INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

THESE
pour obtenir le grade de

Docteur De L'Institut National Polytechnique de Grenoble

Spécialité : Génie Industriel

préparée au laboratoire Gestion Industrielle, Logistique et COnception (GILCO)
dans le cadre de **l'Ecole Doctorale :** ***Organisation Industrielle et Systèmes de Production***

présentée et soutenue publiquement
par

Lilia GZARA

le 12 décembre 2000

Titre :

LES PATTERNS POUR L'INGENIERIE DES SYSTEMES D'INFORMATION PRODUIT

Directeurs de thèse : Dominique RIEU & Michel TOLLENAERE

JURY

Mme. Corine CAUVET	Présidente
M. Michel LEONARD	Rapporteur
M. François VERNADAT	Rapporteur
M. Alain BERNARD	Examinateur
Mme. Dominique RIEU	Co-directeur de thèse
M. Michel TOLLENAERE	Co-directeur de thèse

Table des Matières

Chapitre VII : Expérimentation du Catalogue de Patrons

Conclusion Générale

Liste des Figures

Chapitre I

Chapitre II

Chapitre III

Chapitre IV

Chapitre V

Chapitre VI

Chapitre VII

Liste des Tableaux

Liste des Abréviations

CIM	**C**ircuit **I**mprimé **M**onté
CIP	**C**ircuit **I**mprimé **P**ercé
OCL	**O**bject **C**onstraint **L**anguage
PDM	**P**roduct **D**ata **M**anagement
SGDT	**S**ystème de **G**estion de **D**onnées **T**echniques
SI	**S**ystème d'**I**nformation
SII	**S**ystème d'**I**nformation **I**nformatique
SIO	**S**ystème d'**I**nformation **O**rganisationnel
SIG	**S**ystème d'**I**nformation de **G**estion
SIP	**S**ystème d'**I**nformation **P**roduit
STEP	**ST**andard for **E**xchange of **P**roduct data model
UML	**U**nified **M**odeling **L**anguage

Introduction Générale

Le système d'information de l'entreprise constitue un véhicule de communication dans l'entreprise et entre l'entreprise et son environnement. La vision systémique de l'entreprise issue des travaux de Jean-Louis Lemoigne situe le système d'information à l'interface entre d'une part le système opérant chargé de la production et répondant à la finalité de l'entreprise et d'autre part le système de pilotage qui dirige l'entreprise et maintient le cap sur les objectifs choisis. Le système d'information assure en effet le lien entre ces deux systèmes en informant le système de pilotage des performances du système opérant et inversement en transmettant au système opérant les instructions du système de pilotage. Une analyse plus fine du système opérant de l'entreprise met en évidence deux processus clef qui concourent à la réalisation de la finalité de l'entreprise :
- un premier processus de "Définition de l'offre produit" qui vise à *définir* et à *maintenir* des produits que l'entreprise met sur le marché. Il est important pour la réactivité de l'entreprise pour mettre continûment sur le marché des produits adaptés à une demande évolutive et de gagner ou maintenir ainsi des parts de marché.
- un second processus de "Production de l'offre produit" qui permet à partir d'une commande ou parfois d'un devis demandé par un client de *fournir* à temps et en qualité le produit le mieux adapté à son besoin.
Ces deux processus clef prennent une importance et une temporalité différente selon les types d'entreprise. Ainsi, une telle société de grande distribution n'a que peu d'actions sur son offre produit tandis que sa relation au client lui impose de bien maîtriser son processus de traitement des commandes (par une gestion performante de sa logistique, de ses procédures administratives liées à la réception des commandes, à la facturation, etc.). Une petite entreprise qui livre à chacun de ses clients un produit personnalisé et partiellement spécifiquement étudié devra bien maîtriser son processus de définition de l'offre (par une gestion performante de la configuration de ses divers produits et de la base informationnelle accompagnant le développement des produits, par une rationalisation de l'ensemble des tâches de développement de produits, etc.).
Evidemment, ces deux processus clef sont accompagnés par d'autres processus qui les supportent, tels que le processus commercial (obtention de la commande).

A ces deux processus clef du système opérant, correspondent des besoins différents d'accès aux informations sur les produits. Le processus "Production de l'offre produit" est essentiellement supporté par les systèmes d'information du type ERP (*Enterprise resource Planning*) et implique des informations fréquemment mises à jour sans que leur pérennité à long terme ne soit nécessairement importante. Le processus "Définition de l'offre produit" nécessite la mise en place de systèmes d'information du type PDM (*Product data Management*) que nous appelons SIP (Système d'Information Produit) et implique des informations dont la création et l'évolution sont moins fréquentes et font l'objet de procédures plus strictes (droits d'acteurs, statuts d'information). La pérennité à long terme de ces informations (à tous ses statuts et ses versions) est souvent importante.
C'est à ce deuxième système d'information d'entreprise que nous nous intéressons dans ce travail de thèse.

Les systèmes d'information produit sont devenus pour les entreprises (notamment d'ingénierie) un élément crucial pour supporter leur processus de définition de l'offre produit. En effet, avec la complexité de l'offre produit et dans un contexte d'ingénierie concourante, la maîtrise de l'information technique est devenue un sujet capital et difficile : capital car toute défaillance dans la gestion de l'information se traduit de façon immédiate par des non qualités; difficile compte tenu de la dynamique du système. Les SIP permettent ainsi une gestion performante de la base informationnelle accompagnant le développement de produits et ensuite une rationalisation de l'ensemble des tâches du processus de développement. La réactivité des entreprises aux modifications inhérentes au processus de développement de produits nécessite un pilotage de la circulation de l'information technique, une gestion rigoureuse de son évolution et s'accompagne d'une exigence liée à la maîtrise de la qualité, autant de fonctions qu'assure le SIP. Les entreprises se sont alors intéressées à mettre en place le SIP le mieux adapté à leurs besoins. Face à ce besoin, une offre progicielle appelée Systèmes de Gestion de Données Techniques (SGDT) s'est développée pour répondre aux attentes des entreprises industrielles.

Malgré l'importance des SIP dans l'entreprise, l'ingénierie de ces systèmes, càd les aspects méthodologiques liés à leur spécification et à leur implantation sur un SGDT, demeure un processus de support pour l'entreprise dont les coûts engendrés sont à minimiser. Les enjeux associés à ces aspects méthodologiques sont alors stratégiques. La thématique de recherche de ce mémoire de thèse concerne les aspects méthodologiques liés à l'ingénierie des SIP.

L'ingénierie des SIP reste un domaine complexe qui partant de l'analyse des besoins d'une application, aboutit à une solution logicielle fiable dans un système technologique cible choisi. Actuellement, la pratique industrielle de l'ingénierie des SIP dans les entreprises industrielles rencontre diverses difficultés techniques, méthodologiques, organisationnelles et humaines. Des difficultés qui constituent actuellement des goulots dans le processus de développement, engendrant des coûts considérables et menant dans beaucoup de cas à l'échec des projets SIP. Il s'agit notamment :

- de la lenteur du développement des SIP qui engendre des coûts considérables et mène parfois à la livraison de systèmes déjà obsolètes,
- du manque de modèles formels d'expression des besoins suffisamment compréhensibles pour assurer une communication efficace et non ambiguë entre les acteurs concernés. Ce manque de formalisation ralentit le processus d'ingénierie à cause des multiples itérations nécessaires pour préciser les besoins et valider les spécifications,
- de l'absence de continuum de transformations cohérentes des différents modèles élaborées durant les différentes phases d'ingénierie, ce qui mène à des incohérences entre ce qui est attendu et ce qui est livré.

Par ailleurs, malgré la richesse des développements théoriques effectués dans le domaine de l'ingénierie des SI, les travaux de recherche traitant de l'ingénierie des SIP n'en tirent pas profit. Un pont doit donc être établi entre les recherches dans le domaine de l'ingénierie des SI et celles relevant du domaine des systèmes de gestion de données techniques.

C'est sur ces hypothèses que se base donc la définition de notre problématique et le travail réalisé. Ainsi le travail de thèse présenté dans ce manuscrit a pour objectif de mettre en place un cadre méthodologique pour l'ingénierie des SIP en abordant trois aspects :

- La définition de *modèles de spécification* de différents niveaux d'abstraction, qui couvrent la complexité du SIP (représentation des processus, des versions de produit, etc.), favorisent l'échange d'information entre acteurs et assurent un continuum de transformation dans les spécifications ;
- La mise en place d'une *démarche générale de conduite de projet SIP* pour construire les différents modèles. Cette démarche doit couvrir l'ensemble des phases du processus d'ingénierie des SIP (analyse, conception, réalisation) et elle doit être flexible et adaptable au contexte de chaque organisation ;
- *L'accélération du processus* de développement des SIP. L'objectif est de favoriser la capitalisation et la réutilisation de savoir et de savoir-faire expérimentés et mis en œuvre dans des projets antérieurs.

Ce travail nécessite et implique des investigations dans deux orientations ou domaines complémentaires : d'une part dans le domaine du génie industriel pour capitaliser les connaissances autour des produits industriels et de leur processus de définition et d'évolution ; d'autre part dans le domaine de l'informatique et plus particulièrement des systèmes d'information pour définir une démarche d'ingénierie de SIP favorisant la réutilisation des connaissances associées et dotée de modèles de spécification adaptés aux besoins exprimés.

Ce travail de thèse a été mené en collaboration contractuelle avec l'entreprise Schneider Electric et s'est inscrit dans le cadre du projet POSEIDON, accepté dans l'appel d'offre CNRS Prosper.

Ce manuscrit est constitué de trois parties. La première partie définit le cadre de réflexion dans lequel se situe ce travail de thèse. Elle propose un état de l'art sur les SIP et leur ingénierie, précise la problématique et les objectifs de notre travail et définit les moyens pour approcher la problématique. La deuxième et la troisième partie décrivent notre proposition. Elles décrivent les éléments du cadre méthodologique développé et les résultats de son application dans l'entreprise.

La première partie est constituée alors de trois chapitres : le ***chapitre I*** présente les concepts fondamentaux du domaine de la gestion des données techniques et des systèmes d'information produit. Il propose ensuite un état de l'art sur l'ingénierie de tels systèmes, du point de vue des développements théoriques mais également du point de vue de la pratique industrielle. Il conduit alors à formuler les difficultés rencontrés dans le domaine de l'ingénierie des SIP. Ceci permet de préciser dans le ***chapitre II*** la problématique de notre travail de thèse et d'éclaircir le cahier des charges du travail proposé. Une investigation des techniques et des approches susceptibles de réaliser le cahier des charges ainsi posé permet par ailleurs de déterminer les éléments clés sur lesquels se base le cadre méthodologique proposé. Il s'agit de développer une démarche d'ingénierie de SIP basée sur la réutilisation de patrons et l'utilisation du langage de modélisation UML. Un patron constitue une base de savoir et de savoir-faire pour résoudre un problème récurrent dans un domaine donné. Le ***chapitre III*** définit alors la démarche pour construire des patrons pour le domaine des SIP. Cette démarche est constituée de deux étapes : d'abord une *analyse de domaine* pour identifier les problèmes récurrents dans l'ingénierie des SIP et ensuite une *spécification de patrons* pour résoudre les problèmes identifiés. Le chapitre III clôt ainsi le rapport sur le cadre de réflexion.

Il introduit les deux autres parties du manuscrit, relatives aux deux étapes de la démarche d'ingénierie de patrons proposée.

La deuxième partie du manuscrit illustre ainsi les résultats de l'analyse de domaine SIP. Elle est constituée de deux chapitres : le ***chapitre IV*** qui décrit le référentiel des données produit et le ***chapitre V*** qui décrit le référentiel des données processus.

La troisième partie identifie et spécifie alors les différents patrons du domaine SIP. Elle est constituée de deux chapitres : le ***chapitre VI*** qui présente le catalogue de patrons et le ***chapitre VII*** qui illustre l'expérimentation des patrons sur un projet industriel.

La conclusion de ce mémoire reprend les grandes lignes de cette étude et la contribution de l'approche proposée. Elle montre également les diverses prolongations possibles de ce travail.

Première Partie :

Etat de l'Art et Approche Proposée

L'objectif de cette partie est de présenter le cadre de réflexion dans lequel se situe ce travail de thèse. Elle propose un état de l'art sur les systèmes d'information produit et leur ingénierie, précise la problématique liée à leur ingénierie ainsi que les objectifs de notre travail et définit les moyens pour approcher la problématique.

Cette partie est structurée en trois chapitres. Le chapitre I propose un état de l'art sur les systèmes d'information produit et leur processus d'ingénierie. Cela permettra de dégager les difficultés actuellement rencontrées dans le domaine d'ingénierie de SIP. Le chapitre II précise alors la problématique et le cahier des charges de notre travail. Il puise ensuite dans la littérature des méthodes et des moyens susceptibles de réaliser le cahier de charges posé afin de fixer les éléments clés sur lesquels se base le travail proposé. Le chapitre III initie l'approche proposée.

Chapitre I :

Ingénierie des Systèmes d'Information Produit : Etat de l'Art

1. Introduction

Ce chapitre parcourt la littérature dans l'objectif de présenter un état de l'ingénierie des systèmes d'information produit. Il permet ainsi de situer le domaine de nos travaux.

Pour aborder cette revue, nous présentons d'abord les systèmes d'information produit (§2). Un bref historique sur l'évolution de la gestion des données techniques permet de fixer le "périmètre" actuel des systèmes d'information produit et de souligner la place qu'ils occupent dans l'entreprise industrielle (§2.1). Ensuite, les SIP seront décrits plus en détail à travers une présentation du concept de données techniques, sujet des SIP et une revue des principales fonctions de ces systèmes (§2.2). Enfin, un aperçu des outils informatiques mettant en œuvre les SIP est proposé (§2.3).

La deuxième partie de ce chapitre aborde la thématique de recherche de ce mémoire de thèse en se focalisant sur les aspects méthodologiques liés à l'ingénierie des SIP (§3). Elle est consacrée aux approches d'ingénierie des SIP. Nous en examinerons les aspects théoriques (§3.1) ainsi que la pratique industrielle (§3.2) afin de dégager les problèmes associés à ce processus d'ingénierie. Ceci nous permettra de déduire, dans le chapitre suivant (chapitre II), les objectifs de notre contribution dans ce domaine et préciser la problématique de cette thèse.

2. Système d'Information Produit

2.1. Evolution de la gestion des données techniques

Depuis les années 1970, l'industrie manufacturière a du s'investir d'une façon significative dans l'utilisation des technologies de l'information pour automatiser les divers processus intervenant dans le cycle de vie des produits. Ce besoin d'automatisation a contribué à l'apparition sur le marché d'une offre progicielle importante dédiée à la gestion automatisée de l'information d'entreprise. Ainsi, chaque fonction ou unité opérationnelle s'est vu faire appel à un ou plusieurs outils de gestion informatisés : les bureaux d'études ont fait appel aux outils de CAO pour la gestion des données de définition des produit (plans, dessins, modèles 3D, …), les services de production ont fait appel aux outils de GPAO pour la gestion des données de production (composants de produits, stocks,…), les services qualité ont eu recours aux outils de GQAO pour la gestion des données relatives à la qualité (des produits et des processus), les services de maintenance ont fait appel aux outils de GMAO pour la gestion des données de maintenance, les services de gestion ont adopté les outils de GED pour la gestion documentaire, etc.. L'entreprise manufacturière s'est trouvée alors dotée d'une multitude d'outils pour gérer de très nombreuses données destinées à différents acteurs.
Toutefois, le développement de ces outils en îlots indépendants a conduit au phénomène connu sous le nom des "Iles d'automatisation"[1], où aucune intégration des données communes n'est prise en compte. De ce fait, la gestion des données relatives au produit se trouve "morcelée" sur plusieurs systèmes où on ne fait que gérer des vues partiellement indépendantes du produit. Par ailleurs, la cohérence entre ces vues ne peut être assurée par un

[1] Traduction de : Islands of Automation [Scott, 93].

contrôle manuel des données à cause de la prolifération des données produit générées par ces divers outils en des temps réduits. Une intégration des données techniques relatives au produit s'est révélée alors nécessaire pour maintenir une gestion cohérente du produit. C'est à ce moment là que le concept de gestion de données techniques commence à faire son apparition dans la culture d'entreprise. Il s'agit d'une mutation culturelle dans l'entreprise qui jusqu'alors ne mettait l'accent que sur la maîtrise et la gestion des flux physiques de leurs produits. Initialement, la gestion des données produit ne fut qu'un effet de bord issu de la mise en place de divers outils optimisant les divers processus et générant ou utilisant certaines données produit. Ce n'est qu'au début des années 1980 que le besoin de gérer les données techniques du produit s'est révélé. Un besoin qui vient de la complexité de l'offre produit dans un contexte d'entreprise étendue.

Toutefois, ce besoin ne fut concrétisé au départ que pour les activités de conception de produit. Ainsi, au milieu des années 1980, une nouvelle classe d'outils est apparue sous le terme de Systèmes de Gestion des Données d'Ingénierie[2]. Leur objectif était d'établir des "passerelles" entre les îles d'automatisation. Cependant, ils concernent essentiellement la supervision des évolutions des plans CAO. Ces outils sont apparus initialement sous forme de fonctions adjointes aux outils existants de CAO/FAO/IAO, le tout englobé dans la même application. Cette solution d'intégration a toutefois restreint l'exploitation de ces outils puisque seules les données générées par les outils de CAO/FAO/IAO, c'est-à-dire issues des activités de conception étaient gérées. Ainsi, on a vu l'apparition, au début des années 1990 des outils de type Systèmes de Gestion des Données Produit[3] ou Systèmes de Gestion de Données Techniques (SGDT) qui visent à lever cette restriction en permettant de gérer la configuration de la quasi-totalité des données. Les SGDT se présentent sous la forme d'outils qui consolident et redistribuent l'ensemble des données produit des différents outils. La gestion des données techniques du produit, qui se présentait initialement sous forme de données encapsulées dans divers outils dédiés à diverses fonctions intervenant sur le produit, a donc évolué par la suite vers une gestion centralisée des données techniques à l'aide d'outils qui leur sont dédiés; les SGDT.

Depuis, et avec l'avancée des technologies d'information et informatiques, d'autres fonctionnalités ont été ajoutées, tels que la gestion de configuration[4], la visualisation d'images, la gestion des modifications, la gestion du travail collaboratif, la gestion de projet, etc.

Durant la dernière décennie, l'industrie a acquis une certaine expérience sur ces systèmes, de meilleures pratiques[5] ont été alors développés et de nouvelles technologies d'information et de communication (TIC) ont émergé. Les SGDT n'ont cessé ainsi d'évoluer en intégrant de nouvelles fonctionnalités et en profitant des nouvelles TIC (telles que les applications basées sur le web ; notion de client léger).

Dans ce bref historique, l'évolution de la gestion des données techniques a été présentée en fonction de l'évolution des technologies utilisées pour la supporter. Cependant, il ne faut pas perdre de vue que cette évolution était d'abord une évolution de la vision de la gestion des

[2] Traduction de : Engineering Data Management (EDM).

[3] Traduction de : Product Data Management System (PDMS).

[4] Il s'agit d'approches structurées pour définir et contrôler la structure du produit et des documents associés. Traduction de la définition donnée par CIMdata [CIMdata, 97] pour le concept de "Configuration Management".

[5] Connu sous le vocable de "Best Practices".

données techniques (GDT), due essentiellement aux mutations de l'entreprise et que la technologie disponible a permis ou non de la mettre en œuvre. Initialement, la GDT était vue comme une gestion des données informatisées du produit qui vise à maintenir la cohérence entre les données générées par les différents outils et qui se matérialise par la mise en place d'une solution logicielle. Ensuite, la GDT a évolué progressivement vers une démarche organisationnelle pour la maîtrise de l'ensemble du système d'information associé au produit et surtout pour "optimiser" le cycle de vie du produit. Cette démarche implique la mise en place de meilleures pratiques de travail, en plus du déploiement d'un ensemble de technologies (SGDT, techniques de visualisation, gestion des composants fournisseurs, gestion du travail collaboratif, etc.). Cette vision a évolué avec la nécessité de :

- décloisonner les différents acteurs intervenant sur le produit dans l'orientation "Concurrent Engineering" et "entreprise étendue" qui visent essentiellement à réduire les délais de mise sur le marché de nouveaux produits[6],
- diversifier et personnaliser l'offre produit selon les besoins des clients,
- mettre en œuvre rapidement, efficacement et de façon fiable les évolutions de toute nature qui peuvent concerner l'offre produit pour une meilleure réactivité face aux changements du marché (évolution et diversification des besoins),
- structurer et contrôler les données ainsi que leur évolution dans l'optique de la certification ISO 9000.

L'ensemble de ces visions et des différentes technologies employées ont concouru finalement à ce que nous convenons d'appeler Systèmes d'Information Produit[7] (SIP). Un SIP constitue un dispositif organisationnel permettant de réguler la création, la circulation, l'utilisation et l'évolution du patrimoine informationnel de définition du produit[8]. Le SIP se doit d'être intégré, c'est-à-dire que chaque information n'est saisie (en création ou en modification) qu'une seule fois par un seul acteur et accessible à tous[9]. On parle alors d'un référentiel unique des données produit qui fédère les fonctions de l'entreprise autour d'une vue partagée du produit.
Le SIP constitue désormais le support du processus de création et d'évolution de l'offre[10] et il est une composante permanente du système d'information de l'entreprise. La Figure 1.1 positionne le SIP par rapport au système d'information de l'entreprise.

Il est important tout de même de noter que le domaine de la gestion des données techniques n'a pas encore atteint sa maturité. Il manque encore du recul dans l'industrie sur les résultats que l'on peut attendre, le périmètre à couvrir fait encore l'objet de controverses et les offres ainsi que les offreurs ne cessent d'évoluer. Ainsi, l'ensemble des visions et des technologies est encore amené à évoluer. C'est ainsi qu'une des dernières visions de la GDT concerne le

---

[6] Le "Time to market" [Voirpin, 97], devenant un levier stratégique de différenciation dans le domaine de développement des produits.

[7] Ou encore système d'information technique par opposition au système d'information de gestion.

[8] Nous désignons par patrimoine informationnel de définition du produit, l'ensemble des informations qui définissent comment le produit est conçu, fabriqué, utilisé et recyclé.

[9] Définition de système intégré d'information, proposée par M. Randoing [Randoing, 95], p. 27.

[10] Allant des premières évocation des besoins utilisateurs et du concept de produit jusqu'à son retrait du portefeuille d'activités de l'entreprise (cessation des activités de support du produit).

concept de cPDm[11] ou gestion collaborative de définition de produit qui s'adresse aux entreprises étendues en considérant la totalité de la chaîne logistique de définition du produit (fournisseurs, sous-traitants, partenaires, clients) et en faisant appel à un ensemble de technologies facilitant la gestion des processus collaboratifs, l'intégration de fournisseurs, etc.

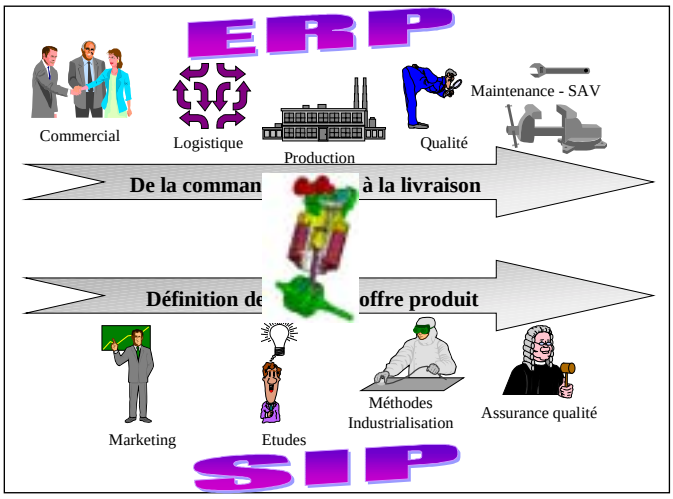


Figure 1.1 - Le SIP dans l'entreprise (*Source : [Tollenaere, 99]*)

Notons enfin que le terme le plus courant dans la littérature française pour désigner la gestion des données techniques est celui de SGDT, un terme qui reste restrictif. Dans la littérature anglo-saxonne, on parle plutôt de PDM. Ce dernier terme, bien que constituant une extension générale des technologies d'EDM (Engineering Data Management), TDM (Technical Data Management), TIM (Technical Information Management) ou encore PIM (Product Information Management), se trouve encore confus et on continue à utiliser parfois ces termes indifféremment pour désigner le PDM. Dans le courant des évolutions, le PDM ne constitue désormais qu'une technologie parmi d'autres pour supporter la gestion des données techniques.

2.2. Présentation des SIP

Comme il en ressort de l'historique précédemment présenté, le domaine de la gestion des données techniques n'est pas aujourd'hui stabilisé. Nous avons convenu alors de parler de Système d'Information Produit, une appellation générale caractérisant la gestion des données techniques du produit. Nous adoptons la définition suivante, préalablement introduite dans la section précédente :
Le système d'information produit est un dispositif organisationnel permettant de réguler la création, la circulation, l'utilisation et l'évolution du patrimoine informationnel de définition du produit, c'est à dire l'ensemble des informations qui définissent comment le produit est conçu, fabriqué et utilisé. Le support informatique de ce dispositif organisationnel est souvent réalisé à l'aide d'un Système de Gestion de Données Techniques (SGDT).

Cette définition introduit trois notions essentielles dans le SIP, que nous citons ci-dessous :
- les informations produit, décrivant le produit tel qu'il est conçu, fabriqué et supporté,

[11] collaborative Product Definition management [Portella, 00].

- les processus organisant la création, la circulation, l'utilisation et l'évolution de ces informations pour réguler leur création,
- les outils informatiques supportant la gestion des informations produit et des processus associés.

Notre définition est proche de celle donnée par CIMdata[12] qui décrit les méthodes et les systèmes PDM comme "une structure où toutes les informations utilisées pour définir, fabriquer et supporter les produits sont stockées, gérées et contrôlées"[13].

2.2.1. Informations Produit

Appelées également informations techniques ou données d'ingénierie[14], les informations produit sont connues le plus souvent sous le vocable de données techniques. Comme l'évolution des visions vis à vis de la gestion de données techniques, le périmètre couvert par les données techniques s'est vu également évolué.
Les premières définitions du concept de données techniques sont issues d'une tentative du *National Bureau of Standards* (NBS), dans les années 60, pour structurer "l'informatique technique" sous le vocable de CIM (*Computer Integrated Manufacturing*). Il découle de cette structuration que les données techniques sont celles spécifiques aux études, à l'ingénierie, la fabrication, la gestion de projets, la gestion de la qualité, la génération des notices, le soutien logistique. C'est à dire, pratiquement, toute l'informatique qui n'est pas qualifiée de "gestion" (gestion du personnel, paye, comptabilité, gestion financière, bureautique).
Cette définition englobe l'ensemble des informations techniques dans l'entreprise, relatives aussi bien aux produits qu'aux processus industriels. C'est pour cette raison, que nous préférons parler de données ou informations produit. Dans la littérature, plusieurs définitions ciblant les informations produit ont été proposées ([CIMdata, 97], [Maurino, 93], [Stark, 00], [Randoing, 95] [15]). Elles conviennent de définir celles-ci comme :
l'ensemble des informations relatives au produit et aux processus qui sont utilisées pour spécifier, développer, produire et supporter le produit.

Pour illustrer l'étendue de cette notion, nous reprenons l'exemple cité par M. Maurino [Maurino, 93] pour préciser la nature de l'information technique du produit qui, selon lui, ne doit pas être interprétée de façon restrictive. Ainsi, les quantités d'unités d'œuvre (temps opérateur, temps machine, etc.) nécessaires au chiffrage prévisionnel d'un sous-ensemble ou réellement consommées sur une période de production rentrent, par exemple, dans le champ de la gestion des données techniques; la valorisation en francs de ces unités d'œuvre, pour reprendre cet exemple, en est exclue parce que couverte par d'autres méthodes de gestion (la comptabilité analytique ou industrielle, en l'occurrence).

---

[12] CIMdata est un société de conseil américaine leader dans le domaine de la gestion du cycle de vie du produit. Elle mène également de la recherche pour l'industrie et organise des conférences dans le domaine. Pour plus d'informations, voir son site web http://www.cimdata.com

[13] Traduction de : PDM systems and methods provide a structure in which all types of information used to define, manufacture, and support products are stored, managed, and controlled.

[14] Traduction de : Engineering Data.

[15] Il élargit toutefois le périmètre des données techniques en définissant ces dernières comme toutes les informations qui se réfèrent à un produit depuis sa première évocation en stratégie d'entreprise jusqu'à sa destruction.

Les informations produit correspondent alors aux informations associées au ***cycle de définition de produit***, au sens de la classification de CIMdata [Portella, 00]. Le cycle de vie du produit comprend en effet trois processus de base (cf. Figure 1.2). Le premier processus est le *cycle de définition du produit* qui comme le cycle de vie global commence aux premières évocations du produit dans l'entreprise (concept produit, besoins utilisateurs) et s'étend jusqu'au retrait du produit du portefeuille d'activités de l'entreprise[16]. Il inclut la description complète du produit (des composants à la documentation) ainsi que l'ensemble des informations qui définissent comment le produit est conçu, fabriqué et supporté. Le second processus est le *cycle de production du produit* qui inclut toutes les activités associées à la production et la distribution du produit, mettant l'accent sur le produit physique ou matériel (contrairement au premier cycle relatif à une propriété intellectuelle). Le troisième processus est le *cycle de support opérationnel* qui se focalise sur les ressources nécessaires pour supporter les activités de l'entreprise.
Par rapport à cette classification, le SIP se focalise sur la gestion du cycle de définition du produit et sur son intégration avec les deux autres cycles (production, support).

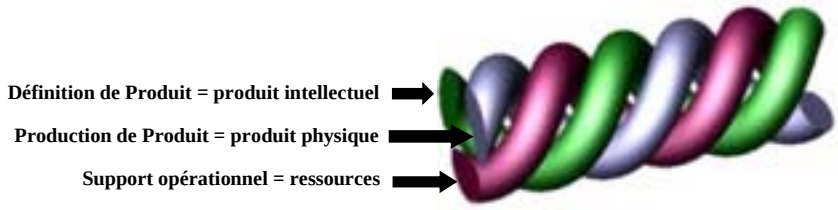


Figure 1.2- Cycle de vie produit (*Source : [Portella, 00]*)

Des exemples de données produit incluent les cahiers de charges, les documents issus des études de marché et des études de faisabilité, le dossier de définition, les plans d'ensembles, les dessins géométriques, les composants standards, les nomenclatures, les notes de calcul, les résultats de simulation, les modèles CAO 3D, les spécifications de fabrication et de contrôle, les plannings de projet produit, les références des programmes de commande numérique, les gammes, les résultats d'essais, les fiches de contrôles, les notices d'utilisation et de maintenance, les lots de rechange, les fiches de réparation, les ordres de modification, etc.
Ces données peuvent avoir différents formats (numérique, graphique, alphanumérique, etc.). Elles sont stockées sur différents supports (papier, bande magnétique, fichier informatique, etc.) et localisées dans plusieurs sites de l'entreprise. Elles sont créées et utilisés par n'importe quelle fonction de l'entreprise (direction stratégie, recherche, études, méthodes, fabrication, essais, soutien logistique, etc.) voire même par des services externes à l'entreprise (sous-traitants, clients, etc.).

