Systems Development*, PhD Thesis, Thesis Publishers Amsterdam, 1991.

[Wistrand 04] K. Wistrand, F. Karlsson, *Method Components – Rationale Revealed*, 16th International Conference on Advanced Information System Engineering (CAISE'04), Riga, Latvia, pp 189-201, June 2004.

[Wynekoop 93] J. Wynekoop, N. Russo, *System development methodologies:unanswered questions and the research-practice gap*, in Proc.14th Intl. Conf. Inf. Syst., New York, ACM Pub. pp 181-190,1993.

[Yu 94] E. Yu, J. Mylopoulos, *From ER to AR_modelling strategic Actor Relationships for Business*

Process Reengineering . In Proceedings of the 13th International Conference on the Entity-Relationship Approach – ER'94, Manchester (UK), 1994.

[Zall 92] R. Zaal, *Method Assistants: Instant Method Knowledge on Screen*, Informatic management, 1992

[Zoukar 05] I. Zoukar, MIBE : *Méthode d'Ingénierie des BEsoins pour l'implantation d'un Progiciel de Gestion Intégré*, Thèse de doctorat de l'université Paris 1.