2.2.2. Processus SIP

Par processus SIP, nous désignons les processus "métiers" du système d'information produit et ce par opposition aux processus "logiciels" du système informatique associé (SGDT en l'occurrence). Les processus logiciels concourent à la réalisation des processus SIP.
Comme présenté en introduction de la Section 2.2, les processus SIP permettent de réguler la création, la circulation, l'utilisation et l'évolution des informations produit. Ces processus

[16] Quand les activités de support sur le produit cessent (la production étant déjà arrêtée). Le cycle de définition de produit peut durer 30 à 50 ans (exemple : les avions Concorde, les centrales nucléaires, les installations haute voltage, etc.).

reflètent implicitement les fonctions du SIP. A ce propos, plusieurs classifications des fonctions SIP sont présentées dans la littérature. La plupart de ces classifications ne distinguent pas toutefois les fonctions du système d'information de celles du système informatique associé, et ce dû à la confusion souvent faite entre SIP et outil informatique associé tel que SGDT. Les processus métiers se trouvent de ce fait imbriqués avec les processus logiciels. Nous pouvons citer toutefois la classification de Maurino des fonctions de la GDT en trois classes : Gestion de l'encours de conception (acquisition des informations), Gestion des modifications (évolution des informations) et Gestion de la configuration (contrôle contractuel des informations).
En s'inspirant de cette classification, nous proposons de classer les processus SIP en deux processus principaux visant à contrôler la manière avec laquelle les intervenants (utilisateurs) du SIP créent, utilisent et modifient l'information :

1. **Le processus de définition** des informations produit. Ce processus consiste en :

- d'abord, la création de l'information dans le système qui l'a générée et sur son support approprié (cela peut être du papier pour une esquisse ou un croquis, un fichier informatique pour un modèle CAO, une bande magnétique, etc.) et ce selon les procédures organisationnelles en vigueur dans l'entreprise pour créer une information (circuit d'approbations),
- ensuite, l'organisation des informations créées autour d'un modèle fédéré de produit pour faciliter l'accès aux informations. Cela revient à la mise en relation des différentes informations. Il peut inclure la classification des informations comme par exemple la classification des composants dans des familles, la classification des documents selon leurs supports, finalités, types, etc. Il peut s'agir également de la structuration du produit c'est-à-dire la définition des différentes nomenclatures du produit (organiques, géométriques, etc.), de l'affectation des documents aux composants ou aux produits qu'ils documentent, de l'affectation des informations à différentes vues sur le produit pour servir des métiers différents ou pour gérer différentes configurations[17], etc..

2. **Le processus d'évolution** des informations produit. Ce processus vise à assurer la cohérence des informations lors des modifications apportées au produit ou à l'un des éléments qui lui sont associés et ce quels que soient les systèmes et les outils qui l'ont générées et les acteurs qui les manipulent. Il contrôle les changements portés aux informations produit déjà créées et ce suite à un besoin d'évoluer un élément du SIP qui peut aller d'un simple besoin de corriger un document donné jusqu'à un besoin de modifier la définition même du produit. Ce processus consiste généralement en :

- l'instruction de la modification, c'est-à-dire l'analyse de l'impact de la modification souhaitée sur l'ensemble des informations gérées dans le SIP et ensuite la décision sur la faisabilité technique et économique de la modification,
- l'exécution de la modification souhaitée en apportant les changements demandés sur l'élément concerné ainsi que sur l'ensemble des autres éléments du SIP impactés par cette modification.

[17] l'organisation interne des composants d'un produit varie selon le métier utilisant le produit d'où l'existence de différentes nomenclatures organiques qui sont basées sur les mêmes composants mais organisées différemment.

Les deux processus qu'on vient de présenter assurent en outre la gestion de projet associée (définition et attribution des ressources aux tâches, planification, etc.), mais surtout la gestion de l'impact des différentes activités sur les informations concernées. Ils assurent en effet deux fonctions principales : la gestion des travaux et la gestion des flux de travaux.

4 *La gestion des travaux* : par la gestion de tout ce qui arrive aux données lorsqu'un utilisateur les manipule. En effet, le travail de conception conduit les ingénieurs à changer fréquemment la définition des objets en cours de conception afin d'arriver à une solution optimale. Chacun de ces changements entraîne des modifications sur l'ensemble des données associées. Ce travail de conception est souvent exploratoire à la recherche de meilleures solutions et les ingénieurs peuvent retenir une version antérieure. Les processus SIP se doivent de tenir compte du caractère essai-erreur en conservant toute trace de manipulation de données afin de faciliter le retour vers des versions antérieures.

4 La *gestion des flux de travaux* ou de Workflow[18] : le workflow permet de faire circuler et gérer la cohérence entre les lots de travaux[19] relatifs à un processus métier donné d'un acteur ou un groupe d'acteurs vers un autre sous forme de "paquets" organisés selon une logique de découpage du processus qui correspond en général à différents sous-objectifs du processus considéré. Le workflow tel que défini par le WfMC[20] consiste en l'automatisation d'une partie ou de la totalité des processus métiers, où documents, informations ou tâches circulent d'un acteur à l'autre pour agir selon un ensemble de rôles définis et ce pour contribuer à un objectif métier donné. Il peut être organisé manuellement mais dans la pratique la plupart des workflow sont organisés dans le cadre d'un système informatique. Cette vision du workflow est essentiellement procédurale : toutes les activités ainsi que leur enchaînement sont prédéfinies et toute information ne peut être transférée d'un acteur à l'autre que si elle a changé d'état. On s'oriente aujourd'hui vers un concept de workflow plus flexible adapté aux processus d'ingénierie, beaucoup moins structuré où les activités du processus sont définies à fur et à mesure de l'avancement du processus et plusieurs échanges informels entre acteurs sont nécessaires. Par ailleurs, le contexte d'ingénierie est caractérisé par un besoin de vérifier, modifier, soumettre et vérifier continuellement des définitions du produit, souvent supportées par plusieurs documents ou représentations eux-mêmes reliés à plusieurs autres informations. Ce contexte implique une gestion de flux complexes des lots de travaux qui peut être alors facilitée par le workflow. Cette gestion favorise en outre la capitalisation de l'ensemble des décisions prises dans le processus pour déterminer qui fait quoi à chaque étape du processus.

   Ainsi, la gestion des travaux par des workflows permet de *gérer l'historique des travaux*. L'exécution d'un workflow est conservée pour garder une trace de tous les états par lesquels un projet donné est passé, ce qui permettra ultérieurement de pouvoir analyser les causes d'un dysfonctionnement constaté mais aussi de réutiliser des savoir et des savoir-faire déjà acquis dans des projets antérieurs pour développer de nouveaux projets.

[18] L'appellation anglo-saxonne est la plus usuelle.

[19] Un lot de travail est un ensemble de documents, d'informations et de tâches à effectuer pour atteindre un objectif métier donné du processus considéré.

[20] Traduction de la définition proposée par le WfMC [Hollingsworth, 95] qui est une organisation à but non lucratif dont l'objectif est de promouvoir l'utilisation de la technologie workflow.

A chacun de deux processus SIP présentés, correspond un ensemble de processus logiciels qui concourent à les mettre en œuvre dans des SGDT. Nous décrivons ces processus logiciels dans le §2.3 consacré à la présentation des SGDT.

2.3. Les SGDT

Comme nous l'avons précisé, plusieurs systèmes ou outils informatiques concourent à l'informatisation des SIP, afin de faciliter la mise en œuvre des divers processus SIP. Les SGDT sont aujourd'hui les plus complets et les mieux adaptés à la gestion des données techniques. Ils proposent un ensemble de fonctionnalités qui concourent à la mise en œuvre informatique des processus SIP présentés. Nous présentons en détail cette classe d'outils.
Pour les définir, nous nous inspirons de la définition proposée par P. Jagou [Jagou, 93] pour présenter les SGDT comme :
des outils logiciels intégrés permettant de consolider et redistribuer l'ensemble des informations de définition du produit, d'en organiser, gérer et contrôler les accès, les modifications, le partage, le groupement, la sécurité, l'approbation des données techniques, créées sous différents formats et d'assurer l'archivage des données techniques dans un environnement hétérogène et distribué.

2.3.1. Le marché des SGDT

Les SGDT constituent une technologie dynamique et très évolutive. Les premiers SGDT ont fait leur apparition il y a 10-15 ans. Cependant, les améliorations les plus considérables n'ont eu lieu que très récemment. Ces améliorations résultent de l'importance croissante que prend la technologie des SGDT dans les organisations industrielles. Les SGDT constituent désormais pour les entreprises un levier d'action, permettant d'implanter les nouvelles approches managériales du type BPR (*Business Process Reengineering*), ingénierie concourante et certifications qualité. Il permettent par ailleurs de favoriser l'échange de données dans un contexte d'entreprises distribuées, virtuelles et réseaux, tout en se conformant à des normes tels que CALS[21] et STEP[22]. Les investissements des entreprises dans les produits SGDT ont considérablement augmenté de ce fait. En effet, d'après une étude menée par CIMdata [CIMdata, 00], les SGDT représentent le taux de croissance le plus élevé dans le domaine du CIM (*Computer-Integrated Manufacturing*). Cette croissance s'est élevé à 30% annuel durant les 10 dernières années. Avec un marché de 1.76 milliard de U.S.$ en 1999, CIMdata prévoit que ce marché atteigne 2.2 milliard U.S.$ en l'an 2000 et évoluera à un taux de 20% jusqu'à l'an 2004 pour atteindre la somme de 4.4 milliard U.S.$ (cf. Figure 1.3).
Quant aux principales améliorations dans l'offre SGDT, elles concernent principalement :

- l'adaptabilité du progiciel : relative au paramétrage du progiciel lors de son déploiement dans une entreprise tel que l'adaptation à la terminologie interne, aux modèles de produit

[21] CALS (Continuous Acquisition and cycle Life Support) est un programme du Department of Defense (U.S.A) définissant des normes pour les données techniques (création, transmission, utilisation) en vue de faciliter la composition des documents techniques des systèmes d'armes et plus généralement le dossier technique. Pour plus de détail, nous référons le lecteur à consulter l'ouvrage de V. Sandoval [Sandoval, 93].

[22] STEP (STandard for Exchange of Product model data) est une norme internationale de l'ISO (International Organisation for Standardization); ISO 10303 dont l'objectif est de traiter la représentation et l'échange de modèles de produit en couvrant leur cycle de vie. Nous développerons ce standard dans le chapitre II.

et de processus existants et tous les autres éléments nécessaires à une implémentation appropriée du progiciel,
- l'interface utilisateur: entraînée par les environnements PC/Windows et Macintosh, l'utilisation de graphismes et dernièrement par la technologie du WEB,
- les fonctionnalités proposées : liées à l'évolution de la vision de la GDT,
- l'architecture : en utilisant davantage des outils standards (tels que les "toolkits" pour interfaces graphiques),
- les plates-formes.

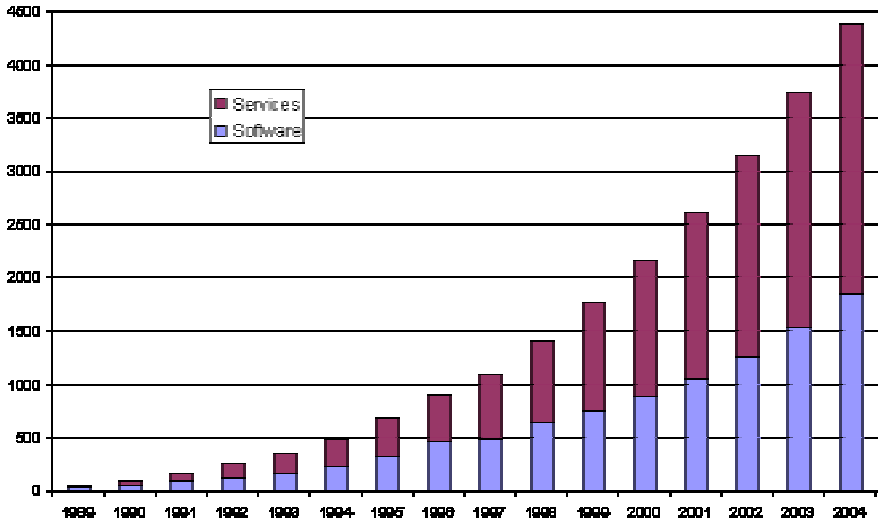


Figure 1.3 - Evolution du marché des SGDT (*Source : CIMdata [CIMdata, 00]*)

Concernant la cible des produits SGDT, les premiers logiciels étaient destinés aux industries à process discret, principalement l'aéronautique, l'automobile et les industries électroniques. Par la suite, les SGDT furent utilisés par les industries à process continu, notamment dans les industries pétrolières, pharmaceutiques et énergétiques. Leur utilisation devrait s'étendre à d'autres secteurs industriels, tels que la gestion des équipements dans les établissements hospitaliers et la gestion des activités et des documents dans les établissements judiciaires [CIMdata, 00].
Toutefois, l'ensemble de l'offre progicielle reste destinée plutôt aux grandes entreprises. Les fonctions offertes par les SGDT actuels sont proposées sur des plates-formes qui ne sont pas à la portée des petites entreprises. Une prise en compte des besoins effectifs des petites entreprises dans l'offre progicielle actuelle est souhaitable[23].

Quant aux produits proposés sur le marché, diverses générations de SGDT sont distinguées, essentiellement selon le type de SGBD utilisé par ces produits. Les plus récentes sont basées sur des SGBD Relationnels avec une couche objet tels que Windchill de PTC[24] ou encore sur des SGBD Objets tels que Adra de Matrix[25].

[23] Nous citons dans ce cadre, le travail de thèse de P. Pernelle mené au laboratoire LLP-CESALP de l'université de Savoie, visant à définir une architecture de SGDT pour les PME [Pernelle, 00].
[24] Site web : www.ptc.com
[25] Site web : www.matrix-one.com

Selon une étude récente de CIMdata, le produit pilote sur le marché est Metaphase de SDRC avec une part de marché de 8%, suivi par Windchill de PTC (7%) et Enovia d'IBM[26] (6%). La Figure 1.4 présente un classement des dix produits les plus vendus.

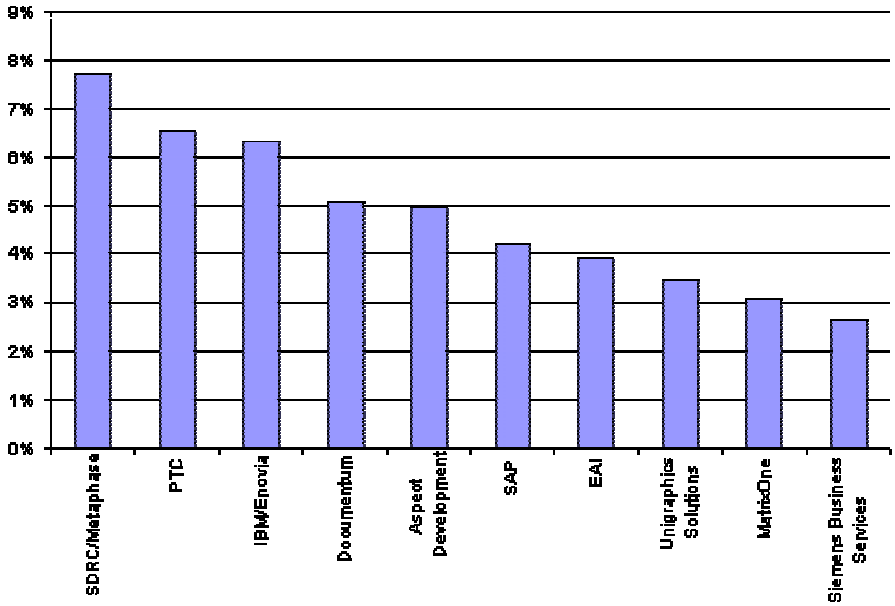


Figure 1.4 - Part de marché des 10 produits les plus vendus (*Source : CIMdata [CIMdata, 00]*)

2.3.2. Fonctionnalités du SGDT

Diverses descriptions des fonctionnalités existantes ou qui devraient exister dans les SGDT sont présentées dans la littérature. Chacune de ces descriptions fait émerger un certain nombre de fonctionnalités. Nous récapitulons dans ce qui suit les principales :

- *Structurer les données* : malgré la capacité des entreprises à acquérir ou enregistrer systématiquement les informations produit (sur différents supports), les informations pertinentes détenues par certains attributs ne sont pas souvent mises en avant. Ainsi, pour un composant donné par exemple, il est difficile de retrouver rapidement ses cas d'emploi ou encore les différents documents qui lui sont associés. Les intervenants sur le produit ont souvent des difficultés pour accéder à l'information dont ils ont besoin ce qui réduit l'efficacité de la gestion du produit. Une description des données techniques par les attributs (d'objets et de liens) et une structuration de ces attributs offre plusieurs possibilités de tri et de recherche sur les données.
- *Classer les données* : avec les quantités de données générées, une technique pour classifier facilement et rapidement ces données est nécessaire. Grâce à la description des données par les attributs, plusieurs possibilités de classification sont offertes. La classification concerne essentiellement les composants.
- *Visualiser et stocker les données* : certaines informations produit sont définies et archivées dans le SGDT grâce aux classes et attributs associés mais d'autres informations documentant le produit tels que les fichiers des plans CAO, les fichiers de photos

[26] Site web : http://www.enovia.com/

numérisés, etc. ne sont que référencés. Ceux-ci doivent être alors visualisés dans le SGDT. Pour certains formats de fichiers, ceci est possible par des "moteurs" de visualisation[27] (*Viewer*) disponibles dans les SGDT. Pour d'autres formats de fichiers, ils ne peuvent être visualisés que sur les moteurs qui les ont générés, éventuellement déportés ou importés dans le SGDT en temporaire. Toutefois, certains fichiers dits "Objets longs" ne peuvent être interprétés, ils sont donc gérés comme un tout dans une unité informatique qui peut être propre ou non au SGDT. Le SGDT doit alors assurer l'import - export de ces objets [Randoing, 95].

- *Structurer les données entre elles*, c'est-à-dire définir les liens entre les différents objets définis (classes essentiellement). Cela permet entre autres de structurer le produit en terme de nomenclatures. Ces dernières définissent la décomposition du produit selon différentes vues du produit pour servir divers acteurs de l'entreprise (nomenclature fonctionnelle, nomenclature organique d'études, nomenclature organique de fabrication, nomenclature géométrique, etc.).
- *Gérer l'évolution* des données : un produit est amené à évoluer durant son cycle de vie et les données qui lui sont rattachées sont alors évolutives. Pour garder trace des diverses modifications apportées aux informations produit, différents indices d'évolution sont affectés à ces informations pour distinguer les diverses versions ou révisions. Par ailleurs, pour changer d'indice d'évolution, une information produit suit un processus de modification. Selon l'avancement de ce processus, l'information modifiée acquit divers statuts (créée, soumise, libérée, validée, etc.).
- *Protéger les données*, par un contrôle des modifications et des accès. Différentes autorisations sont définies pour les acteurs ou les groupes d'acteurs. Une autorisation est un droit pour exécuter certaines fonctions (consulter, créer, modifier) sur les données selon le statut de ces dernières.
- *Distribuer les données*, sur tous les sites, à toutes les fonctions. Les bases de données réparties ne sont pas très courantes, surtout sur des sites distants et hétérogènes. Le transfert de données créées ou modifiées est parfois automatisé grâce à une notion "d'abonnement"[28] proposée dans certains SGDT.
- *Discipliner la création et l'évolution des données* : une création ou une modification d'une donnée est généralement référencée par un n° de dossier (ECO[29]). Dans le cas d'une modification, celle-ci est initialisée par un ECR[30] (qui a le même n° que l'ECO en général). La création ou la modification doit être validée par un responsable de l'ECO. Les modifications sont alors introduites dans le modèle de données, complété éventuellement par une recopie sur les différents sites et le dossier est fermé (l'ECO peut être alors consulté). Le suivi des différentes étapes du processus (instruction, validation, etc.) est assuré par Workflow jusqu'à la validation définitive.

[27] Ce sont des formats d'interprétation pour la représentation alphanumérique ou graphique qui permettent d'archiver la donnée sous son format d'origine.

[28] Seules les données qui concernent la liste des produits auxquels chaque site est "abonné" sont transmises [Randoing, 95].

[29] Engineering Change Order.

[30] Engineering Change Request.

- *Structurer l'instruction d'un dossier* et sa chronologie *(Workflow) :* le workflow consiste en une représentation graphique d'entités et de relations, de type objet. A chaque entité est attachée une classe et des attributs : responsables, autorisés, durée de la tâche, début et fin au plus tôt et au plus tard, description et quantification des ressources, personnes à prévenir, dispatching, validation, etc. Cette gestion des procédures porte sur la nature des tâches, leur enchaînement et la surveillance des délais. Elle nécessite un courrier électronique. Toutefois, il n y a pas d'ordonnancement des ressources ou de calcul des chemins critiques, ni de lien dynamique avec les données manipulés dans ces procédures. Le fait par exemple de décider de changer de version d'un document à un moment donné du workflow, ne met pas à jour automatiquement l'objet concerné et ne propage pas l'impact de ce versionnement sur le reste des données.

2.4. Synthèse

La gestion de données techniques est un domaine relativement récent et en plein mouvement. Cette mouvance résulte de l'intérêt croissant des organisations industrielles pour ce mode de gestion ce qui a favorisé l'enrichissement de l'offre progicielle associée sur le marché. En effet, dans un contexte d'organisations industrielles plus complexes conduisant les entreprises à adopter de nouvelles pratiques comme par exemple l'ingénierie simultanée voire de nouvelles structures comme l'entreprise virtuelle, les SIP deviennent un enjeu stratégique pour les entreprises. Ils permettent une gestion performante de la base documentaire accompagnant le développement des produits et ensuite une rationalisation de l'ensemble des tâches du processus de développement. La réactivité des entreprises aux modifications inhérentes au processus de développement des produits nécessite un pilotage de la circulation de la documentation technique, une gestion rigoureuse de son évolution et s'accompagne d'une exigence liée à la maîtrise de la qualité, autant de fonctions qu'assure le SIP. Malgré l'importance des SIP dans l'entreprise, l'ingénierie de ces systèmes, c'est-à-dire les aspects méthodologiques liés à leur conception et à leur réalisation, demeure un processus de support[31] pour l'entreprise dont il convient de minimiser les coûts. Les enjeux associés à ces aspects méthodologiques sont alors stratégiques. La thématique de recherche de ce mémoire de thèse concerne les aspects méthodologiques dans l'ingénierie des SIP. Dans la partie suivante du présent chapitre (§3), nous nous focalisons sur les approches d'ingénierie des SIP.

3. Ingénierie des Systèmes d'Information Produit

L'ingénierie des systèmes d'une façon générale consiste, comme le définit B.S. Blanchard [Blanchard, 98] en "l'application d'efforts scientifiques et d'ingénierie pour transformer des besoins opérationnels en une configuration système définie à travers un processus top-down itératif, composé de : l'analyse de besoins, l'analyse fonctionnelle et l'allocation, la synthèse, la conception optimisée, les tests et évaluations et la validation".

[31] Dans [Kettani, 98], les processus d'une organisation sont classés selon trois axes : les processus métiers (ceux visibles de l'extérieur de l'organisation et qui sont présentés aux clients, tels que la Gestion du marché), les processus support (ceux qui fournissent le support nécessaire aux processus métier pour s'exécuter - ils ne sont pas visibles de l'extérieur, tels que le Développement de systèmes informatiques) et les processus de gestion (ceux qui fournissent les moyens de gestion des processus métiers - ils ne sont pas visibles de l'extérieur, tels que le Pilotage).

Par *système*, est entendu dire par le même auteur "un ensemble de composants inter-reliés et qui collaborent ensemble pour le même objectif de satisfaire un besoin précis". Ces composants peuvent être sous la forme de personnes, de matériels, d'équipements, de logiciels, de moyens, de données, d'argent, etc. Dans le cas d'un système d'information, et en s'inspirant de la définition de D. Avison [Avison, 97], cet ensemble de composants a pour objectif de fournir l'information sur l'organisation et son environnement.
L'ingénierie des Systèmes d'information quant à elle est définie par D. Avison [Avison, 97] comme le processus par lequel les analystes du système, les ingénieurs informaticiens, les programmeurs et les utilisateurs finaux construisent les systèmes d'information. Elle a pour but comme présenté par A. Front [Front, 97] de transformer les besoins d'une application en une solution logicielle fiable dans un système informatique cible choisi.

Qu'en est il alors pour *l'ingénierie des systèmes d'information produit* ?
Dans la littérature, on ne parle pas vraiment de "L'ingénierie des systèmes d'information produit". Un SIP est considéré comme un système d'information utilisé pour une application particulière, celle de la gestion des données techniques du produit. L'ingénierie des SIP relève alors de l'ingénierie des systèmes d'information en général. C'est pourquoi la première section de la présente partie, consacrée aux aspects théoriques de l'ingénierie des SIP sera dédiée à l'ingénierie des SI en général (cf. §3.1). L'ingénierie des SIP sera ensuite approchée d'une façon plus spécifique au travers de la pratique industrielle de cette activité. C'est l'objet de la deuxième section de cette partie (cf. §3.2).
Dans chacune des deux sections, les problèmes rencontrés dans le domaine (ingénierie des SI et des SIP) sont soulignés à fur et à mesure de la description. Certains de ces problèmes ne sont pas spécifiques aux SIP mais communs à tous les systèmes d'information, d'autres sont plus spécifiques au domaine des SIP. Pour pouvoir préciser la problématique que nous traitons dans la thèse, un état de l'art sur les travaux menés dans le domaine pour résoudre les problèmes soulignés accompagne la description de l'ingénierie des SI et celle des SIP.

3.1. Ingénierie des Systèmes d'Information

L'*ingénierie de SI* peut être présentée comme un processus par lequel les besoins du SI sont transformés en une solution logicielle fiable. L'ingénierie des SI est également connue sous le nom de processus de développement de SI.

C. Rolland [Rolland, 93] présente l'ingénierie des SI sous la forme de deux processus complémentaires (cf. Figure 1.5) :
- *L'ingénierie des besoins* qui consiste à acquérir et représenter les connaissances dont les utilisateurs ont besoin. Ceci se traduit par l'élaboration d'un Schéma Conceptuel du SI.
- *L'ingénierie des systèmes* qui consiste à implanter le schéma conceptuel du SI sur un système technologique en utilisant les technologies d'information. Ce système facilite la gestion des connaissances dont les utilisateurs ont besoin.

C'est le schéma conceptuel du SI qui articule la transformation des besoins en une solution logicielle.

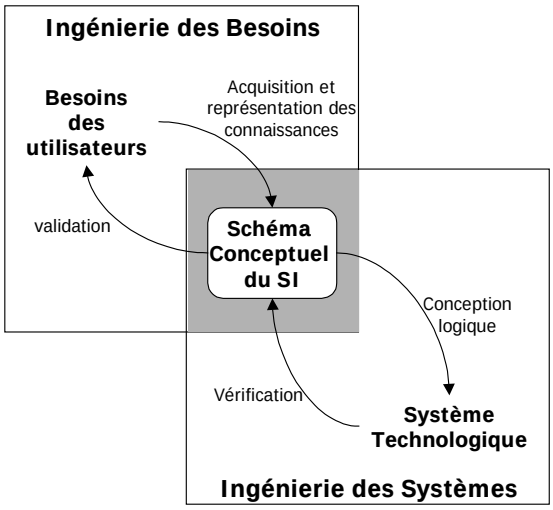


Figure 1.5 - Ingénierie d'un système d'information (*Source : [Rolland, 93]*)

Le processus de développement d'un SI comprend différentes étapes ou activités que l'on peut résumer en : l'*analyse*, la *conception* et l'*implantation* (cf. Figure 1.6). La phase d'analyse part d'un problème du monde réel et dresse une représentation du domaine du problème. Cette représentation est le point de départ de la phase de conception qui aboutit à une représentation du domaine de la solution. Enfin, la phase d'implantation traduit le domaine de la solution dans un système technologique [Front, 97].

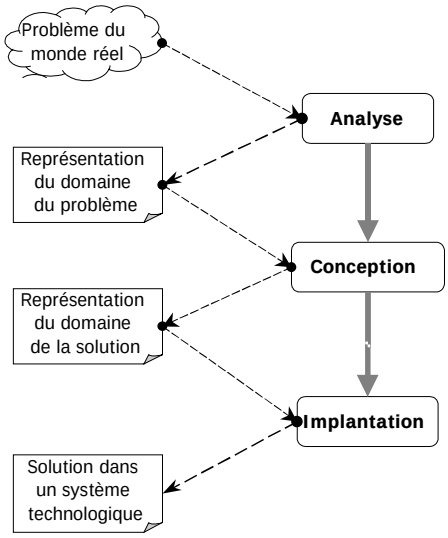


Figure 1.6 - Processus de développement des SI

3.1.1. L'ingénierie des SI comme une ingénierie de produits

Initialement, l'ingénierie des SI se déroulait sans aucune méthodologie explicite [Avison, 97]. L'accent était mis sur la programmation et peu d'intérêt était porté à la bonne formulation des besoins utilisateurs. Les SI conçus étaient par conséquence dans la plupart des cas inadaptés aux besoins et les programmeurs passaient la majorité de leur temps à corriger et adapter l'application déjà en milieu opérationnel. Par ailleurs, la gestion du projet de développement était rendue difficile à cause de la difficulté d'estimation des délais de développement (livraison, tests, etc.) qui étaient souvent trop grands. Cette situation est devenue critique avec

l'expansion de l'utilisation des systèmes informatisés. Un besoin pour des méthodologies organisant le développement des SI s'est fait ressentir.
L'ingénierie des SI se fait désormais selon des méthodes. Plusieurs définitions de la notion de méthode ont été proposées [Avison, 97], [Rumbaugh, 95], [Tessier, 95] et la plupart d'entre elles convergent vers l'idée qu'une méthode est *une manière de faire suivant certains principes et toujours avec un certain ordre*.
C. Rolland [Rolland, 88] structure la méthode en définissant quatre composants indissociables et complémentaires de cette notion:

- des modèles : un modèle est une ensemble de concepts et de règles pour les utiliser, destiné soit à expliquer et construire la représentation des phénomènes organisationnels, soit à expliquer et représenter les éléments qui composent le SI et leurs relations. Une méthode constitue ainsi un modèle d'emploi des modèles;
- des langages : un langage est un ensemble de constructions qui permettent de décrire formellement les spécifications du SI élaborées aux différents stades du processus de conception en s'appuyant éventuellement sur les modèles de la méthode;
- une démarche : la démarche est le processus opératoire grâce auquel s'effectue le travail de modélisation, de description, de réalisation du SI;
- des outils ou techniques : ils peuvent concerner les trois composantes ci-dessus, et ont pour objectif d'aider leur mise en œuvre.

D. Rieu [Rieu, 99c] définit par ailleurs une méthode d'analyse et de conception de SI comme :

- un ensemble de modèles (formalismes) permettant de représenter les systèmes sous différents aspects (statique, dynamique, fonctionnel) et à différents niveaux d'abstraction (expression des besoins, analyse, conception). Ces modèles sont en fait des modèles de produits[32], c'est à dire des formalismes permettant de décrire le SI à développer (par exemple le modèle entité/Association d'un système d'information bancaire).
- une (des) démarche(s) ou processus permettant de guider les activités de développement. C'est une succession d'étapes permettant en particulier de passer d'une modélisation du système à une autre. Ces démarches sont souvent soit trop imprécises (termes floues) soit trop rigides (constituées d'une longue succession d'étapes et donc inadaptées aux petites applications).

De nombreuses méthodes[33] ont été mises au point depuis plusieurs années et il est très difficile d'être exhaustif lorsqu'il s'agit de recenser les méthodes existantes. A ce sujet, J. Martin [Martin, 85] parle de "jungle des méthodes". Le nombre d'ouvrages qui traitent des méthodes est donc considérable. Nous présentons succinctement les principales caractéristiques des méthodes proposées en s'inspirant de la typologie présentée par C. Tessier [Tessier, 95].
En se basant sur le type d'approche, quatre classes de méthodes sont distinguées:

---

[32] Par produit est désigné le produit de l'activité d'ingénierie, c'est le SI à développer. En effet, il est usuel de distinguer la notion de "produit" de celle de "processus" dans de nombreux domaines d'ingénierie et notamment l'ingénierie des SI. T.W. Olle [Olle, 91] décrit le produit comme "le résultat à atteindre" et le processus comme "le chemin qu'il faut parcourir pour atteindre le résultat".

[33] Notons que certains auteurs utilisent le terme de méthodologie à la place de méthode. C. Tessier [Tessier, 95] précise qu'une *méthodologie* se rapporte à l'étude des méthodes et qu'une *méthode* désigne à l'origine un ensemble de démarches qui suit l'esprit pour découvrir et démontrer la vérité dans les sciences.

- *les méthodes d'analyse* qui datent du début des années 1960. Elles sont nées de la préoccupation des informaticiens à standardiser le métier de développeur d'application et à offrir une approche industrielle d'informatisation. Ces méthodes se sont particulièrement intéressées à la réalisation des systèmes et ensuite à leur conception. Elles adoptent deux approches principales[34] :
  - une *approche analytique,* ou par les données, qui recense l'information à fournir en sortie du système et remonte aux entrées comme la méthode MINOS,
  - une *approche synthétique,* ou par les fonctions, qui est axée sur les traitements et recherche les informations d'entrée qui sont nécessaires pour en déduire les résultats comme la méthode CORIG.
- *les méthodes cartésiennes* des années 1970 comme SADT (Structured Analysis and Design Technique) ou JSD (Jackson System Development). Ces méthodes se caractérisent par une approche fonctionnelle qui met en œuvre des concepts et des techniques de décomposition hiérarchique, s'appliquant sur les processus et les flux de données (ou flux d'information) et préconise une analyse et une conception du SI à partir de la définition de fonctions. Le processus d'analyse et de conception débute par l'identification d'une fonction globale de gestion. Le traitement de l'information répond aux règles de procédures de gestion pour produire des sorties. La démarche de travail est descendante, autrement appelée "top-down". Toute fonction est décomposée en sous-fonctions, et ainsi de suite, par raffinements successifs, jusqu'à ce que les ensembles élémentaires soient intelligibles. Cela donne une représentation en arbre : les fonctions globalement perçues sont éclatées en processus spécifiques. Les relations entre ces derniers sont également recherchées. Ces méthodes ont été influencées par la programmation modulaire.
- *les méthodes systémiques* des années 1980 comme MERISE (Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise), AXIAL (Analyse et Conception de Systèmes d'Informatisation Assistées par Logiciels), REMORA, IA-NIAM (Nijesseb's Informations Analysis Method), IDA (Interactive Design Approach) : ces méthodes préconisent une approche conceptuelle globale du SI. Elles adoptent un processus de modélisation par niveaux d'abstraction successifs. Historiquement, les premières propositions ont essentiellement concernés la modélisation des données. Ainsi se sont développées plusieurs techniques de modélisation : relationnelle, entité-association, binaire, sémantique. A leur suite, des formalismes pour les traitements et les communications ont été élaborés. Les méthodes systémiques proposent soit des modèles bien séparés (étude statique puis dynamique), soit des modèles qui synthétisent les deux aspects. A ce sujet, certaines techniques de modélisation associent la représentation de la structure du système d'information et celle de son comportement.
- *les méthodes objet* des années 1990 : ces méthodes sont une conséquence de l'importance croissante des langages de programmation orientés objet de type C++ ou SmallTalk. Les méthodes objet permettent des spécifications détaillées des éléments d'un SI en introduisant la notion d'objet regroupant structures de données et traitements [Chabre, 97]. Elles proposent des spécifications globales d'un SI par l'intermédiaire de spécifications statiques et dynamiques. Certaines méthodes proposées sont réellement nées des concepts orientés objet comme OOA (Object Oriented Analysis), OOD (Object Oriented Design), OMT (Object Modeling Technique) et OOSE (Object Oriented Software Engineering)

[34] cf. [Tessier, 95] p. 82.

[Jacobson, 92], d'autres méthodes ont été étendues pour prendre en compte ces concepts comme MERISE/2 et OOM (Orientation Objet dans Merise). Les méthodes objet étant de plus en plus nombreuses (plus de soixante-dix méthodes[35]), des tentatives d'unification ont été effectuées. En particulier, le langage UML (Unified Modeling Language) né de l'unification des méthodes objet initiée par G. Booch (méthode Booch), J. Rumbaugh (méthode OMT) et I. Jacobson (méthode OOSE). En réponse à l'appel d'offre de l'OMG (Object Management Group) sur la standardisation des langages de modélisation objet, UML1.0 [OMG, 97] a été adopté comme standard de modélisation objet en novembre 1997. Notons qu'un consensus est établi sur le fait qu'UML n'est pas une méthode mais un langage de modélisation. Un langage dans le sens où il propose des concepts, une syntaxe et une sémantique avec une notation supplémentaire sous forme graphique. UML n'est pas une méthode dans le sens où il ne propose pas un processus décrivant la démarche de développement.

3.1.2. De l'ingénierie des produits à l'ingénierie des processus

Comme le signale C. Rolland [Rolland, 96], les méthodes proposées dans le passé mettent l'accent sur la modélisation des produits de l'ingénierie des SI. Elles offrent un ensemble de modèles pour construire les produits alors que les démarches ou les processus qui assurent la construction de ces produits sont négligés. Or, la maîtrise, la rationalisation et donc l'amélioration du développement des SI ne peut se faire que grâce à une représentation explicite des démarches de développement. On parle alors de modèle de processus. L'intérêt porté à l'ingénierie des produits (c'est-à-dire la description, modélisation des produits) s'est déplacé ainsi vers l'ingénierie des processus (description, modélisation des processus). Une méthode doit être alors constituée de modèles de produits mais également de modèles de processus. La modélisation des processus permet par ailleurs d'expliciter les liens de cohérence ou de déduction entre les modèles de produits, au moment du passage d'une modélisation de produit à une autre. Elle est également une source pour évoluer et améliorer les processus et par la suite les méthodes [Rolland, 98a].
C Rolland [Rolland, 96] classe les approches de modélisation des processus en deux groupes apparus chronologiquement : les démarches génériques qui offrent des processus figés et les modèles de processus qui offrent des formalismes permettant de modéliser les processus.

- *Les démarches génériques* **:** jusqu'au milieu des années 80, les approches de modélisation visent à proposer des modélisations du cycle de développement d'un SI sous la forme d'un ensemble organisé d'étapes. Plusieurs modèles on été proposés : le modèle en cascade de W.W. Royce, le modèle en V et le modèle en spirale de B.W. Boehm, et plus récemment le modèle en fontaine de B. Henderson. Toutefois, le niveau de granularité faible dans la spécification des processus de développement et la pauvreté d'expression des enchaînements des tâches, rendent ces démarches inadaptées à la modélisation fine des processus [Rolland, 96].
- *Les modèles de processus* : les modèles de processus sont proposés pour palier aux faiblesses des démarches génériques en offrant un pouvoir d'expression plus grand. En plus de l'expression de l'enchaînement des activités composant les processus de développement et des éléments du produit qu'elles manipulent, les modèles de processus

[35] Cette donnée est prise de [Kettani, 98] p.14.

sont plus dédiés à la phase d'analyse où la part de raisonnement humain est prépondérante. L'exécution des activités de telle phase ne peut être simplement l'application d'un plan préétabli plus ou moins immuable ; elle est surtout la conséquence de décisions humaines. Plusieurs modèles ont été proposés[36] dont les plus récents sont les modèles processus orientés-décision, où le processus de développement est vu comme un processus de prise de décision. Ces modèles, en plus de la spécification des enchaînements d'activités et de la prise en compte du produit résultant des activités, spécifient les intentions motivant l'exécution des activités et leur enchaînement. Ils aident à la prise de décision en raisonnant sur la justification des décisions et en offrant un support au processus de délibération lié à celui de prise de décision [Rolland, 96]. Certains modèles proposés comme PSD [Desclaux, 90] sont descriptifs (centrés sur la trace des processus) et mettent l'accent sur la prise de décision et ses raisons. D'autres modèles comme ceux proposés dans le projet NATURE [Grosz, 97] sont prescriptifs et basés sur la paradigme contextuel. Ils partent du principe que le concepteur réagit souvent par analogie avec les situations dans lesquelles il s'est déjà trouvé impliqué. Il s'agit donc de modéliser l'ensemble des décisions qui peuvent être prises par le concepteur en fonction des situations dans lesquelles il peut se trouver. On parle alors d'approches contextuelles.

3.1.3. De l'ingénierie des processus à l'ingénierie des méthodes

L'ingénierie des processus permet une meilleure maîtrise des processus et donc garantit une plus grande cohérence des modèles produit (lors du passage d'un modèle à l'autre). Elle résout également les problèmes d'imprécision des démarches proposées (souvent exprimées dans des termes flous).

Toutefois, d'autres problèmes subsistent ou sont apparus par la suite. D'abord, la prolifération des méthodes d'ingénierie durant les dernières années a rendu difficile pour une organisation donnée le choix d'une méthode appropriée. Ensuite, comme le souligne D. Rieu [Rieu, 99c], la rigidité de certaines méthodes oblige les organisations à adapter la méthode en fonction de la taille et la nature des applications. Une adaptation qui est souvent difficile. Il s'agit donc pour chaque organisation de choisir une méthode puis de l'adapter en permanence en fonction des applications. Cette adaptation est due essentiellement au fait que les méthodes "traditionnelles" utilisées dans les SI ont été définies de manière "ad-hoc" comme le note C. Rolland [Rolland, 96]. Les modèles de processus sous-jacents sont en effet l'expression de l'expérience des développeurs. Il en résulte qu'il n'est pas possible de savoir comment les modèles de processus ont été générés (dans quelles conditions). Ils restent dépendants du domaine d'expérience et sont donc difficilement adaptables. Il serait intéressant pour une organisation de pouvoir définir "sa méthode". Il fallait alors avoir des modèles de processus indépendants des domaines et pouvoir construire et modifier rapidement ces modèles. La communauté de l'ingénierie des SI s'est depuis peu penchée sur ces problèmes et on a vu l'apparition d'une nouvelle approche; celle de *l'ingénierie des méthodes* qui a pour objectif de donner une manière pour construire, améliorer et faire évoluer les modèles de processus [Rolland, 96]. L'objectif est de permettre la construction de nouvelle méthode à partir de fragments de méthodes existantes. Deux approches prédominent actuellement en ingénierie des méthodes :

[36] Une présentation des différents modèles proposés dans la littérature est faite par C. Rolland [Rolland, 96].

- *la méta-modélisation* : en méta-modélisant[37] les produits et les processus (c'est-à-dire modéliser les modèles de produits et les modèles de processus), on peut obtenir plusieurs modèles de produits et de processus par instanciation du méta-modèle. De nombreux cadres de référence ont été proposés pour faciliter l'adaptation des méthodes. Nous citons en particulier le cadre de référence du groupe de travail FRISCO de l'IFIP, WG8.1[38] et le cadre de référence d'UML avec le MOF (Meta Object Facility) comme méta-modèle [OMG, 99a].
- *l'approche situationnelle* : cette approche est née du fait qu'une méthode n'est jamais suivie à la lettre mais au contraire adaptée à la situation de chaque projet. L'approche situationnelle recommande la construction d'une nouvelle méthode par assemblage en fonction de la spécificité du projet à développer et de la disponibilité de fragments de modèles de processus émis de différentes méthodes. Une méthode situationnelle est alors construite en fonction de la situation particulière d'un projet par agencement de fragments de méthodes, prédéfinis, validés et accessibles dans une base de méthodes. Dans ce cadre, nous citons les travaux de C. Rolland [Rolland, 96] proposant une approche de construction modulaire de méthodes, en utilisant l'approche contextuelle de modélisation des processus du projet NATURE [Grosz, 97] comme technique de modélisation des composants de la base des méthodes stockant différents fragments de méthodes. Nous citons également le projet EUROMETHOD qui fournit un ensemble de critères d'évaluation de la situation spécifique d'un projet, un ensemble de stratégies pour planifier les activités et un ensemble de règles méthodologiques pour choisir et agencer les stratégies les mieux adaptées au projet.

Synthèse

En résumé, *l'ingénierie des SI* est passée d'une *ingénierie de produits* où l'accent est mis sur la modélisation des produits vers une *ingénierie des processus* où l'accent est mis sur la modélisation des processus (c'est-à-dire comment modéliser les produits) et ensuite vers une ingénierie des méthodes où l'accent est mis sur la modélisation des méthodes (c'est-à-dire comment modéliser les processus).

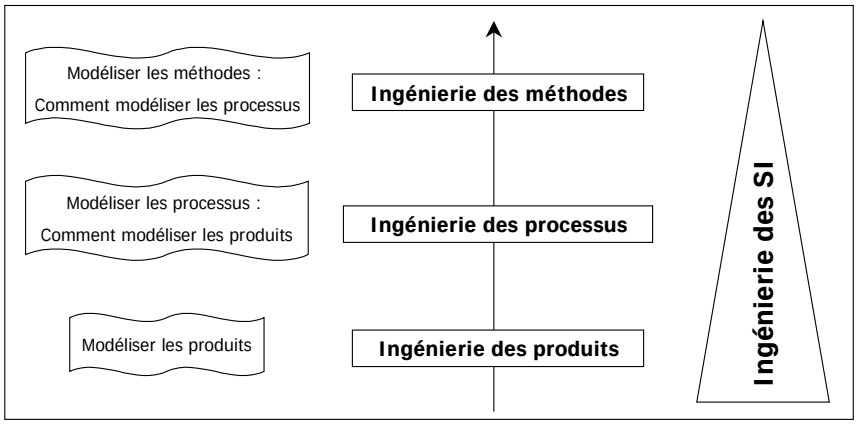


Figure 1.7 - De l'ingénierie des produits à l'ingénierie des méthodes

[37] Un métamodèle est un langage formel ou semi-formel permettant de modéliser des modèles [Rieu, 97].

[38] Site web : http://www.ifip.or.at/

Nous avons fini à présent la description des aspects théoriques liés à l'ingénierie des SIP, à travers l'étude de l'ingénierie des SI en général. Nous approchons dans ce qui suit d'une façon spécifique l'ingénierie des SIP au travers de la description de la pratique industrielle de cette activité et de la revue des travaux portant sur ce thème.

3.2. Ingénierie des SIP

3.2.1. Pratique industrielle de l'ingénierie des SIP

La pratique industrielle de l'ingénierie des SIP que nous décrivons ici est celle relative à notre partenaire industriel, que nous avons déduite à partir de l'observation du projet VEGA1 Mécanique. Ce projet concerne la mise en place d'une application du système d'information produit de l'activité Appareillage Basse Tension (ABT) du Domaine d'Activité Stratégique (DAS) Basse Tension Puissance (BTP). Cette application a pour objectif le suivi des demandes de modification d'un produit industrialisé pour le métier de la mécanique.
La conduite d'un projet SIP se déroule selon le processus général suivant :

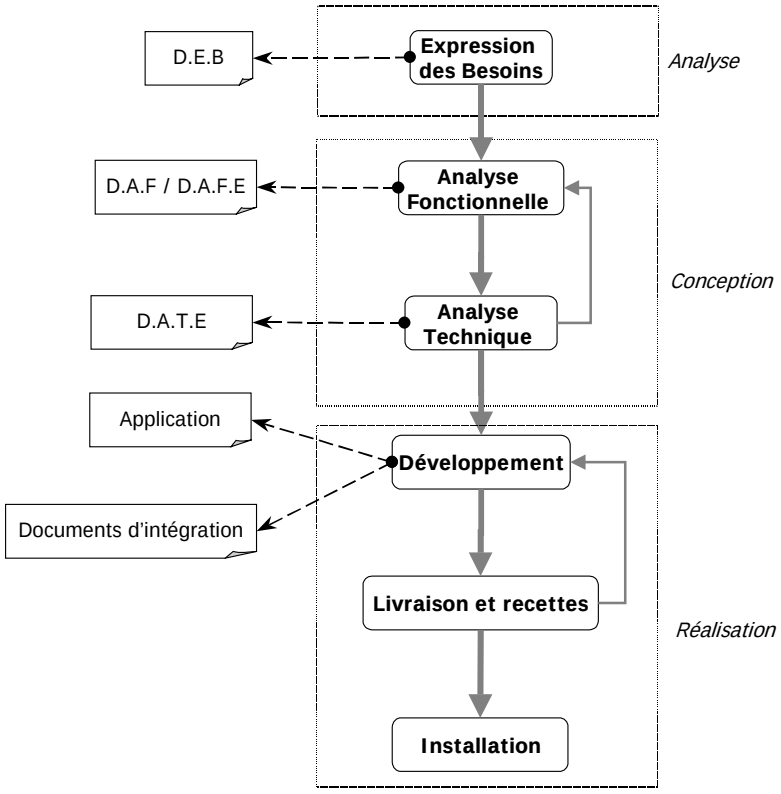


Figure 1.8 - Processus de développement des SIP à Schneider

- **L'expression des besoins** : c'est une phase importante devant conduire à l'élaboration d'un dossier contractuel, dit Dossier d'Expression des Besoins (D.E.B), entre, d'une part le Chef de Projet Utilisateur CPU (représentant des utilisateurs du SIP dans les bureaux d'étude, méthodes industrielles, services qualité, fabrication, etc.) et d'autre part le Chef de Projet Système d'Information CPSI (pilotant la spécification et l'implantation du SIP). Ce dossier précise les contraintes et les objectifs du système d'information cible. Il décrit les besoins des utilisateurs et précise les fonctions à réaliser. Pour chaque fonction, il est précisé le

responsable de cette fonction, l'ensemble des intervenants et les informations "en vrac" à renseigner et à déduire. De ce que précise le client, le CPSI construit le D.E.B. Ce D.E.B est décrit sous forme textuelle, avec parfois des images d'écrans.
Cette première étape est rendue délicate par le manque de modèles "formels" qui soient compréhensibles par les utilisateurs ainsi que de démarches adaptées pour élaborer des spécifications parlantes et non ambiguës des besoins. De ce fait, plusieurs itérations sont nécessaires pour préciser les besoins, allongeant ainsi les délais. Par ailleurs, ce mode de travail peut induire une expression redondante de certains besoins ou à l'inverse un oubli de certaines fonctionnalités à préciser.
Finalement, il est important de noter que l'expression des besoins est souvent davantage approchée en terme de solutions techniques qu'en terme de problèmes à résoudre.

- **L'analyse fonctionnelle** : En s'appuyant sur l'expression des besoins, elle définit les spécifications dites fonctionnelles du SIP qui seront prises en charge ultérieurement par les Ingénieurs d'Application (IA) chargés de leur implantation. Cette phase se fait par écart[39] à l'outil d'implantation, METAPHASE[40]. On y décrit le Dossier d'Analyse Fonctionnelle par Ecart (D.A.F.E) dans lequel le CPSI définit les objets du domaine, le cycle de vie des objets, les traitements associés et l'interface utilisateur.
  Actuellement, cette spécification est faite sans réelle continuité par rapport à l'expression des besoins. Le chef de projet SIP établit généralement deux dossiers distincts : l'un destiné aux utilisateurs et décrit dans des termes métiers proches de leur langage, l'autre pour l'équipe de développement et décrit dans des termes informatiques. Ceci rend la cohérence entre ce qui attendu (ce qui est arrêté avec les utilisateurs) et ce qui délivrable (ce qui est arrêté avec les développeurs) purement intentionnelle. Par ailleurs, le D.A.F.E est rédigé sans outil de modélisation, plutôt sous forme de texte, de tableaux et d'images d'écran.

- **L'analyse technique** : en s'appuyant sur le D.A.F.E, l'Ingénieur d'Application élabore des spécifications dites techniques qui décrivent d'une façon précise l'implantation en METAPHASE, dans un Dossier d'Analyse Technique par Ecart (D.A.T.E). Ce dossier, après validation par le CPSI, est fourni aux développeurs chargés du codage. Cette spécification est également faite par écart à METAPHASE. On y indique les classes et les méthodes de METAPHASE à récupérer et les ajustements à apporter. Dans certains cas, cette phase est précédée par une phase préalable d'analyse fonctionnelle détaillée où on spécifie davantage les interfaces et les dialogues. Durant cette phase préalable, l'Ingénieur d'Application (IA) est souvent amené à échanger avec le Chef de Projet Système d'Information (CPSI) pour préciser les traitements, notamment les cas dits "dégradés", c'est à dire les cas en dehors du fonctionnement "nominal" de l'application (les traitements sont souvent spécifiés sous la forme de séquences de "Si .. Alors .." mais il est rare qu'on spécifie les "Sinon .." dans ces spécifications). Parfois, certaines spécifications

[39] La conception par écart consiste à concevoir par rapport à ce qui est proposé dans l'outil. La conception se fait en indiquant ce qu'il faut récupérer de l'outil et comment l'ajuster. Cette approche de travail permet d'accélérer l'étape de conception.

[40] METAPHASE offre près de 500 classes standards et dispose d'une couche "objet" proposant un ensemble de méthodes standards sur les classes. Ainsi, selon les besoins de l'application, on surcharge les méthodes (publiques) proposées par rajout de code.

fonctionnelles sont remises en cause durant cette phase, lorsque la faisabilité logicielle n'est pas possible (tel que par exemple un champ avec des espaces devant l'attribut).
Actuellement, comme le D.A.F.E, aucun format n'est utilisé pour spécifier le D.A.T.E (texte, tableaux). Par ailleurs, la conception (fonctionnelle et technique) "par écart" vis à vis de l'outil d'implantation (METAPHASE) est complètement dépendante de la connaissance du CPSI et de l'IA pour le SGDT considéré. Les composants proposés par le logiciel sont peu ou mal documentés. En l'absence de démarche formalisant le "Comment" de la spécification par écart à l'outil SGDT, le CPSI ou le IA doivent parfaitement maîtriser les composants afin de pouvoir indiquer ce qu'il faut récupérer et comment.

- **Le développement** : consiste en l'implantation par les développeurs des spécifications décrites dans le D.A.T.E. sur METAPHASE (codage des classes), suivie d'une série de tests unitaires sur chacun des modules développés séparément.

- **La livraison et les recettes** : la phase de développement est suivie par une activité de livraison et de tests internes (recettes) avant la mise en place de l'application en milieu opérationnel. Cette activité se déroule en deux temps. D'abord, l'application est livrée au CPSI qui conduit une phase d'intégration (de différents modules) et une série de tests afin de vérifier la conformité des développements par rapport au D.A.F.E. Cette phase nécessite dans la plupart des cas des corrections et par conséquence de nouvelles livraisons des développements. Ensuite, et en cas de conformité, l'application est livrée au CPU qui effectue également une série de tests sur le site de déploiement de l'application et demande éventuellement des corrections auprès du CPSI. C'est à l'issue de cette deuxième recette que le CPU prend la décision du déploiement de la nouvelle application.

- **L'installation** : a pour objectif l'installation et la validation de l'application chez les utilisateurs. Cette phase d'installation pourrait créer des perturbations organisationnelles non attendues si la mise en œuvre n'était pas anticipée. En effet, l'introduction d'un SIP suppose un changement culturel et une adaptation à un nouveau mode d'organisation où les utilisateurs sont face à un outil qui décloisonne l'entreprise et qui les oblige en conséquence à formaliser, à structurer et à partager l'information technique.

La durée d'un projet est variable. Pour le cas observé, la durée constatée est d'environ 90 jours, répartis comme suit :
- Expression des besoins : 10 jours
- Analyse Fonctionnelle : 10 jours
- Analyse Technique : 10 jours
- Développement & tests unitaires : 30 jours
- Livraison et recettes : 30 jours

Synthèse

En résumé, l'ingénierie des SIP sur le terrain n'est pas abordée d'une façon méthodique. Les entreprises industrielles rencontrent actuellement des difficultés techniques, méthodologiques, organisationnelles et humaines, liées :

- au manque de modèles formels d'expression des besoins suffisamment compréhensibles pour assurer une communication efficace et non ambiguë entre les acteurs concernés, et qui ne fait que ralentir le processus de développement à cause des multiples itérations nécessaires pour préciser les besoins et valider les spécifications fonctionnelles et techniques,
- à l'absence de continuum de transformations cohérentes des différents modèles élaborés durant les différentes phases d'ingénierie, ce qui mène à des incohérences entre ce qui est attendu et ce qui est livré,
- à la lenteur du développement des SIP qui engendre des coûts considérables et mène parfois à la livraison de systèmes déjà obsolètes,

Ce dernier point est particulièrement pénalisant dans le contexte des SIP, à cause de la forte fréquence des développements des applications SIP. En effet les SIP ont pour finalité de supporter le cycle de vie du produit, c'est à dire accompagner le produit au sein de l'organisation industrielle lors de sa conception, sa fabrication, sa production, … Une entreprise industrielle est ainsi amenée à développer constamment de nouveaux SIP pour de nouveaux produits ou de nouvelles familles de produits. De plus, le cycle de vie des produits industriels a tendance à être de plus en plus court. En conséquence, celui des SIP les supportant l'est aussi. Un problème majeur dans le développement des SIP est le déplacement de la cible (le SIP à développer) à atteindre au cours du temps de développement, ce qui conduit à livrer des systèmes obsolètes avant même leur mise en place en milieu opérationnel.

Ce problème est d'autant plus accentué que la taille de l'entreprise est grande. Tel est le cas de Schneider-Electric qui est amenée à développer plusieurs applications relativement proches pour servir à chaque fois un métier (mécanique, électronique, etc.), une activité (Basse Tension, Haute Tension, etc.) ou un site différent (France, Italie, etc.). Des applications à faire évoluer par la suite pour prendre en compte des changements organisationnels ou techniques relatifs à l'offre logicielle utilisée pour mettre en œuvre le SIP.

Une autre raison majeure à l'origine de la lenteur du développement et du déploiement des SIP est la difficulté des utilisateurs à accepter ou à comprendre l'apport des SIP. En effet, comme le rapporte une étude effectuée par CIMdata en collaboration avec IBM et le Ruhr Universität in Bochum [CIR, 99], les utilisateurs ont dans la plupart des cas une vision très floue des fonctionnalités et surtout des contributions que peuvent apporter les SIP dans leur métier. Une vision due à la jeunesse de ce dispositif et au manque d'indicateurs pour mesurer l'apport de tels systèmes. Cette perception retarde la prise de décision au niveau des dirigeants pour l'introduction de SGDT dans l'entreprise et rend difficile une contribution effective des utilisateurs aux phases d'expression de besoins et d'installation. En conséquence, le développement est plus lent et moins efficace et le retour sur investissement est plutôt décevant.

3.2.2. Etat de l'art sur l'ingénierie des SIP

Le domaine des SIP est un domaine relativement récent et donc qui n'a cessé d'évoluer. L'évolution des visions vis à vis de la GDT a fait évolué constamment le périmètre des SIP ainsi que les fonctions de ces systèmes. L'intérêt de la communauté de la GDT s'est orienté initialement vers *la définition du périmètre* de ces systèmes et de leurs fonctionnalités. Ceci va de la définition "conceptuelle" de ces fonctionnalités à la proposition des solutions techniques associées dans les SGDT du marché. Plus tard, les travaux de recherche se sont

intéressé à la *rationalisation de la gestion des données techniques.* Divers types de problèmes sont adressés dans ces travaux, notamment les problèmes d'échange, partage et intégration de données, les problèmes de configuration de produits et les problèmes de gestion de workflow.
D'autres travaux se sont intéressés à d'autres types des données gérées dans les SIP tel que la gestion des documents (projet ESPRIT Condor [Condor, 99]), la rationalisation des processus de modification de produits [Huang, 00] et la gestion des processus non conventionnels tels que la réparation de produit, la rénovation et le recyclage (projet ESPRIT EPSYLON [Bonfatti, 99]), etc.
Plus récemment, les travaux de recherche s'intéressent à la *rationalisation des processus de développement* des SGDT dans les entreprises.
Dans la suite, nous présentons chacun des principaux sujets de recherche que nous venons d'introduire ci-dessus.

1.1.1.1. Echange, partage et intégration de données

Pour une meilleure intégration de l'entreprise dans un contexte d'entreprise virtuelle et d'ingénierie concourante, les entreprises ont dû faire appel à des techniques de communication entre leurs différentes applications, telles que les échanges de données et le partage d'informations, accompagnés de la mise en place d'un modèle de référence fédérateur, supporté par le SGDT. Diverses actions, notamment normatives, ont été menées dans le domaine des échanges, partage et intégration de données techniques depuis les années 1980.

Les premiers travaux mettent l'accent sur *l'échange* de données. Ainsi, les premières normes proposées ciblaient des domaines d'application particuliers telle que la norme allemande *VDA* pour l'industrie automobile [VDA, 86], ou des méthodologies particulières telle que la norme américaine *IGES* [IGES, 80] créée au départ pour l'échange de données géométriques. Par la suite, au début des années 1990, l'ensemble de ces actions normatives ont été unifiées dans une norme internationale de l'ISO : ISO 10303 [ISO, 94a], connue sous le nom de *STEP* (**ST**andard for **E**xchange of **P**roduct Model Data). Cette norme est développée au sein du groupe ISO/TC184/SC4 et a pour objectif de permettre la représentation et l'échange de données produit en couvrant son cycle de vie[41]. STEP est basé sur une architecture de données et sur des méthodes pour développer les éléments de cette architecture. C'est un standard qui se veut multi-utilisations [Chambolle, 99] : pour *l'échange* de données en définissant un format de fichier neutre (la partie 21 de STEP [ISO, 94b]); pour *l'archivage* en couvrant la durée de vie des produits et pour les bases de données en permettant *l'intégration* des applications (le protocole d'accès *SDAI* ; partie 22 de *STEP* [ISO, 98b]). La norme STEP sera présentée plus en détail dans le chapitre II.

La norme *STEP* a été par ailleurs le moteur de plusieurs travaux de recherche et notamment des projets européens. Ainsi, associé à l'initiative AIT [AIT, 93], citons le projet *RISESTEP* [RIS, 98] qui s'intéresse à la spécification et au développement d'une architecture distribuée

[41] Notons que ce groupe développe, à côté de STEP, le standard *P-Lib* : ISO 13584 [ISO, 98a] qui traite des librairies de composants, le standard *Mandate* : ISO 15531 [Cutting-Decelle, 00] qui traite des données de gestion de production et le standard PSL : ISO 18629 [Schlenoff, 00] qui est un langage de spécification de process. Il a développé également une norme semblable à STEP mais destinée aux industries pétrolières et de process : la norme ISO 15926 initiée en 1993 par l'association POSC/CAESAR (http://www.posccaesar.com/).

pour l'échange et le partage d'information dans un contexte de maquette numérique et le projet OPAL [OPAL, 96] dont l'objectif est de fournir des concepts pour l'intégration de haut niveau des processus et de l'information dans le domaine de l'ingénierie et de la production.
Un standard d'échange dédié aux données PDM fut ensuite développé dans la communauté *STEP* : il s'agit du *PDM Schema* développé par PDES Inc. et ProSTEP [PDM Schema, 99]. *PDM Schema* est né d'un effort pour harmoniser les besoins et les modèles de données issus des protocoles d'application de *STEP*[42] qui couvrent les données PDM (notamment les protocoles *AP203* et *AP214*) et qui jusqu'alors n'étaient pas interopérables[43]. L'objectif de *PDM schema* est d'offrir un modèle de données unique supportant les données gérées dans les PDM. Ce modèle devrait permettre aux concepteurs de PDM d'implanter des fonctionnalités PDM en se basant sur STEP et favoriser ainsi l'interopérabilité des PDM avec d'autres systèmes basées sur les différents protocoles d'application de STEP. Ce standard fût adopté comme protocole d'échange dans le projet européen *Eurofighter* [Kerr, 99].

Toutefois, l'échange de données dans *STEP* n'offrait pas une solution satisfaisante aux problèmes d'utilisation coopérative d'outils CAO et SGDT; la conversion de données entraînait une perte d'information à chaque transformation et l'échange était lent entraînant ainsi des problèmes organisationnels à chaque transfert. Pour pallier à ce problème, un couplage fort et une intégration plus directe sont nécessaires [Lämmer, 00]. Deux nouveaux standards sont alors apparus pour supporter l'aspect partage dans les scénarios d'intégration de données produit. Il s'agit des *PDM Enablers* de l'OMG [OMG, 98a] qui supportent l'interopérabilité des systèmes en proposant une architecture d'interfaces standards (API : Application Programming Interface) aux divers services d'un DPM opérant dans un environnement distribué. Les *PDM Enablers* sont en fait des interfaces (API) standards spécifiés en IDL (Interface Definition Language) qui rendent les services d'un PDM disponibles pour d'autres systèmes (des systèmes CAO, IAO, FAO et également d'autres PDM) grâce à un environnement *CORBA*[44] (Common Object Request Broker Architecture). En se basant sur le modèle CORBA, les systèmes du type XAO peuvent utiliser les interfaces standards des *PDM Enablers* pour interagir directement avec tout PDM conforme. Actuellement, les *PDM Enablers* fournissent des interfaces directes pour spécifier la gestion des documents, la gestion des nomenclatures, la gestion des modifications et la configuration de produits (options et variantes du produit) et également l'import et l'export de fichiers d'échange STEP. Le deuxième standard est le langage *XML* (Extensible Markup Language) [W3C, 98] qui offre un format de spécification de données.

---

[42] Un protocole d'application comme décrit dans [OMG, 99b] est un modèle informationnel dans STEP servant un domaine d'application spécifique. Il est constitué de trois éléments : un modèle d'activité (AAM : Application Activity Model) décrivant le processus métier supporté, un modèle d'application de référence (ARM : Application Reference Model) spécifiant les besoins informationnels et un schéma conceptuel de données (AIM : Application Interpreted Model) spécifié dans le langage formel EXPRESS et qui est à la base d'implantation du standard (ISO 10303 Part 11).

[43] C'est à dire qu'un traducteur d'un protocole donné (AP 203 par exemple) n'est pas capable de lire un fichier produit par un traducteur d'un autre protocole (AP214 par exemple) et vice versa.

[44] CORBA est une architecture proposée par l'OMG (Object Management Group), déployant les principes fondamentaux pour définir et gérer les interfaces de systèmes distribués [OMG, 98b]. Elle fut le raffinement d'une première architecture proposée par l'OMG : l'Object Management Architecture (OMA). Ces architectures définissent les interactions entre composants d'un système distribué en terme d'invocation sur les opérations des objets encapsulés dans ces composants.

Actuellement, de nombreux travaux de recherche visent à offrir un cadre de référence pour l'échange de données, en utilisant les différentes techniques et standards d'échange et de partage de données proposées. Nous citons en particulier le projet *TIMS* (Testability of Interaction-Driven Manufacturing Systems) [Morris, 00] initié par le NIST (National Institute of Standards and Technology) dont l'objectif est de rapprocher les deux normes *PDM Enablers* de l'OMG et le *PDM Schema*, en établissant une correspondance entre ces deux standards. Nous citons également le projet européen Brite-Euram *SAVE* (STEP in A Virtual Enterprise) [Bodington, 00] dont l'objectif est de fournir un modèle informationnel supportant l'entreprise virtuelle, basé sur les résultats issus essentiellement des deux travaux normatifs *PDM Schema* et *PDM Enablers* et des projets *RISESTEP* et *AIT-IP* [AIT, 97d]. Ce dernier spécifie et implante une plate-forme d'intégration. Le projet Esprit EP 20408 - *VEGA* (Virtual Enterprises using Groupware tools and distributed Architectures) s'intéresse à l'intégration de STEP avec des modèles d'information, de CORBA avec une couche intermédiaire (middleware) fondée sur un courtier d'objets ORB (Object request Broker) et de standards du WEB comme VRML (Virtual Reality Modelling Language) pour la définition et le partage d'informations 3D à travers l'Internet et dans le cadre d'applications de réalité virtuelle [Zarli, 98].
D'autres projets s'intéressent aux échanges de types particuliers de données produit, tels que le projet *MERCI* (Management, Exchange, and Representation of Component Information) [Wikes, 00] pour la gestion de données de composants, basé sur la technologie *XML* et le schéma *EXPRESS* de *STEP*.

1.1.1.2. Configuration de produit

Durant la dernière décennie, de nombreux travaux se sont intéressés à rationaliser la configuration de produits dans un contexte de diversité et de spécialisation croissante des produits. La configuration de produit consiste à produire différentes configurations du produit selon différents besoins utilisateurs. Elle se fait par la mise en œuvre de diverses combinaisons de composants, d'options et de variantes et ce à partir d'une structure de base prédéfinie (appelée souvent structure générique) décrivant l'ensemble des composants qui peuvent être inclus dans le produit. Le développement de méthodes et d'outils pour supporter les activités de configuration de produit constitue le sujet d'intérêt de nombreux travaux de recherche. Ces travaux s'intéressent notamment à la modélisation du produit générique et à la résolution des problèmes de configuration. Un problème de configuration, comme défini dans [Soininen, 99a] est un problème de production d'une spécification d'un produit comme une collection de composants prédéfinis. Les entrées de ce problème sont un modèle de configuration décrivant les composants qui peuvent être inclus dans la configuration et les règles de combinaison de ces composants pour former un produit possible (réalisable). La sortie du problème est une configuration ; une description précise d'un produit qui peut être fabriqué, qui satisfait les besoins et qui est valide (respecte les règles de combinaison). Différentes méthodes, basées notamment sur les techniques d'intelligence artificielle, sont proposées pour la résolution des problèmes de configuration. Elles sont essentiellement à base de règles, de cas et plus récemment à base de contraintes. Un aperçu de ces méthodes est présenté par M. Stumptner [Stumptner, 97]. Parmi les travaux qui s'y intéressent, nous citons en particulier les travaux de l'équipe *Product Data Management Group*[45] de *Helsinki*

[45] http://www.cs.hut.fi/~pdmg/

University of Technology qui proposent un langage de configuration à base de règles (CRL : Configuration Rule Language) avec une sémantique déclarative pour définir de façon formelle les concepts en configuration de produit (cf. Figure 1.9) [Soininen, 99a] et approchent les problèmes de configuration en tant que problèmes de satisfaction de contraintes dynamiques[46] [Soininen, 99b].

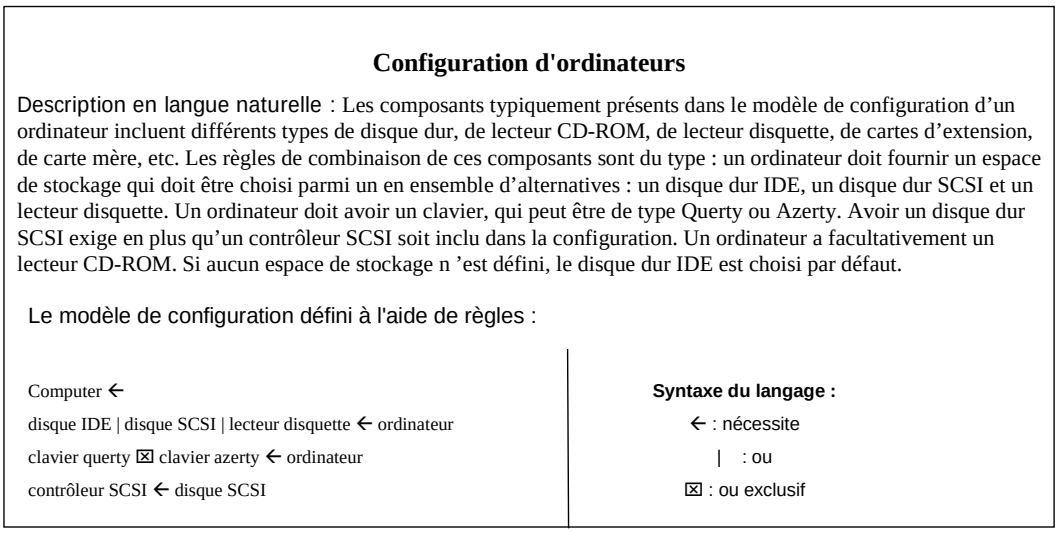


Figure 1.9 - Modèle de configuration à l'aide du langage CRL (*Source : [Soininen, 99a]*)

Dans [Ramachnandran, 99], l'approche EXPECT pour l'acquisition des connaissances pour la configuration propose d'utiliser la stratégie de résolution dite *propose-and-revise*[47] pour résoudre les problèmes de configuration. U. John [John, 99] propose une approche intégrée de résolution de contraintes basée sur une modélisation formelle du problème à laquelle est associé un langage de spécification déclarative appelé ConBaCon (Contraint Based Configuration) basé sur l'utilisation d'un langage CLP (Constraint Logic Programming) pour pallier aux approches utilisées dans la majorité des outils de configuration commerciaux, basés généralement sur la vérification des contraintes.
Dans les travaux de l'équipe de P. Schönsleben du *Swiss Federal Institute of Technology*[48] [Schwarze, 97], le processus de configuration est structuré en trois phases : mapping (mise en correspondance) des spécifications, configuration technique et choix & optimisation (cf. Figure 1.10). Un accent particulier est mis sur la formalisation de la première phase de "mapping des spécifications" qui consiste à transformer les connaissances fonctionnelles fournies par le client (souvent floues, implicites, de formats variés, et orientées application du produit que le produit lui-même) en une spécification technique (structurelle) précise et concrète du produit. Pour cela, un système expert est proposé, intégrant notamment la logique floue. Un modèle de données de configuration, commun à toutes les phases du processus de configuration est proposé pour intégrer les différentes connaissances utilisées [Schwarze, 96].

[46] Dynamique car dans la configuration, certains composants optionnels peuvent être ajoutés ou encore certains composants nécessitent l'existence d'autres composants. Il en résulte un problème de satisfaction de contraintes où les valeurs prises par les variables doivent changer dynamiquement en fonction des choix effectués au cours de la résolution du problème.

[47] Cette stratégie est basée sur des itérations successives d'extension et de révision de solutions. L'extension consiste à affecter des valeurs aux variables (la solution courante) et la révision consiste à vérifier le respect des contraintes et révision de la solution courante en cas de violation de contraintes en agissant sur la valeur des variable à l'origine de cette violation.

[48] http://www.lim.ethz.ch/english/index.html

Ce travail s'inscrit dans le cadre plus général du projet GNOSIS[49] de initiative l'IMS (Intelligent Manufacturing Systems).

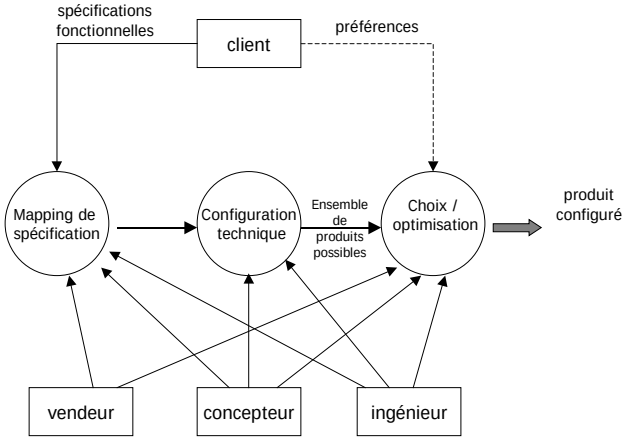


Figure 1.10 - Les trois étapes du processus de configuration (*Source : [Schwarze, 96]*)

1.1.1.3. Gestion du Workflow

La vision classique du workflow est essentiellement procédurale. Toutes les activités ainsi que leur enchaînement sont prédéfinies et toute information ne peut être transférée d'un acteur à l'autre que si elle a changé d'état. En conséquence, les processus workflow sont gérés d'une façon rigide. Les changements sur les processus ne sont pas autorisés et les systèmes de gestion de workflow ne peuvent être utilisés pour supporter dynamiquement les changements dans les processus. Par ailleurs, ces systèmes ne sont plus adaptées à des processus à forte variabilité tels que les processus d'ingénierie, où les activités sont définies à fur et à mesure de l'avancement du processus[50] et où plusieurs modifications sont nécessaires. La mise en œuvre de workflows flexibles adaptés aux processus à forte variabilité constitue un domaine de recherche actif. Dans ce cadre, nous citons en particulier le projet ADEPT (Application Development based on Encapsulated Pre-modeled Templates) [Hensinger, 00] démarré en 1994 et dont l'objectif est de développer un workflow adaptatif, qui supporte en temps réel les changements survenants sur des processus workflow en cours d'exécution. De telles adaptations sont nécessaires lorsque de nouvelles activités doivent être ajoutées au processus en cours ou lorsque les dépendances entre activités en terme de flux de données ou de contrôle doivent être modifiées dynamiquement. Pour cela, une modélisation formelle de workflow supportant la dynamique de changement des processus workflow est proposée. Elle permet de valider les corrections (modifications) apportées sur un modèle de workflow (*workflow template*) en cours d'exécution. Des concepts additionnels relatifs aux aspects temporels dans les workflows sont définis pour contrôler la cohérence lors de changements dynamiques dans un workflow, essentiellement lorsqu'il s'agit de processus itératifs.
L'approche WASA [Weske, 99] propose des workflows adaptatifs basés sur une affectation flexible de divers schémas et instances de workflow. Les schémas et instances de workflows sont modélisés en tant qu'objets, caractérisés donc par un état et un comportement et communiquant entre eux par envoi de messages. Un schéma de workflow complexe peut donc être obtenu par réutilisation et combinaison de ces différents schémas et instances.

[49] Site internet : http://www.lim.ethz.ch/english/index.html

[50] On parle de "ad-hoc workflow".

Nous citons également les travaux de W. Van Der Aalst [Aalst, 99] qui aborde le problème de flexibilité avec les mêmes techniques que celles utilisées dans la configuration de produit. Un modèle de processus générique est proposé à partir duquel différentes configurations de processus sont définies. Cette approche propose néanmoins une flexibilité à priori et reste statique puisque tous les scénarios de processus doivent être prédéfinis.
D'autres travaux s'intéressent plutôt à l'intégration des données d'ingénierie dans les workflows d'ingénierie[51]. Nous citons dans ce cadre le projet Esprit SIMNET (workflow management for SIMultaneous engineering NETworks) [Rouibah, 00].

1.1.1.4. Développement des SIP

L'intérêt de la communauté scientifique à formaliser et rationaliser le processus de développement des SIP est très récent. Cet intérêt pour le processus de développement est dû essentiellement aux délais et coûts considérables engendrés par l'implantation des SIP, créant ainsi une barrière significative pour les entreprises et surtout les PME qui veulent s'investir en SIP. Les travaux qui abordent les problèmes de développement des SIP sont rares. Nous proposons ici de citer quelques travaux traitant ce sujet, qui sont principalement des projets européens.

Ainsi, dans le cadre des projets ESPRIT, citons le projet RapidPDM (EP 26892) [Pels, 00] qui propose une méthodologie générique pour l'implantation de PDM, basée sur un ensemble de méthodes et d'outils informatiques pour supporter les premières phases de développement de PDM (sensibilisation, analyse des besoins, choix du logiciel). Un premier support qui donne un aperçu des avantages que peuvent offrir les PDM permet de sensibiliser les dirigeants d'entreprise sur l'importance des PDM pour leurs organisations. Un second support permet d'identifier les goulots dans le processus de définition du produit de l'entreprise concernée, de sélectionner les solutions à ces difficultés et enfin d'identifier les fonctionnalités de PDM dont l'entreprise a besoin. Enfin, un troisième support permet de sélectionner le progiciel SGDT le plus adapté à l'entreprise, de développer un plan d'implantation et d'évaluer le retour sur investissement de la solution choisie (en terme de progiciel et de plan d'implantation), avant d'entamer la phase d'implantation effective. Dans ce projet, il s'agit donc essentiellement d'outils d'aide à la décision pour les différentes phases de développement et non pas d'une démarche d'ingénierie explicitant en particulier les modèles à produire et la démarche à suivre pour élaborer ces modèles.
Dans les projets européens, nous citons également le projet STEPWISE (EP25110) [Stepwise, 00] qui formalise un processus pour l'implantation de modèles EXPRESS. Il définit une approche pour transformer progressivement les modèles EXPRESS utilisés pour exprimer les besoins (modèles d'analyse) en des modèles d'implantation (modèles de conception). Ce travail a l'avantage d'assurer un continuum de transformation des spécifications et de se conformer à la norme STEP mais a l'inconvénient d'utiliser les modèles EXPRESS pour l'analyse des besoins (analyse) qui, à notre avis, ne sont pas adaptés à cette phase vu qu'ils sont peu expressifs et difficiles à comprendre par les futurs utilisateurs du système.
D'autres travaux de recherche s'intéressent à l'ingénierie des SIP. Nous citons en particulier le projet allemand iViP[52] [Ehrler, 99] qui propose une méthodologie pour la paramétrisation des

[51] Plus connu sous le vocable de : engineering workflow (ewf).
[52] integrated Virtual Product development (projet allemand).

SGDT selon les modèles de STEP. Il offre une méthodologie pour implanter des composants de correspondance sémantique configurables[53] pour combler les faiblesses des modèles proposés dans des standards tels que STEP et PDM Enabler à supporter la définition des attributs et des valeurs d'attributs.
Nous citons également les travaux de D. Spath [Spath, 99] qui proposent une méthode pour la configuration de SGDT. Cette méthode a la même philosophie que celle utilisée dans la conception axiomatique[54] de produits. Elle se base sur l'analyse fonctionnelle.
Citons enfin les travaux de D. Belsnes [Belsnes, 99] proposant un cadre qui a pour objectif de faciliter la spécification et la construction de systèmes distribués inter-opérants. Il propose d'intégrer le modèle produit proposé dans POSC/CAESAR dans une architecture distribuée basée sur les Business Objects "BO" (la Figure 1.11 illustre un exemple de BO). Le développeur utilise alors des blocs de construction (les BO) au lieu des objets du modèle produit de POSC/CAESAR. Ce cadre est constitué d'une architecture de référence (selon laquelle différents systèmes sont conçus), d'un métamodèle (pour pouvoir étendre le modèle de l'architecture de référence), d'une infrastructure de BO et de règles de correspondance (entre BO et modèle de produit de POSC/CAESAR).

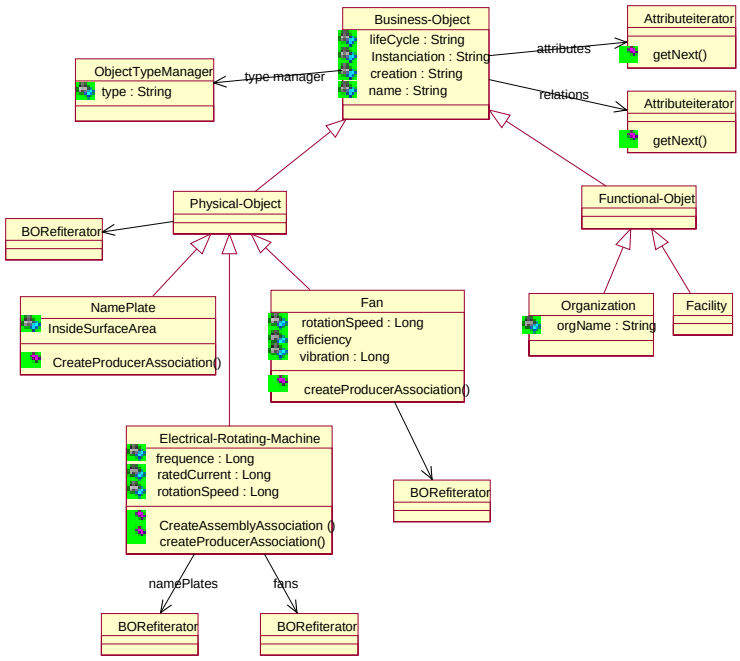


Figure 1.11 - Modèle d'Objets Métiers relatif à la définition d'un moteur électrique avec un ventilateur

Enfin, nous citons les travaux de Y-M Chen [Chen, 98] qui propose une méthodologie structurée pour l'implantation de systèmes EDM (Engineering Data Management) allant de l'analyse des besoins jusqu'à l'implantation du système. Cette approche bien qu'elle couvre la totalité du cycle de développement des EDM (en considérant en particulier les aspects organisationnels lors de la spécification), ne tire pas profit des approches théoriques d'ingénierie des SI, en particulier par l'absence d'un formalisme de modélisation conventionnel (les modèles obtenus par une telle approche ne peuvent être supportés par

[53] Configurable semantic mappers.
[54] Pour plus de détail sur ce sujet, voir [Suh, 99].

l'informatique). Ainsi, au passage de l'analyse à la conception du système informatique, le risque d'incohérence reste grand.

Synthèse

En résumé, les travaux relatifs aux SIP et leur ingénierie peuvent être classés en deux catégories selon le type de problèmes auxquels ils s'adressent :

- les travaux centrés sur les problèmes d'interface du SIP avec d'autres systèmes de l'entreprise. Actuellement les travaux proposés mettent l'accent sur les problèmes d'échange, d'intégration et de partage de données. Les premiers travaux se sont focalisés sur les problèmes d'échange de données produit. Une première norme a été alors proposée (STEP). Des travaux plus récents se sont intéressés davantage aux problèmes d'intégration et de partage de données produit. De nouveaux standards tels que le PDM Schema et PDM Enablers sont proposés à cet effet. Actuellement, les travaux de recherche s'intéressent au rapprochement des divers standards proposés pour offrir un cadre de référence pour l'échange, le partage et l'intégration de données
- les travaux centrés sur les problèmes de structuration des données dans le SIP. Ces travaux visent essentiellement à rationaliser la gestion des données techniques en structurant et formalisant l'ensemble des données gérées dans le SIP et en offrant des méthodes et des outils pour résoudre des problèmes liés à cette gestion. Deux catégories de travaux sont distingués : les travaux orientés produit (problèmes de configuration de produit) et les travaux orientés processus (problèmes de dynamisation du workflow).

Les travaux qui s'intéressent à l'ingénierie même des SIP, dans sa globalité, se font rares et sont plus récents. Les travaux que nous avons identifiés se focalisent essentiellement sur l'implantation de modèles normatifs. Certaines des difficultés rencontrées par les industriels lors du développement des SIP (soulignés au §3.2.1) restent mal ou peu soulevées dans ces travaux, voire non considérées. Il s'agit essentiellement des problèmes liés à la lenteur de développement et au manque de formalisation des démarches d'ingénierie.

4. Conclusion

L'ingénierie des systèmes d'information reste un domaine complexe qui partant de l'analyse des besoins d'une application, aboutit à une solution logicielle fiable dans un système technologique cible choisi. La Section 3.1 de ce chapitre a présenté l'évolution des approches académiques des travaux qui s'intéressent aux méthodologies d'ingénierie des systèmes d'information et a permis d'illustrer la richesse des développements théoriques effectués dans ce domaine.

En contrepartie, l'illustration de la pratique industrielle de l'ingénierie des SIP présentée dans la Section 3.2.1 a souligné la quasi-absence de méthodologies pour conduire les projets de développements de SIP dans les entreprises. Il en résulte un ensemble de difficultés méthodologiques, organisationnelles, humaines et techniques auxquelles font face les industriels. La revue des travaux de recherche qui s'intéressent à l'ingénierie des SIP présentée dans la Section 3.2.2 a montré que les efforts se sont essentiellement focalisés sur l'amélioration des performances des SIP d'entreprises (amélioration des échanges de données, optimisation de la configuration de produits, etc.) et que les travaux qui s'intéressent à la

rationalisation des processus de développement de SIP sont rares, malgré les coûts considérables engendrés par cette activité. Par ailleurs, la confrontation de ces travaux aux besoins industriels identifiés permet de préciser de nouvelles pistes de recherche à développer. Un certain nombre des difficultés rencontrées lors de l'ingénierie des SIP que nous avons déjà soulignées restent en effet non traitées dans les travaux de recherche présentés. C'est l'objet du chapitre suivant (chapitre II) de préciser la problématique de notre travail et présenter l'approche envisagée pour répondre à cette problématique, en tirant profit des développements théoriques effectués dans le domaine de l'ingénierie des SI.

Chapitre II :

Précision et Approche de la Problématique

1. Introduction

Le chapitre I de ce manuscrit a permis de définir le contexte d'ingénierie de systèmes d'information produit dans lequel se place ce travail de thèse. Il a décrit d'une part la pratique industrielle de l'ingénierie des SIP et d'autre part les travaux actuels destinés à améliorer cette ingénierie. Un certain nombre des difficultés rencontrées lors de l'ingénierie des SIP que nous avons déjà soulignées restent non traitées dans les travaux de recherche présentés. C'est l'objet de ce chapitre de préciser la problématique de notre travail et de présenter l'approche envisagée pour répondre à cette problématique, en tirant profit des développements théoriques effectués dans le domaine de l'ingénierie des SI.
Ce chapitre est organisé alors en deux parties. La première partie (§2) a pour objectif d'abord de poser les hypothèses principales sur lesquelles se basent la définition de la problématique et le travail réalisé et ensuite de préciser les objectifs de ce travail de thèse. La deuxième partie (§3) a pour objectif de puiser dans la littérature des méthodes et des moyens susceptibles de réaliser le cahier de charges posé dans la première partie, afin de fixer les éléments clés sur lesquels se base le travail proposé. Elle constitue ainsi la base théorique du travail développé.

2. Précision de la problématique

Il ressort de l'état de l'art présenté au chapitre I que la majorité des travaux destinés à améliorer l'ingénierie des SIP s'intéressent aux problèmes d'échange, de partage et de structuration des données produit, problèmes très sensibles dans le développement de SIP. Cependant, ils n'abordent pas dans sa globalité, l'ingénierie même des SIP, à savoir les aspects méthodologiques liés à leur conception et réalisation mais également à leur maintenance et évolution. Pourtant ce sont ces difficultés méthodologiques qui constituent actuellement des goulots dans le processus de développement, engendrant des coûts considérables et conduisant dans beaucoup de cas à l'échec des projets SIP. Il s'agit notamment :

- de la lenteur du développement des SIP qui engendre des coûts considérables et mène parfois à la livraison de systèmes déjà obsolètes,
- du manque de modèles formels d'expression des besoins suffisamment compréhensibles pour assurer une communication efficace et non ambiguë entre les acteurs concernés, et qui ne fait que ralentir le processus de développement à cause des multiples itérations nécessaires pour préciser les besoins et valider les spécifications,
- de l'absence de continuum de transformations cohérentes des différents modèles élaborées durant les différentes phases d'ingénierie, ce qui mène à des incohérences entre ce qui est attendu et ce qui est livré.

Par ailleurs, malgré la richesse des développements théoriques effectués dans le domaine de l'ingénierie des SI, les travaux de recherche traitant de l'ingénierie des SIP n'en tirent pas profit.

C'est sur ces hypothèses que se base la définition de notre problématique et le travail réalisé.
Ainsi le travail de thèse présenté dans ce manuscrit a pour objectif de mettre en place un cadre méthodologique pour l'ingénierie des SIP en abordant les trois aspects qui sont :

- La définition de *modèles de spécification* de différents niveaux d'abstraction, qui couvrent la complexité du SIP (représentation des processus, des versions de produit, etc.) et favorisent l'échange d'information entre acteurs,
- La mise en place d'une *démarche générale de conduite de projet SIP* pour construire les différents modèles. Démarche qui doit couvrir l'ensemble des phases du processus de développement des SIP (analyse, conception, réalisation), assurer un continuum de transformation dans les spécifications et qui soit flexible et adaptable au contexte de chaque organisation,
- *L'accélération du processus* de développement des SIP.

Telles sont les intentions du travail de thèse proposé. Cette partie du chapitre détaille chacune de ces trois intentions qui constituent le cahier des charges du cadre méthodologique développé. Nous présentons également l'approche adoptée pour répondre à ce cahier de charges.

2.1. Définir des modèles

Une des composantes principales d'une méthode d'ingénierie de SI comme présenté dans le chapitre I (§ 3.1.1) est la notion de *modèles*. Un modèle est un ensemble de concepts et de règles pour les utiliser, destiné soit à expliquer et construire la représentation des phénomènes conceptuels, organisationnels ou techniques, soit à expliquer et représenter les éléments qui composent le SI et leurs relations.
La notion de modèle ne peut être séparée de la notion de *langage*, lui aussi une composante essentielle d'une méthode. Un langage est défini par un ensemble de constructions qui permettent de décrire formellement les spécifications du SI élaborées aux différents stades du processus de conception en s'appuyant éventuellement sur les modèles de la méthode [Rolland, 88].
Il va de soi donc, que dans le cadre méthodologique que nous proposons, de définir d'abord les modèles qui seront employés dans ce cadre méthodologique ainsi que le langage à utiliser pour les spécifier.
Cet ensemble de modèles et ce langage doivent par ailleurs pallier aux difficultés rencontrées dans la pratique d'ingénierie des SIP à savoir :

- être suffisamment compréhensibles pour assurer une communication efficace et non ambiguë entre les utilisateurs et les concepteurs des SIP,
- couvrir la complexité des SIP : pouvoir représenter les divers objets du SIP tels que les processus workflow (avec tout l'aspect dynamique qui lui est rattaché), les versions d'objets, les structures de produit, les données de configuration et les diverses représentations du produit (modèles CAO, notes de calcul, …).

Nous nous arrêtons un peu sur ce dernier point pour illustrer les caractéristiques des données gérées dans le SIP. Conçu pour gérer le cycle de définition de produit, le SIP couvre entre autre le processus de conception de nouveaux produits et le processus de modification de produits déjà industrialisés. Dans ces deux processus, le produit n'est informationnellement stabilisé qu'à l'issue d'une série de transformations et de choix [Rieu, 99c]. Plusieurs transformations sont nécessaires du cahier des charges à la solution technique retenue. A

chaque étape de définition, les connaissances détenues sur le produit sont affinées et précisées ou au contraire déclassées. Par ailleurs les informations relatives aux produits sont :

- de natures différentes. Un produit admet plusieurs représentations correspondant à des niveaux morphologiques différentes (représentation fonctionnelle, représentation organique, etc.). Chaque étape du processus de définition du produit a pour rôle d'autoriser le passage d'un niveau morphologique à un autre (ainsi à la conception préliminaire ou détaillée, on passe d'une description fonctionnelle à une description structurelle constituée d'éléments de structure qui satisfont chaque fonction par exemple).
- de niveaux d'abstraction différents. Chaque représentation du produit est perçue à des niveaux d'abstraction différents. C'est le cas des décompositions successives des fonctions ou des éléments de structure d'un produit. Des informations de même nature, sur la structure par exemple, sont décrits différemment dans plusieurs graphes de décomposition pour exprimer des points de vue métiers différents (nomenclature d'étude, nomenclature de fabrication, nomenclature logistique, etc.).
- de niveaux de précision différents. A chaque étape de définition du produit, de nouvelles informations sont produites et permettent de compléter une ou plusieurs représentations du produit.

Il s'agit donc de gérer l'évolution simultanée de différentes représentations du produit.

En conséquent, ce travail de thèse doit :

- préciser **un formalisme de modélisation** de données qui soit capable de représenter les divers objets du SIP et qui soit suffisamment compréhensible par les utilisateurs,
- définir divers **modèles représentant les éléments qui composent les SIP**, c'est à dire l'ensemble des données techniques gérées dans le SIP et ce selon différents niveaux d'abstraction et de précision.

Dans la deuxième partie du présent chapitre (§3.1), nous étudions les différents formalismes utilisés pour modéliser les données, en particulier ceux adaptés à la modélisation des données techniques. Un accent particulier est mis sur le langage UML (*Unified Modeling Language*) que nous avons retenu pour la modélisation des éléments du SIP. Sont également étudiées différentes approches existantes pour la modélisation des éléments d'un domaine donné (le SIP en l'occurrence). Les modèles du SIP sont présentés dans les chapitres IV et V.

2.2. Définir une démarche

La deuxième composante primordiale dans une méthode d'ingénierie est la notion de *démarche* (cf. chapitre I - §3.1.1). Une démarche est le processus opératoire grâce auquel s'effectue le travail de modélisation, de description et de réalisation du SI. Il s'agit donc ici de la démarche générale de conduite de projets SIP.

Cette démarche doit couvrir l'ensemble des phases de développement des SIP. Elle doit en particulier couvrir les phases d'analyse des besoins et de conception, principales phases goulots dans le processus de développement des SIP. Elle est ainsi définie dans la perspective d'élaborer des spécifications non ambiguës et d'assurer un continuum de transformation des spécifications. Durant les phases d'analyse et de conception, la part de raisonnement humain

est prépondérante. L'exécution de telles activités ne peut être donc une simple application d'un plan préétabli plus ou moins immuable. Elle est surtout la conséquence de décisions humaines. La **démarche** que nous proposons est alors **inspirée du paradigme orienté décision**. Elle ne met pas l'accent sur les activités à entreprendre pour construire les modèles mais plutôt sur les décisions qui représentent des intentions[55].
Les activités d'analyse et de conception reviennent ainsi à un ensemble de prises de décisions sur différents produits à construire. Le concepteur se trouve donc à chaque fois dans une situation, c'est à dire un produit en cours de développement sur lequel il doit prendre une décision, qu'il perçoit avec une certaine intention à l'esprit [Rolland, 96]. Sa réaction dépend à la fois de la situation dans laquelle il se trouve et de l'intention qu'il a, en d'autres termes elle dépend du contexte dans lequel il se trouve. Le concepteur réagit donc contextuellement, souvent par analogie avec les situations dans lesquelles il s'est déjà trouvé impliqué dans le passé. C'est ainsi que la **démarche** que nous proposons **s'appuie** également **sur le paradigme contextuel**. La connaissance capitalisée doit être contextuelle, c'est-à-dire qu'il est indiqué comment adapter cette connaissance selon le contexte dans lequel elle est utilisée. Ce paradigme est basé sur le concept de *Contexte* qui associe une décision à la situation dans laquelle la décision peut être prise.

Ainsi, les paradigmes orienté-décision et contextuel sont-ils centraux dans notre approche méthodologique de développement des SIP. Ils introduisent une nouvelle perspective méthodologique où la vue traditionnelle du processus unique pour une organisation universelle laisse la place à celle d'un processus flexible constitué d'une variété de décisions à prendre, sur les produits à construire, selon la situation dans laquelle se trouve le concepteur. Nous étudions dans la deuxième partie de ce chapitre les moyens ou techniques qui pourront favoriser une démarche contextuelle.

Par ailleurs, cette démarche doit assurer un continuum de transformations cohérentes des différents modèles élaborés aux différentes phases d'ingénierie. Nous définissons dans le chapitre VI (§3.4) une démarche de transformation des différentes modèles proposés.

2.3. Accélérer le processus de développement

Une façon efficace de réduire la durée des différentes étapes de développement des SIP est de permettre des spécifications "en écart" tant d'un point de vue de la capitalisation et la réutilisation des concepts déjà rencontrés que de la prise en compte des ressources logicielles (composants ou systèmes) disponibles.
Ainsi la réutilisation, déjà efficace en génie logiciel, est-elle centrale dans notre approche méthodologique de développement des SIP, au niveau de la spécification et de la réalisation de SIP, mais aussi plus tard lors de leur évolution. Nous nous plaçons dans une **approche de capitalisation et de réutilisation des acquis** en terme de modèles de produits et de processus et ceci, à chacune des étapes du processus de développement du SIP :

[55] Pour illustrer ce paradigme, nous considérons l'exemple donné dans [Rolland, 96] : dans la définition d'un schéma entité/relation, la création d'une entité *client* est secondaire alors que l'intention de *vouloir représenter les clients* dans le SI est plus importante. Cette intention est généralement motivée par des arguments. Dans la cas présent, l'intention est motivée par le *besoin d'informations sur les clients* et la *nécessité de les identifier individuellement.*

- au niveau de *l'analyse* : il s'agit de réutiliser des expressions partielles de besoins provenant soit de SIP standards associés à des produits types et à des processus aussi standards qui pourraient être fournies par les éditeurs de SGDT, soit de SIP déjà développés dans l'organisation industrielle ;
- au niveau de *la conception* : la réutilisation de spécifications déjà proposées par le SGDT, ou issues d'autres SIP déjà conçus devrait permettre une importante réduction des délais ;
- au niveau de *la réalisation* : la réutilisation des composants logiciels standards proposés par les SGDT (qui sont principalement des boites à outils, proposant des bibliothèques de classes), permet d'accélérer l'étape d'implantation du SIP. Cependant, ces composants logiciels sont peu ou mal documentés et le chef de projet SIP doit parfaitement les maîtriser afin d'élaborer des spécifications détaillées en indiquant ce qu'il faut récupérer et comment. Ces spécifications ne peuvent être que réalisées en écart par rapport à l'existant.

Si la réutilisation est actuellement assez bien traitée au niveau de l'implantation (bibliothèques de composants logiciels, d'applicatifs), elle reste très limitée au niveau de la conception, voire inexistante au niveau de l'analyse. Le cadre méthodologique que nous proposons offre donc des éléments d'une part pour pallier aux limites des travaux actuels sur l'ingénierie des SIP (offrir une démarche flexible de développement couvrant la complexité des SIP, favoriser le dialogue entre les acteurs et assurer des transformations cohérentes des spécifications) et d'autre part pour permettre la réutilisation des connaissances acquises par les développeurs d'applications dès le niveau de l'analyse des besoins du SIP.

Pour appréhender cette capitalisation et cette réutilisation des acquis dans ce cadre méthodologique, nous étudions dans la deuxième partie de ce chapitre (§3) les différentes techniques de réutilisation déjà développées dans le domaine de l'ingénierie du logiciel (§3.2) et plus particulièrement celle associée à la technique des patrons que nous avons retenue pour les SIP (§1.1.1.11).

2.4. Synthèse

De la confrontation des développements théoriques et des besoins industriels présentés au chapitre I, les hypothèses principales sur lesquelles se basent la définition de la problématique et le travail réalisé concernent :

- le manque de *modèles formels* d'expression des besoins suffisamment compréhensibles pour assurer une communication efficace et non ambiguë entre les acteurs concernés,
- l'absence de *continuum* de transformations cohérentes des différents modèles élaborées aux différentes phases d'ingénierie,
- la *lenteur* de développement des SIP.

Ainsi le travail de thèse présenté dans ce manuscrit a pour objectif de mettre en place un cadre méthodologique pour l'ingénierie des SIP en abordant les trois aspects qui sont :

- La définition de *modèles* de spécification de différents niveaux d'abstraction couvrant la complexité du SIP et favorisant l'échange d'information entre acteurs,
- La mise en place d'une *démarche* générale de conduite de projet SIP pour construire les différents modèles couvrant l'ensemble des phases du processus de développement

des SIP (analyse, conception, réalisation) et assurant un continuum de transformations dans les spécifications. Cette démarche doit être flexible et adaptable au contexte de chaque organisation,
- L'*accélération* du processus de développement des SIP.

Pour répondre à l'ensemble de ces objectifs, ce travail de thèse se base sur :
- un ensemble de *modèles* représentant les éléments qui composent les SIP à différents niveaux d'abstraction,
- une *démarche contextuelle* pour la conduite de projet SIP,
- la *réutilisation de composants* pour favoriser les spécifications par écart à toutes les phases d'ingénierie.

Tel est le cahier des charges du cadre méthodologique proposé. Dans la partie suivante (§3), nous puisons dans la littérature récente sur la modélisation de l'information et la réutilisation en ingénierie des SI, des formalismes de modélisation et des techniques de réutilisation qui pourraient permettre de satisfaire au mieux le cahier des charges ainsi posé.

3. Bases Théoriques de l'Approche Proposée

L'objectif de cette partie est triple : d'abord délimiter nos choix sur les formalismes de modélisation et sur les techniques de réutilisation à employer, puis justifier ces choix, et enfin démontrer comment nous pouvons approcher une démarche de développement basée sur le paradigme contextuel.

La section 3.1 est dédiée aux techniques de modélisation de l'information. Elle présente, à partir d'une étude des diverses techniques de modélisation proposées dans la littérature, le langage UML (*Unified Modeling Language*) comme le formalisme le mieux adapté au contexte d'ingénierie des SIP. Pour l'appréhender d'une façon la plus complète possible, cette étude est d'abord abordée d'un point de vue général en puisant dans la littérature des techniques de modélisation utilisées en ingénierie des systèmes d'information de gestion (issues de méthodes comme Merise, OMT, etc.). Ensuite, nous étudions les techniques de modélisation spécifiques, dédiées aux systèmes d'informations techniques (issus des projets normatifs tels que STEP). Nous focaliserons notre choix sur le langage UML. Nous présentons ce formalisme ainsi que l'avantage de son utilisation pour notre besoin.

La section 3.2 est consacrée à la présentation d'approches existantes ou actuellement en cours de développement qui s'inscrivent dans le domaine de la réutilisation en ingénierie des SI. Nous abordons cette présentation selon deux dimensions de la réutilisation : produit (de réutilisation) et processus (de réutilisation). La dimension produit présente les différents modèles de composants réutilisables proposés dans la littérature. Nous focaliserons notre choix sur les approches à base de patrons (*patterns*). La dimension processus concerne les processus de développement à base de composants. Cela concerne d'une part les processus utilisés pour l'ingénierie de composants (production de composants) et d'autre part les processus d'ingénierie de SI par réutilisation de composants. Nous focaliserons notre choix sur l'approche d'ingénierie de domaine pour l'ingénierie de composants.

3.1. Techniques de Modélisation en SI

En s'inspirant des propos de P. Maret [Maret, 96], "les connaissances d'un domaine constituent un capital fragile tant qu'elles ne sont pas formalisées et explicitées, car elles ne sont pas réutilisables, ni partageables ni persistantes en l'absence de leur détenteur. Pour rendre les connaissances maîtrisées et pérennisées et donc ré-exploitables, les connaissances doivent être formalisées. La formalisation des connaissances passe par leur description au travers de formalismes ou de langages. Cette description se fait de façon plus naturelle et compréhensible que le formalisme est adapté au contenu à modéliser".
Il est important donc de choisir le bon formalisme pour le domaine que nous étudions, celui des SIP. Les critères de choix du formalisme de modélisation, posés dans le §2 mettent l'accent sur la capacité du formalisme à assurer un continuum de transformations entre les modèles, à couvrir la complexité des éléments gérés dans le SIP et à offrir des modèles compréhensibles par les utilisateurs.
Dans la suite de cette partie, nous étudions les techniques de modélisation de systèmes d'information. Comme précisé précédemment, seront étudiées aussi bien les techniques de modélisation généralistes développées pour les systèmes d'information de gestion que les techniques de modélisation spécifiques aux systèmes d'information technique.
A l'issue de cette présentation, une discussion est menée afin de voir comment les critères posés dans le §2 pourraient être pris en compte dans les techniques proposées. Cela nous permettra de choisir le formalisme le plus adapté à notre contexte.

Néanmoins, des notions différentes existent pour désigner la modélisation de l'information. On parle de formalisme, de langage, de notation, de modèle, etc.. Pour lever toute ambiguïté terminologique, nous proposons dans l'Annexe A de définir et positionner certaines de ces notions. L'étude que nous présentons ici concerne les *formalismes* de modélisation.

3.1.1. Formalismes généralistes

L'histoire des formalismes de modélisation est très liée à celles des méthodes d'ingénierie de SI que nous avons développées au chapitre I (§3.1). Nombreux sont les ouvrages qui décrivent des méthodes d'ingénierie de SI avec les modèles de spécification et les démarches de modélisation associées. Leur description exhaustive sort du cadre de cette thèse. Nous référons le lecteur aux différents ouvrages de référence de ces méthodes[56].
Le but ici est de décrire plutôt les formalismes et modèles que nous considérons à la base de la modélisation dans le domaine des SI et utilisés dans la majorité des méthodes d'ingénierie de SI.
La majorité des méthodes proposées se basent sur une notation ou un ensemble de diagrammes permettant de représenter le SI à développer selon différentes "vues" ou "axes de modélisation" : l'axe structurel (aspect statique du SI : ce qu'*est* le système - les données), l'axe comportemental (aspect dynamique du SI : comme *se comporte* le système - les processus et les transformations sur les données) et l'axe fonctionnel (architecture du SI : ce que *fait* le système - les activités). Cette modélisation se fait progressivement, par niveau de préoccupation, selon différents "niveaux d'abstraction". La méthode Merise propose ainsi quatre niveaux d'abstraction : le niveau conceptuel (description des entités du SI et des

[56] Nous décrivons toutefois en Annexe A, le formalisme retenu dans notre étude : le langage UML.

processus métiers agissant sur ces entités : données et traitements), le niveau organisationnel (qui agit sur quelles données, de quelle façon et où - il conduit à faire émerger le système informatique du système d'information), le niveau logique (architecture d'application - modélisation du système informatique) et le niveau physique (l'outil informatique).
F. Vernadat [Vernadat, 99] parle plutôt de "points de vue" à modéliser dans une méthode, et qui sont relatifs aux aspects fonctionnel (fonctions et processus du système), informationnel (entités manipulées), organisationnel (structure organisationnelle manipulant les entités et les processus), etc.
Ces axes de modélisation, points de vue et niveaux d'abstraction se retrouvent partiellement ou complètement et sous diverses formes dans les méthodes existantes d'ingénierie de SI.

Dans la suite, nous présentons les formalismes ou modèles utilisés pour chacun des axes de modélisation du SI (structurel, dynamique et fonctionnel). Plusieurs formalismes et modèles sont proposés dans la littérature pour chacun de ces axes. Les plus connus et les plus utilisés sont : le *modèle relationnel* et le *modèle entité-association* pour l'aspect statique, les *diagrammes de flux* pour l'aspect fonctionnel et les *diagrammes d'états-transitions* ainsi que les *réseaux de pétri* pour l'aspect dynamique.
Nous présentons succinctement ces différents modèles. Le lecteur souhaitant plus de détails sur les modèles utilisés en ingénierie des SI pourra se reporter à [Giraudin, 00].
Mais avant cela, nous souhaitons présenter les deux grands paradigmes qui ont orienté la modélisation en SI, le paradigme fonctionnel et le paradigme objet.

1.1.1.5. Paradigmes de modélisation

En terme d'"approche" de modélisation utilisée dans les méthodes, l'ensemble des méthodes incorporent des composants de modélisation similaires et supportent plus ou moins les trois aspects (statique, dynamique, fonctionnel). Elles diffèrent toutefois par la place donnée à chaque composant de la modélisation. Chaque méthode est en effet dominée par un paradigme donné. Nous distinguons le paradigme fonctionnel d'un coté et le paradigme objet de l'autre.

- Dans le ***paradigme fonctionnel***, le système est conçu d'un point de vue fonctionnel, en partant d'une vue de haut niveau affinée successivement afin d'obtenir une conception plus détaillée. Un système est d'abord vu comme offrant une ou plusieurs fonctions à l'utilisateur [Rumbaugh, 95]. L'état du système est centralisé et partagé par les fonctions qui agissent sur cet état. L'analyse et la conception proposées dans les travaux d'E. Yourdon par exemple utilisent cette stratégie [Sommerville, 92]. L'accent est mis sur la spécification et la décomposition des fonctionnalités du système. Celui-ci est alors décomposé selon un critère fonctionnel. Les fonctions du système sont identifiées, puis décomposées en sous-fonctions et cela d'une façon récursive jusqu'à l'obtention d'éléments simples, directement représentables dans les langages de programmation. Dans cette approche où la fonction et la hiérarchie sont les mécanismes intégrateurs, la fonction induit la structure [Rumbaugh, 95]. Du fait du couplage statique entre architecture et fonctions, il en résulte que des évolutions fonctionnelles peuvent impliquer de lourdes modifications structurelles. Les méthodes telles que Merise et IDEF sont basées sur ce paradigme.

- Contrairement à cela, dans le ***paradigme objet***, le système est vu comme un ensemble d'objets, plutôt que comme un ensemble de fonctions. L'état du système est décentralisé, et chaque objet gère l'information concernant son propre état. Les méthodes objet proposent une décomposition non pas basée uniquement sur ce que le système fait mais plutôt sur l'intégration de ce que le système est et ce qu'il fait [Muller, 97]. Il se concentrent d'abord sur l'identification des objets du domaine d'application puis font coller les procédures à ces objets [Rumbaugh, 95]. Le système est en fait une société d'objets dissociés (comprenant à la fois une structure de données et un comportement) et qui coopèrent. Pour coopérer, les objets utilisent des messages qu'ils s'envoient entre eux par divers mécanismes qui dépendent de l'environnement de mise en œuvre [Kettani, 98]. Le système est entendu au sens large dans ces définitions (entreprise, logiciel, système informatique, etc.). Dans ce paradigme, les fonctions apparaissent donc sous la forme de collaborations entre les objets qui composent le système. Le couplage est dynamique et les évolutions fonctionnelles ne remettent plus en cause la structure statique de l'architecture. Ainsi le paradigme objet tire sa force de sa capacité d'intégrer statiquement et dynamiquement les constituants du système. Les méthodes objet telles que Booch et OOSE sont dominées par ce paradigme.
  En termes de concepts de base, certains auteurs [Kettani, 98] présentent le paradigme objet comme basé sur cinq concepts fondateurs : les *objets*, les *messages*, les *classes*, l'*héritage* et le *polymorphisme* pour exprimer de manière uniforme l'analyse, la conception et la réalisation d'une application informatique. D'autres considèrent que l'ensemble de ces points ne sont que la conséquence de la capacité d'intégration, d'unification du formalisme objet [Muller, 97]. J. Rumbaugh [Rumbaugh, 95] considère les aspects: identité, classification, polymorphisme et héritage comme étant les caractéristiques exactes du paradigme objet. Il considère comme concepts de base, les notions d'*objet*, de *classe*, d'*association*, de *généralisation* et d'*héritage*. Par ailleurs, plusieurs mécanismes sous-jacents mais non spécifiques à l'approche objet lui sont associés tels que l'abstraction (raisonnement sur ce qu'est et ce que fait un objet sans se soucier de comment l'implanter), l'encapsulation (séparation des aspects externes et donc visibles d'un objet des détails de son implémentation internes et donc invisibles aux autres objets), la combinaison de la structure des données et du comportement (dans des classes) et le partage (grâce à l'héritage) [Rumbaugh, 95].
  Une description détaillée de l'ensemble des concepts de bases associés au formalisme objet est proposée dans l'ouvrage collectif de C. Oussalah [Oussalah, 97a].

1.1.1.6. Modélisation Fonctionnelle

La dimension fonctionnelle vise à décrire *ce que fait le système*. Elle est essentiellement basée sur l'utilisation de modèles de flux. Un modèle de flux est une représentation des mouvements de données à l'intérieur d'un SI et entre ce système et son environnement. Il permet de formaliser les flux d'information entre les acteurs du système sous forme d'échanges de messages entre les acteurs. Nous définissons dans ce qui suit les principaux éléments d'un modèle de flux :

- Un *acteur* est une classe de personnes ou de systèmes qui interagissent avec un système, qui est source ou destination des données. Il déclenche, contrôle ou réalise des activités, élabore ou utilise des informations et reçoit ou émet des messages.

- Une *activité* est un ensemble de traitements homogènes qui transforment ou manipulent des données [Rieu, 99b].
- Un *message* est un élément de communication entre acteurs. Un message a un acteur expéditeur et un acteur destinataire. Il déclenche une activité de traitement de l'information chez l'acteur destinataire et véhicule de l'information de nature donnée ou décision [Giraudin, 00].

Les modèles de flux sont utilisés dans Merise sous forme de MFC (Modèle de Flux Conceptuel). Dans OMT et UML, ils se présentent sous forme de diagrammes de collaboration entre objets.

1.1.1.7. Modélisation des données

La modélisation des données vise à décrire *ce qu'est le système* en définissant les informations traitées dans le SI. H. Gomaa [Gomaa, 93] distingue deux approches principales pour la modélisation de données : la modélisation sémantique et l'approche JSD (*Jackson System Development*). Dans les modèles sémantiques, le domaine du problème est modélisé sous la forme d'objets (du domaine), d'attributs et de relations entre les objets. Dans l'approche JSD, les objets du domaine du problème sont modélisés en terme de tâches concourantes (processus).

L'approche la plus connue et la plus utilisée étant celle des modèles sémantiques. Nous nous focalisons sur celle-ci. Divers modèles ont été proposés dans cette approche. Nous retenons en particulier le *modèle relationnel* (n-aire et binaire) et le *modèle entité-association*.

- Le *modèle relationnel* : à la base des SGBD relationnels, ce modèle est certainement l'un des premiers modèles formalisant les bases de données. Il fut proposé par E.F. Codd en 1970. Le modèle relationnel s'appuie sur la notion mathématique de *relation* et il est basé sur l'algèbre relationnelle. Une base de données est alors structurée en tables (ou relations).

  Une relation est un sous-ensemble du produit cartésien de domaines où un *domaine* représente un ensemble de valeurs de données (semblable à la notion de type dans les langages de programmation). Une relation est ainsi un ensemble de n-uplets où n est le nombre d'attributs de la relation. Un attribut est défini sur un domaine ; plusieurs attributs d'une relation peuvent être définis sur un même domaine. Un n-uplet correspond donc à une ligne de la table (une instance) et le nombre *n* correspond au nombre de colonnes de la table (appelé *degré* de la relation). Le SI est alors vu comme des ensembles de données (des relations) quasi-autonomes. Ces relations sont toutefois dépendantes les unes des autres et plusieurs contraintes peuvent être exprimées sur les relations, dont la plus connue est la contrainte d'intégrité référentielle. Cette dernière contrainte constitue désormais la seule forme exprimant des associations entre les relations.

  Un langage relationnel est associé à ce modèle pour manipuler les relations et extraire différentes informations à l'aide de différents opérateurs.

  D'autres travaux ont ensuite porté sur la définition de modèle relationnel binaire à l'origine des méthodes NIAM[57] [Habrias, 88].

[57] L'acronyme NIAM désignant initialement "Nijessen's Information Analysis Methodology" a été généralisé pour désigner "Natural language Information Analysis Method". Cette méthode est également désignée par la nomination plus générale d'ORM (Object-Role Modeling) [Hay, 99].

- Le *modèle Entité-Association* : introduit par P. Chen au milieu des années 70, ce modèle constitue certainement le modèle plus connu et le plus utilisé pour la modélisation conceptuelle de données. Il est à la base du modèle conceptuel de données de la méthode Merise mais aussi de la plupart des modèles statiques de classes d'objets dans les méthodes objets (avec quelques extensions). Il s'agit d'un modèle sémantique de données fournissant une représentation axée sur la compréhension de la nature des données et de leurs associations de manière indépendante de l'implantation qui en sera faite. Il vient pour pallier à la difficulté du modèle relationnel à exprimer d'une façon explicite des associations conceptuelles entre ensembles de données [Giraudin, 00]. Ainsi, contrairement au modèle relationnel basé sur un concept unique ; celui de relation, le modèle entité-association est basé sur deux concepts principaux : les *entités* et les *associations*. On distingue clairement les ensembles de données (entités) et les relations entre les ensembles de données (associations). Ce modèle se base en plus sur une représentation graphique (diagrammes entité-association).
  - Une entité correspond à quelque chose qui existe dans l'environnement à modéliser. Elle correspond à la notion d'élément d'un ensemble (objet, occurrence, instance) que l'on peut distinguer et décrire. Les entités de même nature et partageant un ensemble de propriétés communes sont regroupées dans des *classes d'entités* selon le principe courant de classification. Par abus de langage on confond généralement entité et classe d'entités [Giraudin, 00].
  - Une association décrit une liaison entre au moins deux entités. De même que les entités, les associations sont regroupées dans des classes d'associations et par abus de langage on confond souvent association et classe d'associations.

  Les caractéristiques des entités ou des associations sont détenus par les attributs. Les entités ont chacune un attribut clé (ou identifiant) qui permet d'identifier de façon unique chaque occurrence d'une entité. Les associations n'ont pas d'attributs clé spécifiques et explicites (elles sont constituées de la concaténation des clés des entités liées). Les associations sont caractérisées en plus par des cardinalités. Les cardinalités d'une relation permettent d'exprimer le nombre minimal et maximal de fois qu'une occurrence d'une entité peut être impliquée dans une occurrence de l'association.

  Ainsi, dans une architecture normalisée de SGBD à trois niveaux telle que préconisée dans le rapport d'ANSI [ANSI, 75], le modèle entité-association constitue le niveau conceptuel (structure conceptuelle des données) alors que le modèle relationnel peut être utilisé au niveau interne (structure logique des données). Dans certaines méthodes (telles que Merise), des règles de passage d'un schéma entité-association à un schéma relationnel sont définies.

  Le modèle entité-association fut ensuite étendu pour introduire le concept de sous-type et le concept de composite. Il s'agit du modèle entité-relation étendu. Cette extension a été décrite par M. Flavin et R. Brown [Hay, 99]. Le sous-type d'une entité représente un sous-ensemble d'occurrences d'une autre entité qui est son super-type. Ainsi une occurrence d'un sous-type est aussi une occurrence de ce super-type et une occurrence du super-type est aussi une occurrence d'exactement l'un ou l'autre des sous-types. Un sous-type est lié à son super-type par une relation du type "est un" ("*is a*").

Les modèles de classes d'objet proposés dans les méthodes objet reprennent les concepts du modèle entité-association avec les extensions comme la "généralisation" (déjà introduite dans le modèle entité-relation étendu) et la "composition" (agrégation). Ils introduisent en plus le point de vue dynamique directement au niveau des classes d'objets par les notions d'opérations de gestion des objets de chaque classe. Dans la section suivante, nous parlons des aspects dynamiques dans la modélisation des SI.

1.1.1.8. Modélisation de la Dynamique

L'aspect dynamique ou comportemental du SI se manifeste principalement à deux niveaux:
- à un niveau local : comportement individuel des éléments (entités, objets) d'un SI,
- à un niveau global : fonctionnement dynamique du SI.

Par ailleurs, diverses formes pour la représentation de l'aspect dynamique sont proposées dans les méthodes d'ingénierie des SI. Nous proposons de les étudier dans ce qui suit.

Dynamique locale :

Pour la dynamique des éléments, il s'agit de *modèles d'évolution des entités* (ou objets). Ils sont apparus avec les premières méthodes orientées objet et sont principalement basés sur des automates d'états finis. Ces automates formalisent l'évolution des entités ou objets du SI par une succession d'états et des modalités de changement d'états. La représentation graphique de l'évolution des entités ou des objets, en utilisant un automate d'états finis, constitue ce qui est communément connu sous le nom de "diagramme d'état"[58]. Cette notion de diagramme d'état fut ensuite formalisée et étendue par D. Harel qui propose les "statecharts" pour combler certaines lacunes de l'usage des diagrammes d'états telles que leur inadaptation dans le cas d'un grand nombre d'états et de transitions, l'absence de définition de concurrence, etc. [Giraudin, 00]. Les "statecharts" sont des automates hiérarchiques qui possèdent des concepts d'orthogonalité, d'agrégation et de généralisation (d'états).
Dans ce qui suit, nous définissons quelques concepts de base dans les modèles d'évolution des entités ou objets (état, transition, événement, condition de garde, etc.).

- *Etat* : un état est une situation caractéristique au cours de la vie d'une entité ou objet. Il est caractérisé par un nom et une durée [Giraudin, 00]. Une entité ou objet est toujours dans un état donné pour un certain temps et ne peut être dans un état inconnu ou non défini. Il est l'image de la conjonction instantanée des valeurs contenues par les attributs de l'entité ou l'objet et de la présence ou non de liens, de l'entité ou l'objet considéré vers d'autres entités (ou objets) [Muller, 97].
- *Transition d'états* : lorsque les conditions dynamiques évoluent, les entités ou objets changent d'état en suivant les règles décrites dans l'automate associé à leurs classes. Une transition est une connexion unidirectionnelle entre deux états. Suite à un événement qui survient dans le domaine du problème, une transition est déclenchée et provoque le passage d'un état à un autre. Le passage d'un état à un autre est instantané car l'entité ou l'objet doit être dans un état connu [Muller, 97].
- *Evénement* : un événement correspond à l'occurrence d'une situation donnée du domaine du problème. Il peut être externe, interne ou temporel. Lorsqu'il est interne, l'événement est généré par au moins une transition d'états d'entités ou d'objet [Rieu,

[58] Appelé Cycle de Vie des Objets (CVO) dans Merise/2, diagramme d'états-transitions dans OMT, etc.

99b]. Les événements déterminent quels chemins doivent être suivis. Un événement peut déclencher plusieurs transitions. Dans ce cas, des conditions de garde sont placés sur les transitions.

- *Garde* : une garde est une condition booléenne qui valide ou non le déclenchement d'une transition lors de l'occurrence d'un événement. Lorsque plusieurs transitions peuvent être déclenchées par le même événement, les gardes (qui doivent être mutuellement exclusives) sont évaluées et une seule transition est validée puis déclenchée.
- *Diagramme d'état* : c'est un graphe orienté où les nœuds sont les états et les flèches sont les transitions entre états. Pour un automate déterministe, le graphe d'état ne doit pas laisser de place aux constructions ambiguës. En particulier il faut toujours décrire l'état initial de l'entité ou de l'objet. En revanche, il est possible d'avoir plusieurs états finaux qui correspondent chacun à une condition de fin différente. Il est possible également de n'avoir aucun état final, dans le cas par exemple d'un système qui ne s'arrête jamais [Muller, 97].

Dynamique Globale

La modélisation de la dynamique de l'ensemble du SI concerne les traitements, leurs conditions de déclenchement, leurs actions vis-à-vis des informations, leurs enchaînements, leurs modes d'exécution, leurs ressources (matérielles, humaines), etc.. Elle part généralement de la modélisation fonctionnelle du SI, en y introduisant la notion de temps et de séquence et en raisonnant sur des activités de granularité plus fine. Chacune des méthodes d'ingénierie a proposé des modèles différents. Ainsi dans Merise, la représentation dynamique est décrite par le MCT (Modèle Conceptuel de Traitement). Dans Merise/2, c'est le MCTA (Modèle Conceptuel de Traitement Analytique) qui met en évidence les interactions entre les données (objets) et les traitements (grâce au CVO). Dans les méthodes objet OMT, UML, il s'agit essentiellement de diagrammes d'activités.
Quelque soit le modèle utilisé, il s'agit de décrire l'activité de l'entreprise en éliminant les considérations d'organisation telles que la répartition du travail, la nature des tâches et les contraintes techniques [Rieu, 99b].
J-P. Giraudin [Giraudin, 00] note que la majorité des modèles proposés sont fondés sur la technique des *réseaux de pétri* où les places représentent les types d'événements, les marques correspondent au nombre d'occurrence de ces événements, les transitions sont assimilées aux opérations de traitement de l'information et le marquage correspond à un état du système qui évolue en appliquant les règles de franchissement des transitions.

3.1.2. Approches spécifiques : la norme STEP

Outre les formalismes généralistes provenant des méthodes d'ingénierie de SI, le second type de formalismes utilisables pour la modélisation des SIP est celui proposé dans les projets normatifs dédiés à la gestion des données techniques. Sera donc étudié dans cette section le formalisme proposé dans la norme STEP[59] (ISO 10303) et les autres normes[60] développées

[59] STandard for the Exchange of Product-model data.

[60] Plib : ISO 13584 [ISO, 98a], Mandate : ISO 15531 [Cutting-Decelle, 00], PSL : ISO 18629 [Schlenoff, 00] et l'initiative de POSC/CAESAR : ISO 15926.

par le même groupe de l'ISO; le TC184/SC4 qui, rappelons-le, a pour mission la normalisation de l'information pour le partage et l'échange dans les applications industrielles et manufacturières. Nous nous focalisons sur l'étude de STEP, qui est destiné exclusivement à la représentation des données produit. Les autres standards (Plib et Mandate) utilisent la même notation que celle utilisée dans STEP.
Pour commencer, nous présentons la documentation de STEP. Celle-ci est constituée d'une série de parties (*part*) qui sont les suivantes :

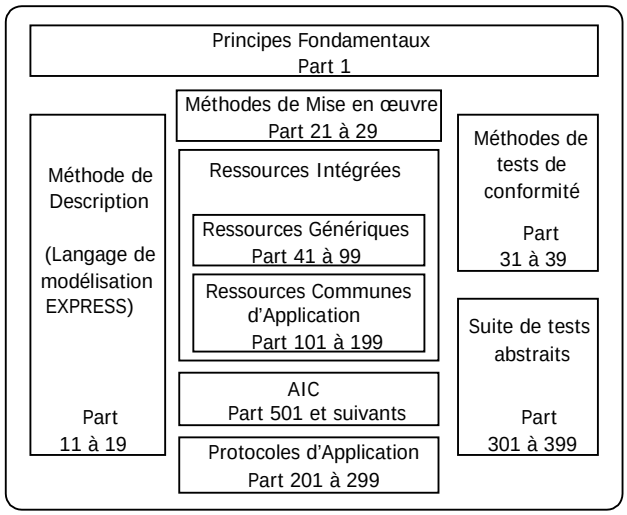


Figure 2.1- Structure de la norme STEP (*Source : [Féru, 98]*)

Elle est par ailleurs basée sur la structure à trois niveaux d'abstraction (externe, conceptuel, interne) préconisée dans le rapport de standardisation des architectures des SGBD [ANSI, 75], comme l'indique la figure ci-dessous (cf. Figure 2.2).

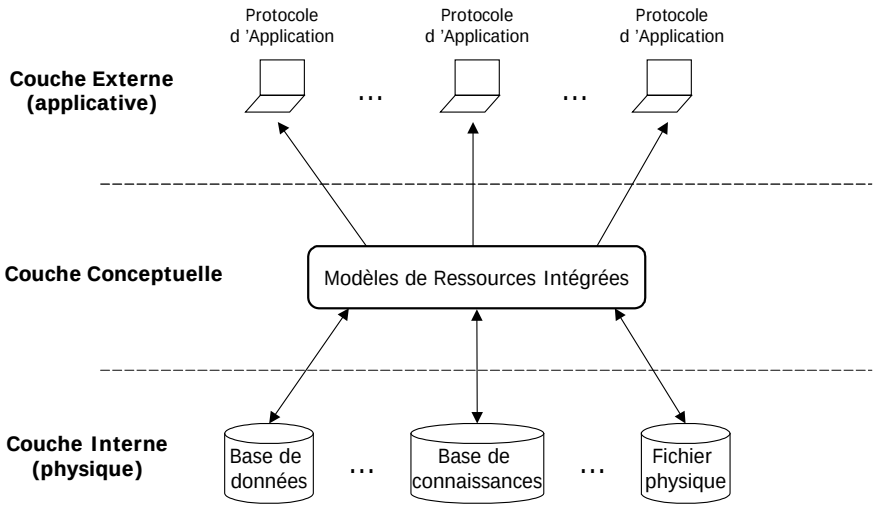


Figure 2.2 - Architecture de STEP (*Source : [Ghodous, 95]*)

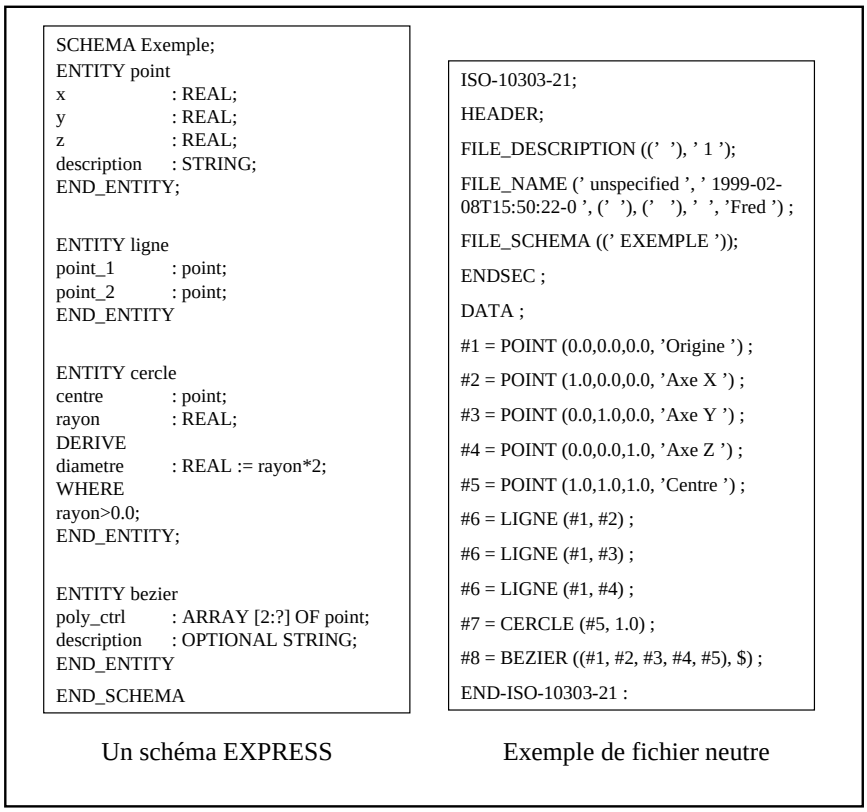
- Le niveau conceptuel est constitué:
  - des *Ressources Intégrées - IR* (part 41 à 99 et 101 à 199) : elles sont composées de Ressources Génériques indépendantes de tout contexte et de Ressources Communes d'Application plus spécifiques à un groupe de domaines d'application. L'ensemble

constitue une bibliothèque de définitions d'entités qui permettent de construire les descriptions des données d'un produit dans un Protocole d'Application.
- des *Application Interpreted Construct - AIC* (part 501 et suivants) : ce sont des librairies de modèles, complémentaires aux Ressources Intégrées et qui sont utilisables par plusieurs Protocoles d'Application.

- Le niveau externe est matérialisé par les *Protocoles d'Application - AP* (part 201 à 299). Ces AP résultent de la sélection de certaines Ressources Intégrées pour des utilisations dans un contexte donné. Chaque AP comprend une expression des besoins informationnels (donc un modèle de données) pour un domaine donné. Un domaine correspond à un type de produit, à certaines phases de son cycle de vie. A ce titre, l'AP 203 se focalise sur la conception 3D de produits mécaniques avec gestion de configuration.

Les modèles informationnels de STEP se localisent dans les niveaux conceptuel et externe, au niveau des Ressources Intégrées, des AIC et des Protocoles d'Application. Ces modèles sont décrits à l'aide des *Méthodes de description* de STEP (part 11 à 19), constitués de langages normalisés pour la représentation de données dont essentiellement le langage de modélisation EXPRESS (norme ISO 10303-11). C'est un langage de modélisation formelle, interprétable informatiquement. Il utilise le formalisme entité-association étendu, en intégrant des concepts pour spécifier la généralisation, l'agrégation et les contraintes. Un des principaux concepts sur lesquels se base ce langage est le concept de schéma. EXPRESS permet la définition de modèles de données modulaires en partitionnant le modèle complet en schémas. Chaque schéma correspond à un domaine d'intérêt particulier caractérisé par un ensemble d'objets. Un modèle de données peut être alors composé d'un ou de plusieurs schémas (cf. Figure 2.3). Une présentation détaillée des concepts de base du langage EXPRESS est proposée par F. Chambolle [Chambolle, 99].

- Le niveau interne est constitué par les *Méthodes de Mise en Œuvre* (part 21 à 29) qui permettent de traduire et de manipuler les données d'un domaine d'application sous forme informatique. Elle concerne la couche physique de l'architecture d'un SGBD. Deux formes de mise en œuvre sont disponibles aujourd'hui : l'échange par fichier neutre et le partage au moyen de requêtes normalisées [Féru, 98].
  - Le fichier neutre STEP est analogue aux fichiers d'échange de données classiques. C'est un fichier ASCII contenant des données. La structure des données est définie en EXPRESS. Le format de ce fichier est défini par la partie 21 de STEP. Un fichier STEP contient une section *header* qui indique entre autres le nom du *schema* utilisé, et une section *data* qui contient des données correspondant au *schema*. La Figure 2.3 présente un exemple de fichier neutre associé à un schéma EXPRESS.
  - Pour le partage de données, le protocole SDAI (partie 22 à 27 de STEP) constitue une interface fonctionnelle - ensemble de routines - entre une application et les données qu'elle manipule. La structure de ces données doit être décrite en EXPRESS. SDAI est indépendant du format de stockage des données. Les parties 22, 23, 24 et 26 de STEP définissent SDAI et sa mise en œuvre avec un langage de programmation.



```
SCHEMA Exemple;
ENTITY point
x               : REAL;
y               : REAL;
z               : REAL;
description     : STRING;
END_ENTITY;

ENTITY ligne
point_1         : point;
point_2         : point;
END_ENTITY

ENTITY cercle
centre          : point;
rayon           : REAL;
DERIVE
diametre        : REAL := rayon*2;
WHERE
rayon>0.0;
END_ENTITY;

ENTITY bezier
poly_ctrl       : ARRAY [2:?] OF point;
description     : OPTIONAL STRING;
END_ENTITY

END_SCHEMA
```

Un schéma EXPRESS

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION ((' '), ' 1 ');
FILE_NAME (' unspecified ', ' 1999-02-
08T15:50:22-0 ', (' '), (' '), ' ', 'Fred ') ;
FILE_SCHEMA ((' EXEMPLE '));
ENDSEC ;
DATA ;
#1 = POINT (0.0,0.0,0.0, 'Origine ') ;
#2 = POINT (1.0,0.0,0.0, 'Axe X ') ;
#3 = POINT (0.0,1.0,0.0, 'Axe Y ') ;
#4 = POINT (0.0,0.0,1.0, 'Axe Z ') ;
#5 = POINT (1.0,1.0,1.0, 'Centre ') ;
#6 = LIGNE (#1, #2) ;
#6 = LIGNE (#1, #3) ;
#6 = LIGNE (#1, #4) ;
#7 = CERCLE (#5, 1.0) ;
#8 = BEZIER ((#1, #2, #3, #4, #5), $) ;
END-ISO-10303-21 :
```

Exemple de fichier neutre

Figure 2.3 - Exemple de schéma EXPRESS et du fichier neutre associé (*Source : [Chambolle, 99]*)

Par ailleurs, une notation graphique est associée au langage EXPRESS. Il s'agit de l'EXPRESS-G qui est basée sur le notation IDEF-1X. Elle supporte la notion de schéma et les liens inter-schémas et à un niveau d'abstraction plus bas, les concepts d'entité, type, attribut, relation et cardinalités. La Figure 2.4 illustre le diagramme EXPRESS-G associé au schéma EXPRESS présenté dans la Figure 2.3.

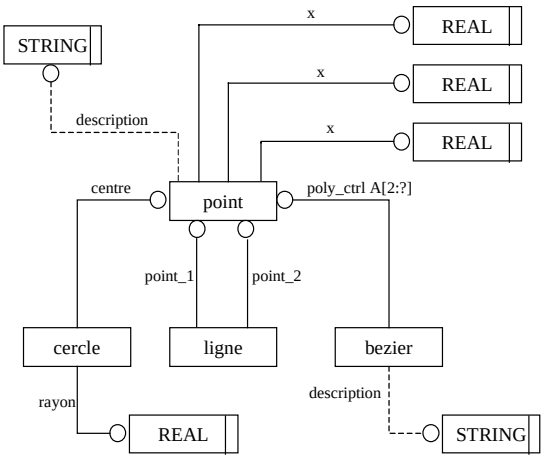


Figure 2.4 - Un exemple de diagramme EXPRESS-G (*Source : [Chambolle, 99]*)

A cela s'ajoute le langage d'instanciation EXPRESS-I (norme ISO 10303-12) qui permet d'affecter des valeurs aux attributs d'un schéma défini en EXPRESS.
Des langages de mise en correspondance ou mapping entre différents schemas de STEP sont également définis, tels que les langages EXPRESS-M et EXPRESS-V.

Un autre langage en marge des langages ci-dessus mentionnés est développé pour la modélisation des processus de fabrication et organisationnels; il s'agit du langage EXPRESS-P qui est une extension d'EXPRESS.

Quant à la définition de modèles de produit conformes à STEP, l'ISO propose une méthodologie pour le développement de protocoles d'application. Cette méthodologie peut être utilisée pour élaborer des modèles normalisés de produit propres à chaque organisation [Kramer, 92]. Une expérimentation de cette approche est proposée par P. Ghodous [Ghodous, 95]. Un système générateur de modèles STEP y est développé.

Cette méthodologie de développement se présente comme suit :

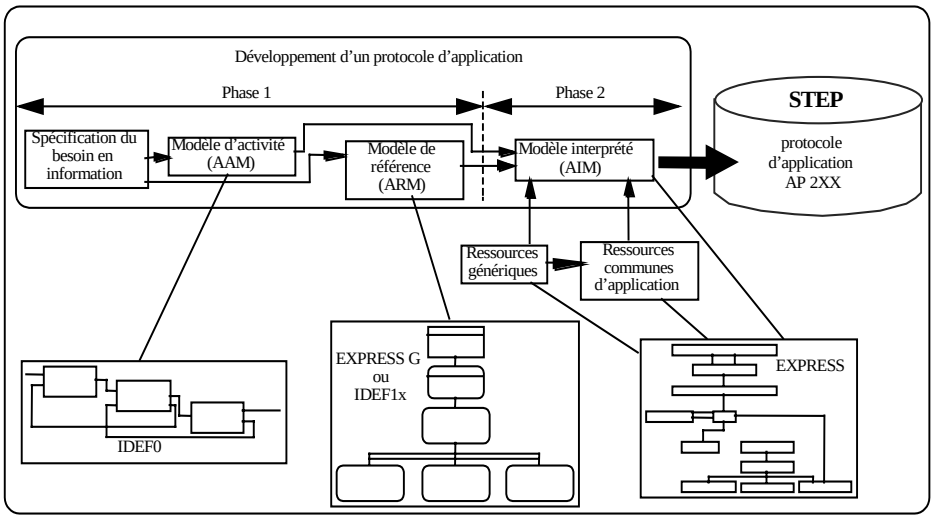


Figure 2.5 - Cycle de développement d'un AP (*Source : [Féru, 98]*)

Le développement d'un protocole d'application est décomposé en deux phases [Féru, 98]:

- La phase de spécification des besoins avec l'implication des utilisateurs (cf. Phase 1 sur la Figure 2.5),
- La phase de conception du modèle interprété de l'application (AIM) avec l'utilisation des Ressources Intégrées communes aux AP (cf. Phase 2 sur la Figure 2.5).

- La première phase consiste à spécifier le besoin en information du point de vue des utilisateurs.
  Tout d'abord, les différents objets d'application (*application objects*) sont structurés en Unités de Fonctionnalité (UoF) et décrits individuellement sous forme textuelle. Les relations et les contraintes portant sur chaque objet d'application sont également spécifiées. Dans un deuxième temps, afin de décrire les flux de données correspondant aux activités associées au domaine de couverture, un modèle d'activité de l'application (AAM) est formalisé (généralement grâce au formalisme SADT). La lecture de l'AAM permet de comprendre l'imbrication des activités : quel objet d'application participe à quelle activité, quel objet en résulte, qui contrôle ou agit sur ces objets d'application, etc.

  Les étapes précédentes permettent d'aboutir à la formalisation la plus avancée du besoin des utilisateurs : le modèle de référence de l'application (ARM). Pour ce faire, différents formalismes sont autorisés. EXPRESS-G est de plus en plus utilisé. Toutefois, IDEF-1x

est également autorisé. Ainsi, aux concepts identifiés lors des étapes précédentes (tels que produit, version de produit, vue métier, ...) sont associés des objets, des attributs ou des relations. Avec la spécification du modèle de référence de l'application (ARM) s'achève le travail de formalisation du besoin des utilisateurs. Ce besoin sera interprété au moyen des Ressources Intégrées (IR) de STEP.

- La deuxième phase consiste à créer le modèle interprété de l'application (AIM). Celle-ci se décompose en deux étapes :
  - La sélection pour chaque objet de l'application des entités normalisées adéquates dans les Ressources Intégrées. Cette mise en correspondance est formalisée sous forme de table à 5 colonnes avec le nom de l'objet de l'application, le nom de l'entité des ressources intégrées correspondantes, l'identification de la ressource intégrée dont est issue l'entité, les contraintes applicables et le chemin (*reference path*) permettant l'instanciation du concept dans l'AIM.
  - L'AIM est le schéma en EXPRESS contenant les entités sélectionnées et spécifiant les contraintes complémentaires afin de correspondre au besoin spécifié dans l'ARM.

  Afin de correspondre aux choix de mise en œuvre et d'utilisation du Protocole d'Application, l'AIM est décomposé en classes de conformité. Chaque classe de conformité correspond à un sous-ensemble de l'AIM.

3.1.3. Vers un choix ...

De l'étude des divers formalismes de modélisation en SI, aussi bien généralistes que spécifiques aux SIP, nous concluons sur l'inadaptation des formalismes spécifiques aux SIP actuellement proposés dans la littérature aux objectifs de notre étude. Le formalisme EXPRESS proposé dans la norme STEP n'est pas adapté au contexte d'ingénierie de SIP proposé. Il ne répond pas aux critères de choix du formalisme de modélisation posés dans le §2. Trois remarques s'imposent à ce sujet :

- le formalisme EXPRESS ne facilite pas à notre avis la communication autour du besoin. Les représentations de STEP sont relativement complexes et peu parlantes.
- les modèles offerts dans EXPRESS ne permettent pas de couvrir l'ensemble des niveaux d'abstraction (statique, dynamique, fonctionnel) et de prendre en compte le système métier.
- EXPRESS ne se base pas sur un formalisme totalement objet, ce qui contraint à spécifier l'aspect comportemental dans le SI (aussi bien le comportement de l'application en terme de traitements que le comportement des objets à l'intérieur du système). La correspondance entre le comportement du SI et le comportement des objets n'est pas évidente.

Ainsi, nous puisons notre recherche dans les formalismes généralistes développés pour les SIG.

Le contexte des SIP implique des besoins différents que ceux présents dans les applications de gestion traditionnelles. Les données gérées sont plus complexes (graphiques, texte, etc.), les modèles de données sont plus riches (structures de données hiérarchisées), à forte sémantique et impliquent une dynamique riche à plusieurs niveaux d'abstraction (instance, classe, modèle). L'approche orientée objet, ayant déjà fait ses preuves dans de précédents travaux de

recherche relatifs au développement d'applications classiques mais également au développement d'applications d'ingénierie[61], répond à ces besoins en offrant des modèles permettant la prise en compte d'objets complexes et l'expression de la dynamique des objets d'un SI. La technologie objet associée (langages OO, SGBDOO) renforce ces avantages en offrant des possibilités de partage et de réutilisation du code des applications.
Une approche objet doit de ce fait permettre d'exprimer la sémantique des systèmes et de leurs concepts, de faciliter la mise en œuvre de mécanismes d'évolution et d'élaborer des environnements adaptables à des besoins particuliers [Rieu, 99c].
Nous nous plaçons délibérément dans un contexte d'ingénierie objet. Un choix qui fut de plus en plus adopté dans les différents travaux des dix dernières années pour la promesse qu'offrent les méthodes objets à favoriser la réactivité et l'évolutivité des applications, la productivité en développement et la réutilisation de composants existants.

S'agit-il alors ici d'adopter une des méthodes objet proposées dans la littérature ?
Intuitivement, le choix porte sur le langage UML de part le fait qu'il constitue une unification des méthodes objets, tirant donc profit des avantages de chacune de ces méthodes à l'origine de l'unification[62] et qu'il constitue un standard pour la modélisation orientée objet. Ce choix est également justifié par plusieurs autres avantages qu'offre UML par rapport à d'autres méthodes objet et par rapport à nos besoins. Nous les décrivons dans ce qui suit :

- UML est fondé sur un métamodèle lui-même décrit en UML. Ce métamodèle donne une vision plus formalisée du langage et définit toutes les possibilités d'extensibilité de celui-ci[63].
- UML se veut un langage non fermé : les éléments de modélisation qu'il propose sont génériques, extensibles et configurables par l'utilisateur. Comme le précise P-A. Muller [Muller, 97], UML ne recherche pas la spécification à outrance : il n'y a pas une représentation graphique pour tous les concepts imaginables ; en cas de besoins particuliers, des précisions peuvent être apportées au moyen de mécanismes d'extension (qui sont les stéréotypes, les étiquettes et les contraintes ; cf. Annexe A) et de commentaires textuels.
- la simplicité et la capacité d'expression visuelle qu'offrent les diagrammes d'UML, et qui permettent de faciliter la communication autour des besoins entre les différents acteurs du SI. UML offre une bonne communication avec les utilisateurs. Les concepts manipulés en UML correspondent à des réalités concrètes pour l'utilisateur. L'écart entre les concepts modélisés et les concepts implantés est réduit par rapport aux méthodes précédentes. Les distorsions entre les attentes des utilisateurs et les résultats sont minimisées. L'approche par les cas d'utilisation remet par exemple l'utilisateur au centre de l'analyse et de la conception.
- la diversité des diagrammes qu'offre UML (au nombre de neuf) et qui permet de présenter plusieurs perspectives ou vues de l'architecture du système à développer :

---

[61] Le projet SHOOD par exemple [Djeraba, 93].

[62] Dans [Kettani, 98] une étude comparative entre UML et les autres méthodes à objet montre la complétude d'UML en terme de couverture des l'ensemble des axes de modélisation et du cycle de développement de SI.

[63] D'ailleurs jusqu'à un temps proche, UML souffrait de manque de support pour la simulation. D. Harel, concepteur des machines à états de même nom, a démontré la capacité d'UML à modéliser des systèmes exécutables et donc UML est lui même exécutable ce qui laisse la possibilité de construction d'outils de simulation et de génération de codes formels [Kettani, 98].

  - Les fonctions du système (de point de vue utilisateur) grâce aux diagrammes de cas d'utilisation. Ces diagrammes offrent un moyen efficace pour obliger les utilisateurs à articuler leurs besoins parce qu'ils les forcent à exprimer la manière dont ils entendent utiliser le système, à préciser quelles informations ils entendent échanger et à décrire ce qui doit être fait pour obtenir le résultat escompté. Par ailleurs, ces cas d'utilisation font profiter le langage UML des avantages de l'approche systémique, telle que dans Merise en s'occupant à la problématique du pourquoi du système par ajout des notions de finalité (raison d'être), de but (formalisation des finalités) et d'objectif (concrétisation des buts).
  - Le comportement des objets mais surtout du SI dans sa globalité grâce aux diagrammes de séquence, de collaboration et d'activités. En effet, si les premières méthodes objet ont su exprimer le comportement d'un objet en représentant son cycle de vie grâce aux notions d'événements et d'états, elles n'offrent pas réellement des moyens efficaces pour exprimer la dynamique d'un SI [Front, 97]. Les trois diagrammes ci-avant cités permettent d'exprimer le comportement dans le SI à différents niveaux d'usage (à un niveau métier pour documenter les cas d'utilisation ou à un niveau informatique pour documenter les objets).
- la capacité du langage de modélisation proposé dans UML à représenter divers phénomènes de l'activité de l'entreprise et ce indépendamment des techniques d'implémentation. Cela va des processus métier, des systèmes d'information jusqu'aux systèmes informatiques et composants logiciels. Dans un contexte SI, UML présente l'avantage de modéliser d'un côté le système d'information organisationnel (processus métier, organisation de l'entreprise), et de l'autre côté le système d'information informatique, support de ce système d'information organisationnel. Il est nécessaire en effet dans l'ingénierie des SI d'avoir une vision générale qui parte du métier, de ses processus et de son organisation avant d'aboutir au système informatique associé. Il utilise par ailleurs des concepts généraux et unifiés pour modéliser aussi bien des métiers que des phénomènes informatiques. En particulier, les concepts de modélisation proposés ne sont pas spécifiques aux métiers ou à l'informatique. Un même modèle peut être utilisé à différentes étapes du processus de développement et à des niveaux d'abstraction différents d'une étape à l'autre.
- la capacité d'UML à modéliser aussi bien le système métier que le système informatique pallie aux faiblesses des méthodes objets qui ne ciblaient que le système informatique, et ce contrairement aux méthodes fonctionnelles telles que Merise où l'accent est également mis sur l'aspect métier. Ceci devra également faciliter le passage des modèles métier aux modèles du système informatique. La représentation des deux systèmes dans des formalismes communs devrait assurer un continuum de transformations[64]. Nous émettons toutefois une réserve sur ce dernier point. D. Rieu [Rieu, 99c] souligne en effet que l'utilisation du paradigme objet favorise les développements "sans couture" mais ne les garantit pas. Aucune démarche n'a été proposée dans les méthodes objets pour guider les transformations inter-modèles[65].

[64] Cet avantage n'est pas toutefois réservé à UML mais commun aux autres méthodes objet qui favorisent le raffinement de spécifications sur la base du même formalisme et ce contrairement aux méthodes classiques (fonctionnelles) où il s'agit par exemple de transformer un diagramme entité/association en un schéma relationnel.

[65] A cet égard, remarquons que les ouvrages dédiés aux méthodes objet proposent rarement une formalisation de leurs démarches de développement.

- UML est doté d'un langage formel OCL (*Object Constraint Langage*) permettant d'exprimer des contraintes supplémentaires sur les modèles graphiques (qui sont souvent insuffisants pour avoir une spécification précise et non ambiguë). OCL peut être utilisé pour spécifier toutes sortes de contraintes, pré et post conditions sur les opérations de classes des différents modèles et conditions invariantes qui doivent être prises en compte dans la modélisation des systèmes. Il a les caractéristiques d'un langage d'expression, d'un langage de modélisation et d'un langage formel.

Il est important enfin de souligner que le choix du langage UML au dépend d'EXPRESS ne rompt pas toutefois la conformité de notre travail avec le cadre normatif de STEP. Les tentatives de plus en plus nombreuses de coupler les modèles de STEP avec d'autres modèles plus puissants (tels que XML) devraient assurer un lien entre les modèles UML et STEP. A ce titre, citons les recherches effectuées dans le cadre du projet européen *XML/EDI Pilot Project* et qui proposent d'ores et déjà une démarche pour convertir[66] les modèles UML en modèles XML [Rivers, 00], lesquels font l'objet d'un certain nombre de projets de recherche pour les coupler avec les modèles de STEP. Remarquons aussi, qu'en terme de *contenu* des modèles de données que nous proposons dans le cadre méthodologique réalisé, nous avons essayé de se conformer au contenu des modèles proposés dans les protocoles de STEP.
Nous décrivons le langage UML ainsi que le langage OCL dans l'Annexe A.

3.2. Réutilisation dans l'ingénierie des SI

3.2.1. La réutilisation

La réutilisation, terme à la mode aujourd'hui, est une idée ancienne qui a de tout temps consisté à ne pas réinventer la roue. A fur et à mesure qu'on élabore des solutions aux problèmes courants, ces solutions sont essayées pour des problèmes similaires. Seuls quelques éléments des solutions sont appliqués avec succès aux nouveaux problèmes. Les anciennes solutions sont typiquement modifiées, combinées et adaptées pour résoudre le nouveau problème. Quand une solution prouvée est utilisée plusieurs fois pour résoudre le même type de problème, elle devient alors acceptée, généralisée et normalisée.
La réutilisation consiste ainsi à construire un système à partir de composants existants préalablement décrits, mis en œuvre, testés et acceptés dans des projets antérieurs. L'objectif est d'éviter de refaire tout à chaque nouveau développement. Il s'agit de composants stables non modifiables mais adaptables à chaque nouveau projet. A ce propos, S. Olcoz [Olcoz, 99] souligne la différence entre ré-ingénierie et ingénierie par réutilisation. En ré-ingénierie, on modifie les sources d'informations existantes pour construire un nouveau système à partir d'un système existant (généralement après une action de rétro-ingénierie pour comprendre comment le système existant a été élaboré). L'ingénierie par réutilisation, par contre, consiste juste à adapter les sources d'informations (qui restent non modifiées) à chaque nouveau projet.

Cette réutilisation lors de l'ingénierie des système s'est souvent manifestée sous deux formes : la réutilisation opportuniste (*opportunistic reuse*), qui survient "accidentellement" et la réutilisation systématique (*systematic reuse*), qui est prévue dans les processus de développement de l'organisation.

[66] via XMI : *XML Metadata Interchange*

En général, toutes les organisations pratiquent la réutilisation opportuniste en récupérant l'information existante d'une manière ad-hoc. Les acteurs gagnent ainsi un peu de temps en modifiant les conceptions existantes afin de les adapter au besoin courant. Cette façon de faire a toutefois des limites. Les conceptions récupérées et modifiées subissent le même processus, coûteux en temps et en argent, qu'exigent les nouvelles conceptions (configuration, test, documentation, etc.). Cette réutilisation non structurée ne réalise que des gains marginaux qui n'ont pas d'impact au delà de la phase de conception dans un projet donné.
De l'autre côté, les organisations qui pratiquent la réutilisation systématique intègrent l'ingénierie pour et par la réutilisation dans un processus bien défini. La réutilisation systématique vise trois objectifs : améliorer la qualité de la conception, diminuer les coûts et réduire les délais de mise sur le marché.

La conception et le développement de systèmes à partir d'éléments réutilisables est devenue ainsi une caractéristique de l'industrialisation dans de nombreuses industries. Ce mode de travail est aujourd'hui éprouvé dans plusieurs industries, notamment celles qui basent leur offre sur la différenciation de produit, tels que l'industrie électronique et la construction automobile.

Dans le secteur de l'informatique, le principe de réutilisation fut évoqué à la fin des années 60 par M. MacIlroy qui proposait une solution à la crise du logiciel basée sur la réutilisation de composants logiciels produits en masse. Les premières applications des principes de réutilisation furent produites en informatique de gestion dans les années 70-80. Elles se manifestaient sous la forme de progiciels paramétrés adaptables à des systèmes d'information spécifiques dans divers domaines : comptabilité, gestion du personnel, etc.. Plus tard, la réutilisation s'est généralisée dans tous les secteurs de la production de logiciels, dans les grandes organisations aussi bien que dans les petites entreprises de développement. Des gains de productivité qui s'élèvent à hauteur de 20-40% sont affichés [Poulin, 93]. Un succès qui est dû à la nature reproductible quasiment instantanément à coût très faible des logiciels. Ce besoin de réutilisation a été d'ailleurs l'une des bases du succès des technologies objet dont l'objectif était de faciliter et d'améliorer les tâches de conception et de codage grâce à la réutilisation de composants.

Cependant, cette réutilisation n'est apparue qu'au niveau des étapes d'implantation par la réutilisation de quelques classes ou fonctions des nombreuses bibliothèques de composants logiciels qui furent fournies avec les environnements de développement. Si les bibliothèques de composants logiciels réutilisables représentent une réelle progression vers la réutilisation maximale dans la construction des applications logicielles, elles n'apportent toutefois que des gains marginaux, qui ne vont pas au delà de l'implantation. Le taux de réutilisation reste en effet limité à 15-20%, le souligne A. Front [Front, 97]. L'amélioration du triptyque coût-délai-qualité ne peut être obtenue qu'avec une application d'une démarche de réutilisation tout au long du cycle de développement, c'est-à-dire depuis l'analyse jusqu'à la maintenance de l'application. Les tendances actuelles visent à exploiter la réutilisation dans les phases amont du développement, à l'analyse et la conception. L'objectif est de favoriser la réutilisation de connaissances acquises lors de l'analyse ou de la conception et de permettre aux concepteurs de travailler à un niveau d'abstraction plus élevé que celui des classes logicielles. Pour

plusieurs spécialistes du domaine, cette tendance constitue un facteur clé de succès pour la mise en œuvre d'une réutilisation plus systématique dans le processus de développement.

Il existe dans la littérature plusieurs publications faisant un état des recherches et des développements en matière de réutilisation [Frakes, 94] [Semmak, 98]. La description que nous faisons dans cette section constitue une synthèse sur les moyens et les méthodes qui nous sont essentiels pour définir une méthode d'ingénierie de SIP par réutilisation. Une telle méthode doit être basée d'abord sur des composants à réutiliser, qui correspondent en fait aux produits de la méthode, et ensuite sur une démarche de développement intégrant ces composants, autrement dit les processus de la méthode.
Ainsi la description faite ici sera abordée selon les deux dimensions : produit et processus. La *dimension produit* (§3.2.2) concerne les composants réutilisables et la *dimension processus* (§3.2.3) concerne les processus de développement à base de composants. Cette dernière dimension est relative, d'une part, à la production de composants réutilisables et d'autre part à l'usage de ces composants pour développer de nouvelles applications.

3.2.2. Dimension "Produit"

1.1.1.9. Différents types de composants

Une variété de composants réutilisables a été proposée dans le domaine du génie logiciel, de l'intelligence artificielle et des systèmes d'information. W. Tracz [Tracz, 94] énumère plus d'une vingtaine de technologies qui ont été proposées pour favoriser la réutilisation, essentiellement dans le développement des logiciels (phase d'implantation).

Les premiers modèles de composants étaient des composants logiciels (c'est-à-dire exécutables). Ils sont essentiellement centrés sur la notion de classe, qui constitue une bonne unité de réutilisation au niveau du codage. Ces composants logiciels permettent en général la réutilisation de fragments de code. De véritables bibliothèques de composants logiciels (*toolkits*) ont été développées. Elles offrent des ensembles de composants logiciels dans lesquels les programmeurs peuvent rechercher un élément et l'intégrer à un programme tout en l'adaptant pour répondre à un besoin. La granularité de ces composants est directement liée aux constructeurs (*constructs*) d'un langage de programmation (la classe, la procédure, la fonction....) et leur niveau d'abstraction est peu élevé puisque ces composants ne fournissent ni le contexte dans lequel on peut les utiliser ni les adaptations que l'on peut leur apporter. Utilisés essentiellement dans la phase de programmation, ces composants restent inadaptés aux activités amont de développement car loin encore des connaissances à réutiliser lors des spécifications et qui sont essentiellement de nature métier.

Les nouveaux modèles de composants proposés s'orientent davantage aux phases de conception et d'analyse et ont pour objectif de capturer les connaissances du domaine. Ils se font toutefois moins nombreux que les composants logiciels. Nous pouvons distinguer trois grands types de composants : les *frameworks*, les *patrons* et les *modèles de domaine*. Nous les présentons dans la suite de cette section.

- **Les Frameworks**

Les frameworks [Johnson, 92] sont de véritables architectures logicielles. Ils définissent de manière générique la structure globale de l'application, les collaborations entre les classes et entre les objets ainsi que les flots de contrôle de l'application.

Ils sont le plus souvent dédiés à un domaine d'application et ils proposent des architectures-types globales pour un domaine, imposant ainsi une architecture particulière à l'application implantée. On parle alors de *frameworks verticaux*. D'autres frameworks, dits *frameworks horizontaux*, proposent des fonctionnalités indépendantes de tout domaine d'application. Ils offrent des fonctionnalités générales par exemple pour le développement d'interfaces graphiques.
Pour illustrer ce concept, nous reprenons l'exemple utilisé dans [Cauvet, 00a] relatif au framework MVC (*Model-View-Controller*) de Smalltalk 80 dédié au développement d'interfaces utilisateur (cf. Figure 2.6).

Le noyau de MVC est composé des trois classes abstraites *Model*, *View* et *Controller* et de nombreuses classes concrètes.

Les trois classes abstraites réalisent les trois principaux concepts du framework et imposent une même structure et un même comportement pour tout système de gestion d'interfaces développé avec le framework :

- Un *modèle* (model) représente la structure interne d'une interface utilisateur. Il mémorise des données et réalise des comportements pour cette interface. Il contient en général des liens vers d'autres objets du domaine de l'application.
- Une *vue* (view) représente l'aspect visuel de l'interface. Elle spécifie la manière dont un modèle est visualisé à l'écran. Elle contient le code pour afficher le modèle.
- Un *contrôleur* (controller) interprète les actions de l'utilisateur sur l'interface et envoie au modèle les messages lui permettant d'effectuer les mises à jour correspondantes.

Le framework contient le mécanisme de dépendance (sous forme d'envoi de messages) entre le modèle et sa (ou ses) vue(s). Selon ce mécanisme, une vue peut accéder aux informations mémorisées dans son modèle afin de se réafficher et un contrôleur peut déléguer de manière appropriée des messages vers son modèle lors d'actions utilisateur.

Les classes concrètes du framework sont essentiellement des sous-classes des classes "View" et "Controller" spécialisées pour certains composants d'interfaces. Ils facilitent la construction d'interfaces plus ou moins spécifiques. C'est ainsi que des sous-classes de "View" réalisent des composants d'interface élémentaire (listes, boutons, champs de saisie, menus, etc.) ou des classes de vues composites.

Figure 2.6 - Un exemple de Framework : le MVC

Les frameworks ont un réel avantage par rapport aux bibliothèques de composants; ils dispensent le développeur de choisir les classes, de fournir les interconnexions, de découvrir quelles méthodes sont disponibles, de trouver celles qui doivent être appelées et dans quel ordre, etc.. Ils cachent ainsi toute cette complexité en offrant un niveau d'abstraction plus élevé.
L'utilisation d'un framework se fait lors des étapes d'analyse, dans le cas de frameworks verticaux ou lors de la conception architecturale dans le cas de frameworks horizontaux [Rieu, 99c].

- **Les Patrons**

Généralement, le terme patron désigne un modèle sur lequel travaillent les artisans pour fabriquer certains objets. C'est la traduction du terme anglais "pattern" qui désigne un modèle schématique ou un modèle simplifié d'une structure [Robert, 93]. Ce concept a été introduit dans le domaine de l'ingénierie du logiciel par K. Beck et W. Cuningham [Beck, 87]. Ils proposaient une première adaptation du concept de patron de conception (design pattern), introduit préalablement par C. Alexander [Alexandre 77] dans le domaine de l'architecture des bâtiments, à la conception et à la programmation orientée objets.

Un patron tel que défini par C. Alexandre "décrit à la fois un problème qui se produit très fréquemment dans un environnement et l'architecture de la solution à ce problème de telle façon qu'on puisse utiliser cette solution des millions de fois sans jamais l'adapter deux fois de la même manière"[67].

Quelques années plus tard, P. Coad [Coad, 92] propose des patrons pour la modélisation conceptuelle des systèmes d'information. Il propose de faciliter l'analyse d'un système en identifiant les besoins selon sept patrons pré-établis. Un patron orienté objet est alors vu comme une abstraction d'un ensemble de classes qui peut être réutilisé encore et encore pour le développement d'applications. Dans les travaux d'E. Gamma, un premier catalogue de patrons spécifiquement destiné à la conception de logiciels est élaboré [Gamma, 95].

Un patron orienté objet offre alors une solution à un problème de modélisation orientée objet, en proposant des artefacts prédéfinis, adaptables à des problèmes similaires et à des technologies différentes d'implantation. Selon que le problème traité apparaît lors de l'analyse des besoins ou de la conception ou de l'implantation du SI, différents types de patrons sont distingués. On parle respectivement de patron d'analyse [Coad, 96] [Fowler, 97], de conception [Gamma, 95] et d'implantation [Coplien, 92]. Les *patrons d'analyse* étant les plus récents dans cette famille de composants. Ils aident le concepteur d'un système dans la construction de modèles objets représentant au mieux les besoins de son système. Les *patrons de conception* identifient, nomment et abstraient des thèmes communs du domaine de la conception orientée objet (ce sont des descriptions d'objets et de classes communicants qui sont personnalisés pour résoudre un problème général de conception, dans un contexte particulier). Les *patrons d'implantation* (ou idiomes) sont spécifiques à un langage et décrivent comment implanter dans un langage particulier certaines caractéristiques généralement absentes (par exemple comment simuler l'héritage multiple en Java) [Rieu, 99a].

Pour illustrer le concept de patron orienté objet, nous présentons dans la Figure 2.7 un patron d'analyse[68].

---

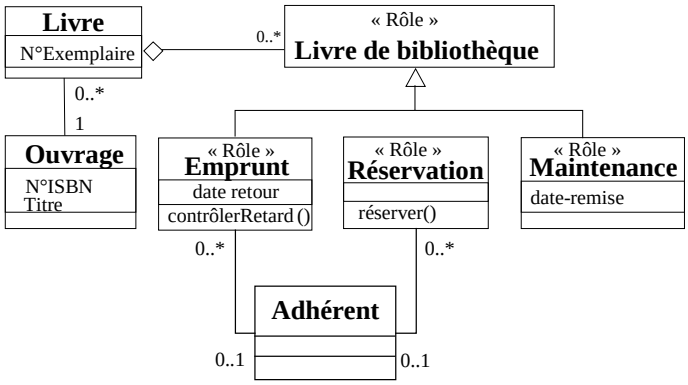
[67] traduction de : Each pattern describes both a problem which occurs over and over again in our environment, and then describe the core solution to that problem, in such a way that you can use this solution million times over, without ever doing it the same way twice.

[68] Afin de simplifier la compréhension du patron, nous utilisons le formalisme de Gamma [Gamma, 95] en ne retenant que les 4 rubriques : nom, intention (problème auquel le patron s'adresse), motivation (un scénario d'application du patron) et solution (exprimée à l'aide de diagrammes UML décrivant en particulier les classes participantes et leurs collaborations).

Nom : patron "Rôle"

Intention : A un moment donné, un objet peut jouer plusieurs rôles. Le patron Rôle donne la possibilité à l'objet d'adhérer et/ou de perdre ces rôles.

Motivation : Le patron Rôle est utilisé lorsqu'un objet a des propriétés variables selon le rôle joué. Un livre a par exemple des propriétés intrinsèques telles que le numéro d'exemplaire, le N° ISBN, le nombre de pages, etc. Dans un contexte particulier, par exemple une bibliothèque, il pourra jouer plusieurs rôles fortement dépendants des processus métiers de la bibliothèque : l'emprunt, la réservation et la maintenance.



Différents rôles d'un livre de bibliothèque

Solution : La classe Acteur est associée à une classe Rôle abstraite. Chaque acteur est lié à des instances des classes Rôles concrètes (Rôle1, Rôle2, etc.), chacune d'elle représentant un de ses rôles. Cette approche a l'avantage d'être plus concise et flexible que l'utilisation de l'héritage multiple.

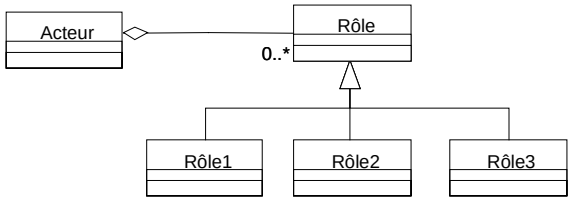


Diagramme de classes du patron Rôle

Figure 2.7 Le patron "Rôle" de P. Coad [Coad, 92]

- **Les Modèles de Domaine**

Les modèles de domaine expriment des connaissances de haut niveau, orientées sur l'expression de problèmes ou de besoins, réutilisables dans le développement de tous les systèmes d'un même champ d'application (appelé domaine) [Semmak, 98]. Un modèle de domaine est suffisamment générique pour constituer un cadre de référence pour le développement de tous les systèmes appartenant à ce domaine.

Différents types de modèle de domaine ont été proposés dans la littérature. Nous citons en particulier les abstractions de domaine proposées par N. Maiden [Maiden, 95], dont le nombre s'élève à 150 abstractions. Leur représentation est basée sur un métamodèle offrant un ensemble de concepts et de liens entre concepts qui permettent d'organiser les abstractions de domaine et de mettre en évidence les éléments discriminant entre des domaines et entre des structures d'un même domaine. Les abstractions de domaine s'expriment ainsi par assemblage de concepts types et de liens entre concepts. Cette approche permet la réutilisation

d'architectures et donc de réduire le travail d'assemblage du concepteur d'application. Cependant, les liens prédéfinis entre composants peuvent réduire la réutilisabilité de cette forme de composants.
Nous illustrons ce concept sur deux abstractions de domaine génériques, proposées par N. Maiden, et applicables à l'allocation de ressources (cf. Figure 2.8). Un concept de transition d'état est utilisé pour exprimer les différences entre le domaine d'allocation de ressources avec restitution (a) tel que la gestion de bibliothèques (un livre emprunté doit être restitué) et sans restitution (b) tel que la gestion de stock de produit (un produit livré n'est pas rendu).

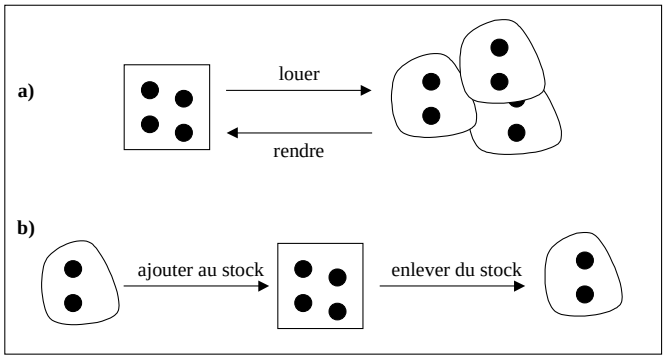


Figure 2.8 - Deux abstractions de domaine génériques (*Source : [Cauvet, 00a]*)

Les abstractions de domaine sont des cas particuliers de modèles de domaine. Ils correspondent à des bibliothèques de composants domaine, au sens de la typologie proposée par F. Semmak [Semmak, 98][69]. Dans cette même catégorie, nous citons aussi les composants domaine proposés par H. Gomaa [Gomaa, 93] modélisant un domaine comme une collection d'objets concourants qui communiquent à l'aide de messages.

1.1.1.10. Vers un choix

Différents critères entrent en jeu pour choisir le modèle de composant réutilisable. Pour déterminer notre choix, nous nous basons sur les critères proposés dans [Rieu, 99c]. Il s'agit essentiellement de la granularité (mesurée en nombre de classes), du degré de généralité (mono-domaines, multi-domaines), de la portée (étape d'ingénierie à laquelle le composant s'adresse) et de la nature de connaissances qu'ils permettent de réutiliser (fragments de produits, fragments de démarche).
En terme de degré de généralité, les frameworks (horizontaux) comme les patrons et les modèles de domaine (bibliothèque de composants domaine) traitent des problèmes qui sont fréquents dans de nombreux domaines d'applications. Ils sont ainsi des composants multi-domaines, garantissant un taux plus élevé de réutilisabilité.
En terme de granularité, les patrons de conception ou d'analyse déjà proposés[70] proposent des diagrammes constitués de deux à six classes. Les modèles de domaine ont une granularité similaire aux patrons. Au contraire, la plupart des frameworks proposent une architecture

[69] Trois formes de modèles de domaine sont distinguées : les *référentiels de connaissances* sur le domaine (qui expriment de manière structurée le vocabulaire du domaine, son environnement, ses fonctions, etc.); les *bibliothèques de composants* domaine réutilisables pour le développement de systèmes et les *spécifications des processus* de développement (englobant à la fois la connaissance de domaine et le processus qui permet à développeur d'utiliser la connaissance de domaine pour le développement d'un système particulier).
[70] en se référant par exemple aux patrons de Gamma et de Coad.

logicielle complète. Ils sont constitués d'une très grande variété de classes. Compte tenu de leur taille, les frameworks peuvent devenir trop rigides. Ainsi, la granularité d'un patron ou d'un modèle de domaine fournit une unité de raisonnement très modulaire. De plus, chaque patron ou modèle de domaine existe pour répondre à un problème type. De ce point de vue, les patrons et les modèles de domaine présentent un avantage par rapport aux frameworks.
En terme de portée des composants, les frameworks peuvent être utilisés lors des étapes d'analyse (frameworks verticaux) mais aussi lors de la conception architecturale (frameworks horizontaux). Les patrons s'adressent à l'analyse, à la conception et à l'implantation. Quant aux modèles de domaine, ils s'adressent uniquement à la phase d'analyse.
Enfin, en terme de nature de la connaissance capitalisée, les patrons permettent la réutilisation de fragment de modèles mais également de fragments de démarche de modélisation, contrairement aux autres types de composants qui sont uniquement orientés "modèle". En effet, un patron capitalise un modèle général répondant à un problème spécifique (un fragment de modèle) mais également la manière d'obtenir ou d'adapter ce modèle (fragment de démarche). Les patrons constituent une base de savoir et savoir-faire. Ils constituent ainsi un guide d'ingénierie en organisant hiérarchiquement et fonctionnellement les problèmes et la manière de les résoudre.

Compte tenu de éléments de comparaison ci-dessus énumérés, la réutilisation de patrons nous semble la forme de réutilisation la plus adaptée à l'ingénierie des SIP. Elle peut être utilisée dans toutes les étapes du cycle de développement d'un SIP (analyse, conception, implantation). Les patrons utilisés dans l'étape d'analyse des besoins fournissent des solutions à des problèmes de domaine alors que ceux utilisés par exemple au moment de l'implantation fournissent des solutions à des problèmes techniques dans le contexte de SGDT particuliers. Par ailleurs, l'intégration dans un même patron d'un problème type et d'une solution constitue une aide à la recherche et à l'intégration de composants.

Dans la section suivante (§ 1.1.1.11), nous développons la réutilisation de patrons et montrons ses avantages pour le domaine d'ingénierie des SIP.

1.1.1.11. Les Patrons

Cette section est destinée à développer le concept de patron. Quelques exemples serviront à illustrer ce concept.

- **Les patrons en général**

Dans sa généralité, le terme patron a été utilisé dans plusieurs domaines, bien avant son utilisation dans le domaine de l'informatique. On parle de patron en confection, en décoration, en manufacture, en aviation, en linguistique, etc.

Le meilleur exemple pour illustrer ce concept est celui des patrons de confection vestimentaire (cf. Figure 2.9). Un patron est un modèle de papier ou de toile préparé sur un mannequin et utilisé pour créer des vêtements pareils. Selon les mesures des personnes, ce patron est adapté à chaque nouvelle confection par ajustement des dimensions du patron.

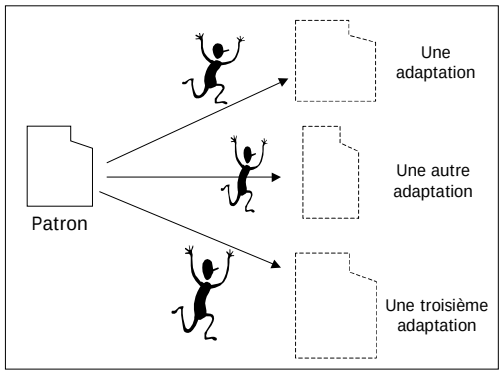


Figure 2.9 - les patrons en confection

Ainsi un patron capitalise un problème récurrent d'un domaine et sa solution de manière à faciliter la réutilisation de cette solution lors d'une nouvelle occurrence du problème [Rieu, 99a]. Il constitue donc une base de savoir et de savoir-faire qui :

- permet d'identifier le *problème* à résoudre (dans l'exemple : fabriquer un vêtement d'une forme particulière),
- propose une *solution* correcte et si possible consensuelle pour y répondre (dans l'exemple : un modèle papier permettant de couper le tissu qui réalise la forme désirée),
- offre les moyens d'adapter cette solution à un *contexte* spécifique (dans l'exemple : ajuster le modèle papier par augmentation ou diminution des dimensions selon les mesures de la personne considérée).

Nous retrouvons cette même triptyque (problème/solution/contexte) dans la définition donnée par D. Lea [Lea, 99] où un patron est généralement défini comme une solution à un problème dans un contexte.
Cette notion de patron a été particulièrement développée par C. Alexander [Alexander, 77], dans les années 1970, pour proposer les patrons dits de conception (*design patterns*) dans le domaine de l'architecture des bâtiments. Partant de l'idée que les architectures, au fil des siècles, avaient été adaptées les unes aux autres et à leur environnement, C. Alexander propose un *langage de patrons*, formé d'un ensemble de patrons (253 patrons) dont chacun décrit comment résoudre un problème particulier de la construction d'une maison (comment disposer les pièces dans la maison, quel matériel utiliser pour la construction, comment décorer les pièces, comment installer l'électricité, etc.). Nous illustrons ci-dessous un de ces patrons, fréquemment cité : "*a place to wait*".

Quelle que soit la raison pour laquelle des personnes attendent (salle d'attente d'un médecin, embarquement dans un avion, rendez-vous d'affaires, etc.), il est inévitable que ces gens tournent en rond et perdent leur temps à ne rien faire. Ils ne peuvent pas mettre à profit ce temps, parce que l'attente n'est pas prévisible et qu'ils doivent attendre sur le lieu même de leur rendez-vous. Tant qu'ils ne sauront pas exactement quand leur tour viendra, ils ne pourront aller se promener ou même s'asseoir à l'extérieur. C'est pourquoi, dans les endroits où les gens sont susceptibles d'attendre, il faut créer une situation pour rendre cette attente positive. Associer à cette attente une autre activité (journaux, café, télévision) afin que cette pièce ne devienne plus seulement un lieu d'attente mais un lieu d'activité, ou à l'inverse transformer celle-ci en un lieu de relaxation, de rêverie et de repos.

Figure 2.10 - Patron "*A place to Wait*" de C. Alexander

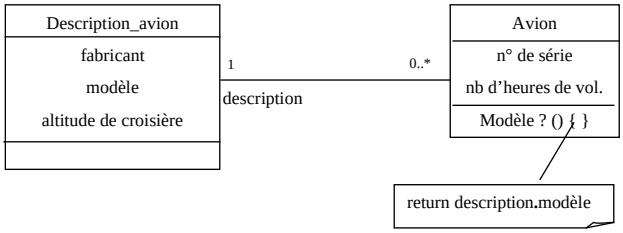
- **Les Patrons orientés objet**

Les premiers patrons orientés objet ont été présentés par K.Beck et W. Cunningham [Beck, 87]. P. Coad [Coad, 92] définit un patron orienté objet comme "une abstraction d'un doublet, triplet ou autre petit groupe de classes qui peut être utile encore et encore dans tout développement orienté objet". Nous avons illustré dans le §1.1.1.9 un premier exemple de patron orienté (patron d'analyse "Rôle" de P. Coad). Nous donnons ici deux exemples de patrons de conception[71].

Nom : patron "item-description"

Intention : ce patron est utilisé lorsque certaines valeurs d'attributs peuvent s'appliquer à plus d'un objet dans une classe. Il permet la gestion de données de différents niveaux.

Motivation : l'objet "avion" connaît son propre n° de série (par exemple N123ABC) et le nombre d'heures de vol (par exemple 5000 heures). Il connaît également un unique objet "description_avion". Un objet "description_avion" connaît ses propres fabricant (par exemple Boeing), modèle (par exemple 747-400) et altitude de croisière (par exemple 8333 miles). Il connaît également un certain nombre d'objets "avion" qui dépendent de cette information.



Un objet "description_avion" détient toutes les propriétés communes aux objets "avion", telles que le fabricant, le modèle, etc.. Ces propriétés partagées par tous les objets "avion" sont décrites dans le niveau "description_avion" et ne sont pas détenues dans les objets "avion". Par ailleurs, une objet "avion" peut consulter certaines de ces propriétés. Les propriétés du niveau "description_avion" restent visibles au niveau "avion" en définissant des méthodes de consultation permettant de propager la valeur d'un attribut à travers les associations. Prenons l'exemple de la propriété *modèle* : quand on demande à un "avion" son modèle, il exécute la méthode *Modèle ? ( )*, définie dans sa classe "Avion". Cette méthode (pseudo-code return description.modèle) consiste à retourner la valeur de l'attribut *modèle* dans la *description* de l'avion, c'est-à-dire dans "description_avion", à l'objet demandeur, c'est-à-dire l'avion, (à travers le rôle *description* de l'association qui lie un avion à sa description).

Solution : le patron "Item-description" propose deux classes : "item" et "item-description". La classe "item-description" détient les valeurs des attributs qui peuvent s'appliquer à plus d'un objet "item". La classe "item" détient les valeurs de ses propres attributs et diverse méthodes de consultation pour accéder aux valeurs des attributs de la classe "item-description".

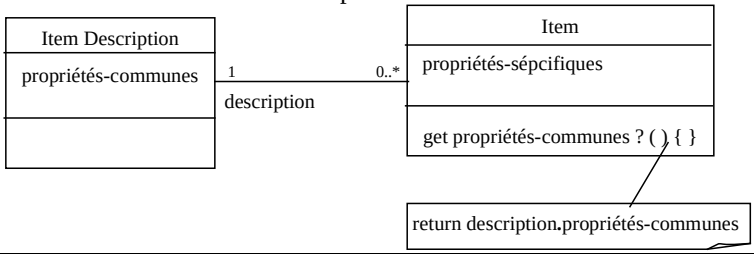


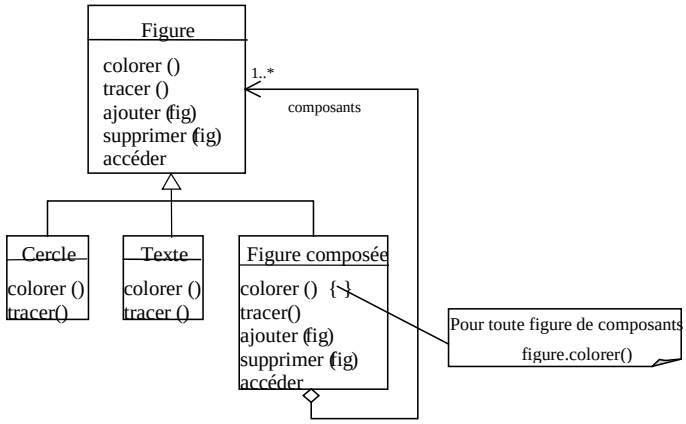
Figure 2.11 - Patron de conception "item-description" [Coad, 92]

[71] Il est à noter que les solutions que nous proposons ici pour les exemples de patrons présentés (patron "Rôle" et patron "Composite") ont parfois été complétées ou au contraire simplifiées.

Nom : patron Composite

Intention : ce patron permet de gérer une composition récursive d'objets. Il définit des hiérarchies de classes d'objets simples et d'objets composites et facilite l'ajout des nouveaux composants.

Motivation : Les éditeurs graphiques autorisent l'élaboration de figures composites à partir de figures simples et prédéfinies et ceci de manière récursive. Une solution consiste à définir une classe pour gérer les figures complexes et une classe pour gérer les figures élémentaires (texte, cercle, etc.). Dans ce cas, les objets simples sont traités différemment des objets composites ce qui alourdit les applications.



Une classe abstraite (notée *Figure*) représente à la fois les objets composites et les objets simples (ou feuilles). La classe *Figure* détient des opérations primitives telles que *tracer* ou *colorer* mais aussi les opérations de gestion des composants (*accéder, supprimer, ajouter un composant*).

Les sous-classes (*Cercle, Texte*, etc.) implantent les primitives graphiques telles que *colorer* ou *tracer* pour colorer et tracer un cercle, un texte, etc. La classe *FigureComposée* réalise elle aussi les opérations *colorer* et *tracer* par des appels récursifs aux opérations de ses composants : pour colorier une figure composée, il faut colorier toutes les figures (simples ou composées) qui la composent.

Solution : le patron composite propose 4 classes : composant, composite, feuille et client.

- *Composant* : définit l'interface commune des objets feuille et composite.
- *Feuille* : définit le comportement des objets feuilles.
- *Composite* : définit le comportement des objet composites et implante les opérations de gestion des composants.
- *Client* : manipule les objets feuilles et composites à travers l'interface Composant.

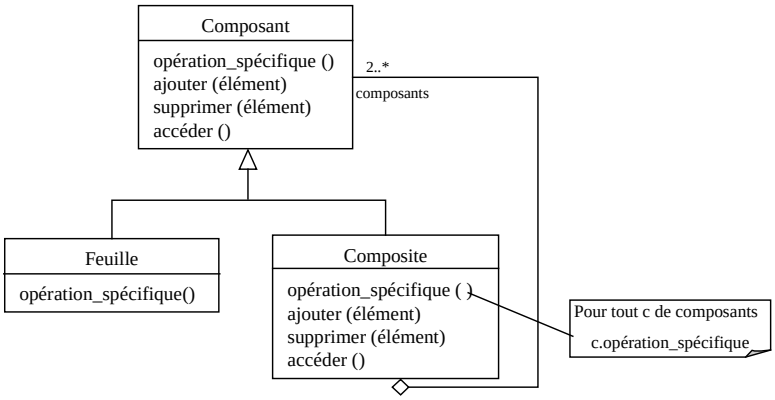


Figure 2.12 - Patron de conception "Composite" [Gamma, 95]

- **Les patrons en ingénierie des SI**

Les patrons permettent au concepteur du SI de *construire des modèles* représentant au mieux *les besoins du SI* considéré. Ils capitalisent ainsi d'un coté les besoins du SI et de l'autre coté les moyens pour spécifier ces besoins. Deux types de connaissances sont mises en jeu dans les patrons : les connaissances du domaine, détenues par les experts et les utilisateurs du SI (les besoins ; les problèmes) et les connaissances de développement, détenues par les concepteurs de SI pour spécifier les besoins (les techniques de spécification ; les solutions).

- **Les connaissances du domaine** sont des connaissances métier, détenues par les experts (les acteurs) du métier auquel s'adresse le SI à concevoir. Ces connaissances métier correspondent aux différents concepts manipulés dans le SI. Il s'agit alors de savoir et de savoir-faire du métier. Les savoir correspondent aux éléments manipulés dans le métier, tels que par exemple les clients et les comptes bancaires dans le cadre d'un SI pour la gestion bancaire ou encore les documents et les nomenclatures de produit dans le contexte de SIP. Il s'agit donc des produits du métier considéré : on parle dans ce cas de ***patrons produit***. Les savoir-faire correspondent aux manières de manipulation des éléments dans le métier considéré. Ainsi, dans le cadre du SI pour la gestion bancaire, les patrons peuvent capitaliser les procédures d'ouverture d'un compte bancaire ou de tout autre type d'opération bancaire, etc.. Dans le cas de SIP, il peut s'agir des processus organisationnels pour définir ou modifier une nomenclature de produit. Il s'agit donc des processus du métier. On parle dans ce cas de ***patrons processus***.
- **Les connaissances de développement** sont des connaissances informatiques, détenues par les concepteurs de SI. Ces connaissances correspondent à des techniques de spécification et d'implantation de systèmes d'information. Il s'agit alors de savoir et de savoir-faire d'ingénierie de SI. Un patron peut être utilisé pour capitaliser un modèle général répondant à un problème spécifique telle que par exemple le modèle pour représenter, dans un formalisme orienté objet, des compositions récursives d'objets. Le patron capitalise ainsi des fragments de modèles (donc des fragments de produits de développement). Il est à ce titre un ***patron modèle***. Le patron capitalise également des savoir-faire de modélisation, c'est-à-dire les manières pour élaborer les modèles, telles que par exemple la manière de construire un modèle de composition récursive d'objets selon différents cas particuliers de composition (composants optionnels, contraintes d'incompatibilité entre composants). Le patron permet alors d'organiser hiérarchiquement et fonctionnellement les problèmes et les manière de les résoudre. Le patron capitalise ainsi des fragments de démarches (donc des fragments de processus de développement). Il est à ce titre un ***patron démarche***.

En résumé, les patrons sont des composants qui capitalisent les connaissances relatives aux produits ou aux processus du domaine auxquels ils s'adressent. Ils capitalisent par ailleurs des fragments de modèles pour représenter ces connaissances mais également des fragments de démarches pour élaborer ces modèles (cf. Tableau 2.1).
La technologie des patrons favorise ainsi, en plus de la capitalisation et de la réutilisation, l'intégration des connaissances de domaine et des connaissances de développement. Un patron permet en effet de spécifier des *problèmes* liés aux besoins des utilisateurs du domaine (connaissances domaine) à l'aide de *solutions* de développement adaptées (connaissances de développement).

		Connaissances de Développement (solutions)	
		Modèle	**Démarche**
Connaissances de Domaine (problèmes)	**Produit**	Modèle pour représenter les produits	Méthode pour modéliser les produits
	Processus	Modèle pour représenter les processus	Méthode pour modéliser les processus

Tableau 2.1 - Typologie des connaissances capitalisées dans les patrons en ingénierie des SI

La section suivante (§3.2.3) a parmi ses objectifs de passer en revue les démarches proposées dans la littérature pour l'ingénierie des composants. A la lumière de cette revue, nous déterminons dans le chapitre suivant (chapitre III) la démarche retenue pour identifier les patrons pour le domaine des SIP.

3.2.3. Dimension "Processus"

Si les propositions concernant les types de composants réutilisables sont nombreuses, la recherche sur le processus de développement à base de composants reste peu abordée. Il existe aujourd'hui un réel besoin de rendre plus systématique la réutilisation en ingénierie des SI. Ce besoin nécessite d'abord de *produire* les composants à réutiliser et ensuite de *réutiliser* ces composants d'une façon systématique.
Ces deux activités impliquent deux nouveaux processus complémentaires, connus dans la littérature sous le vocable de "processus *pour* réutilisation" et "processus *par* réutilisation"[72] :

- Le processus *pour* réutilisation concerne la production (ou l'ingénierie) de composants. Il consiste à identifier, spécifier et organiser des composants réutilisables.
- Le processus *par* réutilisation concerne l'ingénierie de SI par réutilisation de composants. Il consiste en la recherche, la sélection, l'adaptation et l'intégration de composants pour développer des SI. Ce processus implique une re-formulation du processus classique d'ingénierie des SI et des outils de développement en intégrant de manière systématique la réutilisation de composants.

L'objet de la présente section est d'étudier les principales propositions autour de ces deux processus. Cette étude a pour objectif est de définir une technique d'identification des patrons. Un accent est donc mis sur le premier processus. L'étude du second processus sert pour positionner ultérieurement le travail réalisé dans son cadre global d'ingénierie de SIP.

1.1.1.12. Ingénierie de composants pour la réutilisation

Comme le précise D. Rieu [Rieu, 99c], il n'existe pas aujourd'hui de méthodes spécifiquement dédiées à l'ingénierie de composants mais on dispose désormais d'un ensemble de techniques allant dans ce sens. Un recensement et une classification des approches actuelles d'ingénierie de composants sont proposés dans [Semmak, 98] et [Cauvet, 00a]. La présentation qui suit est une compilation de ces travaux.

[72] Les concepts de "Design For reuse" et de "Design By reuse" ont été introduits dans plusieurs travaux. Nous citons en particulier la méthode FODA pour l'ingénierie de domaine [Kang, 90].

L'ingénierie des composants concerne deux problèmes essentiels : l'identification des composants puis leur conception.

- L'identification a pour objectif d'identifier des éléments susceptibles de contenir des éléments réutilisables. L'identification de composants réutilisables constitue désormais un problème de recherche essentiel. Deux approches (qui peuvent être complémentaires) sont distinguées dans la littérature : l'approche de *rétro-ingénierie* et l'approche d'*ingénierie de domaine*. Ces deux approches peuvent être complémentaires. Nous présenterons ultérieurement ces deux approches.
- La conception a pour objectif de spécifier et d'organiser des composants réutilisables. En terme de spécification, les techniques doivent favoriser la réutilisabilité des composants. Pour cela, trois principes clés doivent être respectés : l'abstraction, la généricité et l'héritage. En terme d'organisation, ces techniques doivent favoriser la recherche et la sélection de composants devant répondre à un problème de développement particulier. Les techniques de spécification et d'organisation sont par ailleurs nombreuses et variées. Elles dépendent du type de composant considéré. Les patrons font désormais partie de l'approche de représentation orientée problème. Cette approche fait partie des quelques approches qui respectent pleinement le principe d'abstraction en distinguant entre la spécification du composant et sa réalisation. Dans ce type d'approche, la spécification du composant est centrée sur la description d'un problème et la réalisation du composant correspond à la solution.

- *Approche par Rétro-ingénierie :*

Cette approche vise à abstraire des éléments réutilisables à partir de l'analyse de produits existants. Ces produits peuvent être des logiciels, des schémas conceptuels, etc. Cette approche est basée sur l'utilisation de comparaisons permettant de rechercher les similarités et les différences entre produits afin de dégager les éléments réutilisables. Trois approches sont distinguées [Semmak, 98], [Cauvet, 00a] :

- L'analyse des similarités, basée sur la comparaison d'éléments provenant de différents produits d'un même domaine. Elle met en œuvre deux techniques :
  - des techniques de recherche de similarités entre éléments qui permettent de comparer et de hiérarchiser les éléments, en s'appuyant sur des métriques utilisées dans le domaine des méthodes d'analyse de données.
  - des techniques de généralisation qui visent à définir des composants génériques à partir d'éléments similaires.
- La recherche d'analogies qui, comme l'approche d'analyse de similarités, vise à identifier des éléments réutilisables par comparaison des éléments communs à plusieurs systèmes. La différence essentielle réside dans le type des éléments communs examinés. Alors que l'analyse des similarités évalue la similitude syntaxique des éléments (les attributs des éléments), la recherche d'analogies évalue plutôt les similitudes sémantiques entre les éléments (les liens entre éléments ; les structures relationnelles) [Rieu, 99c]. Ainsi dans la recherche d'analogies, deux structures sont déclarées similaires si elles sont deux instances d'une même structure générique.
- Les modèles de réutilisation qui sont des ensembles d'attributs (attributs de réutilisation) permettant de mesurer la réutilisabilité d'un élément. Ce type de modèle fournit aussi pour

les attributs les valeurs ou les plages de valeurs acceptables pour qualifier un élément de composant candidat à la réutilisation. Cette approche favorise l'automatisation du processus d'analyse des produits à partir duquel les composants sont extraits.

- *Approche par ingénierie de domaine*

L'ingénierie de domaine consiste d'une part à acquérir les connaissances sur un champ d'application, à les structurer et à les abstraire et d'autre part à spécifier et réaliser un système qui en permette l'utilisation dans le développement de tous les systèmes appartenant à ce champ d'application [Semmak, 98].
Dans un contexte de réutilisation, l'ingénierie de domaine a pour but donc de produire un modèle de domaine[73] suffisamment générique pour constituer un cadre de référence pour le développement de tous les systèmes d'un domaine donné. Elle constitue une technique d'identification de connaissances génériques sur un champ d'application dans la perspective de les réutiliser.
Deux stratégies de mise en œuvre de l'ingénierie de domaine sont identifiées dans [Semmak, 98], selon la manière avec laquelle un domaine est appréhendé :

- une *stratégie descendante* où l'ingénierie de domaine est centrée directement sur la modélisation des problèmes et des décisions qui conduisent le processus de résolution de ces problèmes. Un domaine est vu comme une classe de problèmes pour lesquels différentes solutions sont envisageables. Elle met en avant la définition de problèmes pour lesquels plusieurs systèmes d'information différents peuvent être décrits. Les modèles de domaine ainsi obtenus sont orientés problèmes. Cette stratégie est qualifiée d'*approche dynamique* dans [Cauvet, 00a].
- une *stratégie ascendante* où l'ingénierie de domaine utilise et analyse des familles de systèmes existants pour produire les objets et les opérations communs qui caractérisent ces systèmes. En partant d'une famille de systèmes de fonctionnalités similaires, l'objectif est de produire un modèle de domaine exprimant les ressemblances et les différences entre les systèmes (un référentiel de domaine). Un domaine est alors vu comme un champ d'application pour lequel des systèmes d'information existent déjà. Contrairement à l'approche descendante où le modèle de domaine obtenu est orienté problèmes, dans l'approche ascendante le modèle de domaine obtenu est orienté solutions. Cette stratégie est qualifiée d'*approche statique* dans [Cauvet, 00a].

Comme l'ingénierie des systèmes d'information, l'ingénierie de domaine est organisée selon des méthodes. Dans un contexte de réutilisation, F. Semmak [Semmak, 98] décrit le processus d'ingénierie de domaine comme un processus composé de trois étapes:

- *l'analyse* : centrée sur la l'acquisition et la description de connaissances de domaine. Elle peut aller de la définition des frontières d'un domaine jusqu'à la conceptualisation détaillée des connaissances du domaine. Une des phases ultimes de cette étape est la classification des connaissances qui consiste à organiser les connaissances, à travers des hiérarchies, afin de mettre en évidence les différences et les ressemblances entre toutes les applications du domaine. A l'issue de cette étape, on obtient ce qu'on appelle un *référentiel de domaine*. Il contient des taxonomies d'objets, des modèles de contexte, des modèles fonctionnels, des cadres terminologiques, etc.

[73] Voir définition dans §1.1.1.9

- *la conception* : le référentiel de domaine ainsi obtenu présente peu d'intérêt pour la réutilisation lors de la conception d'un SI. Cette étape a pour objectif de structurer la connaissance sous forme de composants et de définir les mécanismes de recherche, d'accès et de sélection de ces composants. Elle produit l'*architecture* du système de réutilisation (celui qui sera exploité pour développer divers SI par réutilisation). Cette phase reste peu formalisée dans les travaux proposés et peu supportée par les outils.
- *l'implantation* : elle consiste à implanter le système de réutilisation en fonction d'une technologie. Il s'agit en particulier de créer *un environnement* permettant la recherche et l'accès automatique des composants réutilisables.

Selon la stratégie de mise en œuvre adoptée (ascendante ou descendante), des techniques de rétro-ingénierie (cas de la stratégie ascendante) ou des techniques traditionnelles de collecte d'informations auprès des experts du domaine (cas de la stratégie descendante) peuvent être utilisées pour capturer la connaissance de domaine. Les approches de rétro-ingénierie et d'ingénierie de domaine peuvent ainsi être utilisées de manière complémentaire. Pour la modélisation, ce sont principalement des techniques de structuration qui sont utilisées, les mêmes que celles utilisées en ingénierie des systèmes d'information.

Dans la littérature, plusieurs méthodes d'ingénierie de domaine ont été proposées. Nous citons en particulier la méthode FODA (*Feature Oriented Domain Analysis*) [Kang, 90]. Une étude comparative des principales méthodes est proposée dans [Semmak, 98].

Dans le chapitre suivant (chapitre III), nous présentons la démarche proposée pour l'ingénierie de patrons, en se basant essentiellement sur l'ingénierie de domaine.

1.1.1.13. Ingénierie de SI par réutilisation

L'ingénierie de systèmes par réutilisation a pour objectif de mettre à la disposition des développeurs de SI des méthodes et des environnements adaptés au développement d'applications par réutilisation, dont les principales fonctionnalités sont :

- la gestion des bibliothèques de composants,
- la spécification de systèmes à partir de composants,
- l'implantation de systèmes par génération de code.

Des travaux de plus en plus nombreux s'inscrivent dans ce cadre. Nous citons en particulier le projet européen Esprit : REBOOT (*REuse Based on Object Oriented Techniques*) qui propose un modèle de processus d'ingénierie intégrant les activités de réutilisation à toutes les phases du cycle de développement d'application ainsi qu'un environnement pour gérer une base de composants réutilisables (de natures variées) en support à ces modèles de processus [Morel, 93]. Nous citons également la méthode Catalysis, basée sur des patrons pour la description du processus de développement [D'Souza, 98].

Dans une perspective de réutilisation à base de patrons, quelques ateliers commercialisés tels que Objecteering de la société Softeam[74] commencent à intégrer l'usage des patrons, notamment les patrons de conception de Gamma.

[74] Site internet : http://www.objecteering.com/us/b_obj.htm

D. Rieu [Rieu, 99c] définit les fonctionnalités attendues d'un environnement de développement par réutilisation de patrons comme suit :

- la gestion de catalogues de patrons : ceci suppose l'organisation des catalogues de patrons par des relations inter-patrons afin d'en faciliter leur sélection. Dans [Rieu, 99d], un certain nombre de relations est mis en évidence. Ces relations seront présentées dans le chapitre VI.
- la spécification de systèmes à partir de patrons : cela nécessite de disposer d'opérateurs qui permettent la recherche de patrons qui répondent à un problème particulier d'analyse ou de conception. Ensuite, une fois les patrons sélectionnés, ils s'agit de permettre leur instanciation en les adaptant au contexte applicatif puis de permettre l'intégration des instances de patrons pour générer une spécification complète du système.
- la génération de code : l'instanciation de patrons offre des spécifications qui peuvent être soit semi-formelles (patrons de conception) soit formelles (patrons d'implantation). Plusieurs approches de génération de code à partir de patrons ont été proposées. Nous citons en particulier les travaux de G. Sunyé [Sunyé, 99] pour l'implantation de patrons de conception.

Dans le cadre des travaux de l'équipe Sigma du laboratoire LSR (IMAG-INPG), un prototype de recherche basé sur le concept de patrons a été développé[75]. Ce prototype nous a servi à tester la faisabilité logicielle du cadre méthodologique réalisé, et que nous décrirons dans le chapitre VII.

4. Conclusion

Le cadre méthodologique proposé doit reposer, comme nous l'avons décrit dans le chapitre précédent sur :

- un ensemble de modèles représentants les éléments du SIP à différents niveaux d'abstraction,
- un formalisme adéquat supportant les éléments gérés dans le SIP, assurant un continuum de transformation des modèles et favorisant le dialogue entre les acteurs,
- une démarche contextuelle pour l'ingénierie de SIP,
- la réutilisation de composants pour favoriser les spécifications par écart.

En terme de formalisme de modélisation, notre choix s'est focalisé sur le langage de modélisation UML. En terme de réutilisation, le choix a porté sur une approche de réutilisation à base de patrons. Un patron capitalise des fragments de modèles mais également des fragments de démarches d'ingénierie. De ce fait, les deux autres éléments de base du cadre méthodologique proposé (modèle et démarche) se trouvent supportés par les patrons.
La démarche globale d'ingénierie du SIP sera donc un "assemblage" de fragments de démarche, de même que les modèles du SIP seront obtenus par assemblage de fragments de modèles, et ce par réutilisation de divers patrons.

[75] AGAP : Atelier de Gestion et d'Application de Patterns.

Par ailleurs l'approche à base de patrons favorise une démarche contextuelle puisque dans un patron, il est explicité le problème à résoudre, le contexte dans lequel le problème est résolu et la manière de résoudre ce problème. Le concepteur choisit ainsi à chaque étape du processus d'ingénierie le fragment de modèle et le fragment de démarche associé les plus convenables au contexte dans lequel il se situe.

Néanmoins, un effort reste à faire pour pallier à certaines lacunes dans les techniques que nous avons choisies :

- En terme de modélisation et malgré la richesse du langage UML et sa capacité à favoriser un continuum de transformations, il n'assure pas ce continuum. Ainsi, nous nous efforçons dans ce travail de *définir une démarche de transformation entre les différents diagrammes d'UML à différents niveaux d'abstraction*, qui assure un continuum.
- En terme de processus de réutilisation, il s'agit de *définir une démarche d'identification de patrons*, en s'inspirant des démarches générales d'ingénierie de composants pour la réutilisation présentées dans le §1.1.1.12 et plus particulièrement de l'analyse de domaine. Il s'agit également de *définir les moyens pour assurer une gestion efficace des patrons*, notamment en matière d'organisation des patrons afin de faciliter leur réutilisation.

L'ensemble de ces éléments constituent les bases théoriques du cadre méthodologique proposé. La suite de ce manuscrit sera consacrée à la présentation de ce cadre.

Le cadre méthodologique proposé est entièrement supporté par un catalogue ou une collection de patrons. L'essentiel du travail réalisé consiste à définir un catalogue de patrons pour le domaine du SIP. Les chapitres qui suivent illustrent la démarche adoptée pour l'ingénierie des patrons (chapitres III, IV et V), présentent le catalogue de patrons développé (chapitre VI) et finissent par illustrer la démarche d'ingénierie des SIP à base des patrons ainsi définis, sur un cas industriel (chapitre VII).

Chapitre III :

Processus d'Ingénierie de Patrons pour les SIP

1. Introduction

Comme nous l'avons noté dans le chapitre précédent (chapitre II), l'adoption d'une approche d'ingénierie de SIP basée sur la réutilisation de composants nécessite d'abord de produire les composants à réutiliser et ensuite de réutiliser ces composants d'une façon systématique. L'introduction de composants réutilisables dans le processus d'ingénierie des SIP fait donc émerger deux nouveaux processus complémentaires (cf. Figure 3.1) :

- Un processus *pour la réutilisation* est dédié à *l'ingénierie des composants.* Il consiste à identifier, spécifier et organiser les composants à utiliser durant la spécification des SIP.
- Un processus *par la réutilisation* est dédié à *l'ingénierie des SIP*. Partant des spécifications utilisateurs, ce processus facilite la recherche, la sélection, l'adaptation et l'intégration de composants pour spécifier et implanter le SIP.

Dans le cas d'une approche à base de patrons, le processus *pour réutilisation* revient à identifier d'abord les problèmes récurrents lors de l'ingénierie des SIP et ensuite à spécifier différents patrons proposant des solutions aux problèmes identifiés.
Le second processus, celui *par réutilisation* revient à offrir des moyens pour identifier les problèmes à résoudre, sélectionner les patrons qui résolvent ces problèmes et enfin adapter les solutions proposées au contexte dans lequel se trouve le concepteur.

Nous soulignons que ces processus sont récurrents; les modèles obtenus dans les SIP, en résultat du processus "par réutilisation", alimentent le processus "pour réutilisation" afin de tenir compte régulièrement de nouveaux problèmes ou de nouvelles solutions à des problèmes existants.

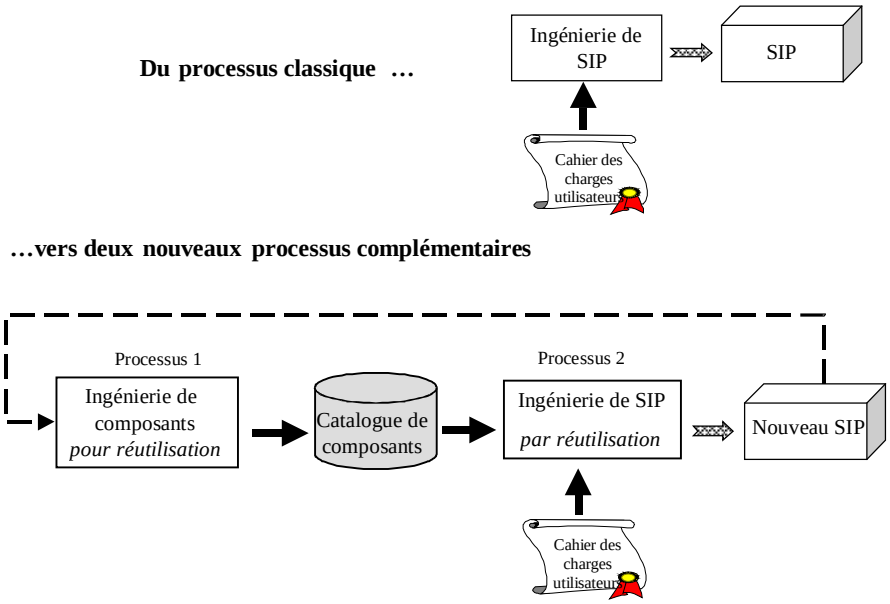


Figure 3.1 - Processus *pour* et *par* réutilisation

L'objectif de ce chapitre est de proposer une démarche pour l'ingénierie des patrons (processus *pour réutilisation*). L'ingénierie de composants réutilisables reste un domaine de

recherche récent, tout comme le domaine d'ingénierie à base de patrons. Ce qui justifie, lors de l'étude présentée au chapitre II, l'absence d'approches d'ingénierie de composants dédiées aux patrons et plus généralement aux composants orientés problèmes. D'où l'originalité de la démarche proposée dans le présent chapitre, qui se veut spécifique aux patrons.

Toutefois et avant de présenter la démarche proposée pour l'ingénierie de patrons, nous souhaitons préciser la nature des patrons dédiés à l'ingénierie des SIP. C'est l'objectif de la première partie de ce chapitre (§2) de présenter ces patrons. La deuxième partie du chapitre développe la démarche d'ingénierie de patrons proposée (§3).

2. Les patrons pour l'ingénierie des SIP

Le cadre méthodologique que nous proposons dans cette thèse repose sur un ensemble cohérent de modèles de différents niveaux d'abstraction pouvant être élaborés par réutilisation de patrons. Chaque niveau proposé doit permettre la résolution d'un problème (de nature conceptuel, organisationnel, technique, etc.) récurrent lors du développement des SIP. Comme pour la conception de systèmes d'information de gestion, la conception de SIP peut être appréhendée selon au moins deux grands niveaux de préoccupation, d'une part un niveau organisationnel conduisant à spécifier le Système d'Information Organisationnel (SIO) et auquel seront associés des patrons dit "métiers" et d'autre part un niveau technique (informatique) conduisant à spécifier la partie de ce SIO qui donnera lieu à informatisation, c'est le Système d'Information Informatisé (SII) et qui conduira à spécifier et réutiliser des patrons dit "logiciels".

- Au niveau organisationnel, les **patrons métiers** sont importants durant les étapes d'analyse des besoins et de spécification fonctionnelle. Ils offrent des solutions à des problèmes du domaine d'application et doivent prendre en compte un ensemble de besoins informationnels associés au produit et aux divers processus agissant sur celui-ci. Dans la définition du niveau organisationnel, deux formes de modélisation sont essentielles : la modélisation des produits (industriels) et la modélisation des processus SIP (processus de définition de produit, processus de modification de produit, etc.). Par rapport à la classification de patrons présentée au chapitre II (§1.1.1.9), les patrons métiers correspondent essentiellement à des patrons d'analyse, voire de conception.
- Au niveau technique, les **patrons logiciels** sont importants durant les phases d'implantation. Leur définition est fortement liée aux SGDT qui sont à la base du développement des SIP. La généricité d'une modélisation à partir de patrons logiciels provient de son indépendance vis à vis d'un SGDT. Cette modélisation est l'expression d'une solution technique qui prend en compte deux problèmes essentiels, celui de l'implantation des modèles de produits et de processus et celui de la communication du SIP avec d'autres systèmes. Concernant l'implantation des modèles de produit et de processus, les SGDT utilisent des modèles de bases de données (relationnelles ou objets) et des modèles de « workflows ». Dans les deux cas, les modèles proposés dans les outils sont largement utilisés et peuvent donc être intégrés dans le cadre méthodologique

proposé. Par rapport à la classification de patrons présentée au chapitre II (§1.1.1.9), les patrons logiciels correspondent à des patrons d'implantation.

Durant cette thèse, nous nous sommes focalisés sur les phases amont de l'ingénierie des SIP (analyse et conception). Le catalogue de patrons développé est dédié aux *patrons métiers*.

3. Démarche d'ingénierie de patrons

L'ingénierie de patrons peut être abordée de la même manière que l'ingénierie de composants en général. Deux aspects sont considérées dans l'ingénierie de composants. D'abord l'identification et ensuite la conception de composants.
- L'identification de composants a pour objectif d'identifier des éléments susceptibles d'être réutilisables et qui soient les plus indépendants possible les uns des autres.
- La conception de composants a pour objectif de spécifier et d'organiser des composants.

Un patron fournit des solutions à des problèmes types, en visant des objectifs de réutilisabilité et de flexibilité. Les deux étapes d'ingénierie de composants introduites ci-dessus se déclinent comme suit pour les patrons :
- L'identification des patrons consiste à identifier les différents problèmes qui sont souvent rencontrés lors de l'ingénierie des SIP. Il s'agit donc d'identifier des problèmes susceptibles d'être récurrents et qui soient les plus indépendants possible les uns des autres, auxquels sont associés différents patrons.
- La conception de patrons consiste à proposer des solutions aux divers problèmes identifiés et de formaliser l'ensemble des problèmes/solutions sous forme de patrons.

La première étape d'identification de patrons constitue désormais un problème de recherche essentiel. Dans ce qui suit, un accent particulier est mis sur ce point. Pour mettre en œuvre cette phase d'identification, il fallait d'abord *identifier les sources* de connaissances susceptibles de contenir des éléments (donc des problèmes) réutilisables et ensuite *identifier ces éléments* (ces problèmes).
Pour identifier les sources de connaissances, nous adoptons une approche d'analyse de domaine[76] classique qui étudie et analyse des systèmes SIP existants (donc des produits d'ingénierie ; des solutions) afin de produire un référentiel du domaine exprimant les ressemblances et les différences entre les différents systèmes existants (en terme d'objets et d'opérations qui caractérisent ces systèmes). Le résultat de cette analyse est un référentiel qui présente les solutions du domaine étudié (les modèles de SIP, auxquels nous devrons aboutir par application des patrons). Il s'agit donc ici d'une *démarche ascendante,* où un domaine est

[76] Dans le chapitre II, nous avons dégagé deux tendances pour l'ingénierie de composants : l'approche de rétro-ingénierie et l'approche d'analyse de domaine. Nous préférons l'approche d'analyse de domaine car, reposant sur des techniques qualitatives (basées sur le recensement et la classification des connaissances d'un domaine), elle nous semble plus adaptée à l'identification des besoins informationnels en SIP, de nature souvent informelle. L'approche de rétro-ingénierie utilise en effet des techniques souvent quantitatives et est plutôt adaptée à des composants de type logiciel et d'une façon générale à des composants dont la description est formelle et quantifiable.

constitué par une famille de systèmes similaires et le modèle de domaine associé (référentiel) est orienté solutions.
Ce référentiel constitue par ailleurs les sources de connaissances susceptibles de contenir des problèmes récurrents (d'ingénierie de SIP). C'est à partir de ce référentiel que les divers problèmes seront identifiés et donc les patrons associés. L'idée consiste à décomposer le modèle de domaine obtenu de manière à ce qu'il correspond à divers bouts de modèles (ou plus généralement divers bouts de spécifications) réutilisables, résolus par des patrons. Le référentiel obtenu à l'issue de l'analyse de domaine est donc analysé autrement. Il est alors perçu comme une classe de problèmes pour lesquels différentes solutions sont envisageables. Il s'agit ici donc d'une *démarche descendante* (orientée problèmes) où le domaine est considéré comme une classe de problèmes pour lesquels plusieurs solutions différentes peuvent exister. L'accent est mis sur la modélisation du domaine sous forme de problèmes, auxquels sont associées des solutions de modélisation.

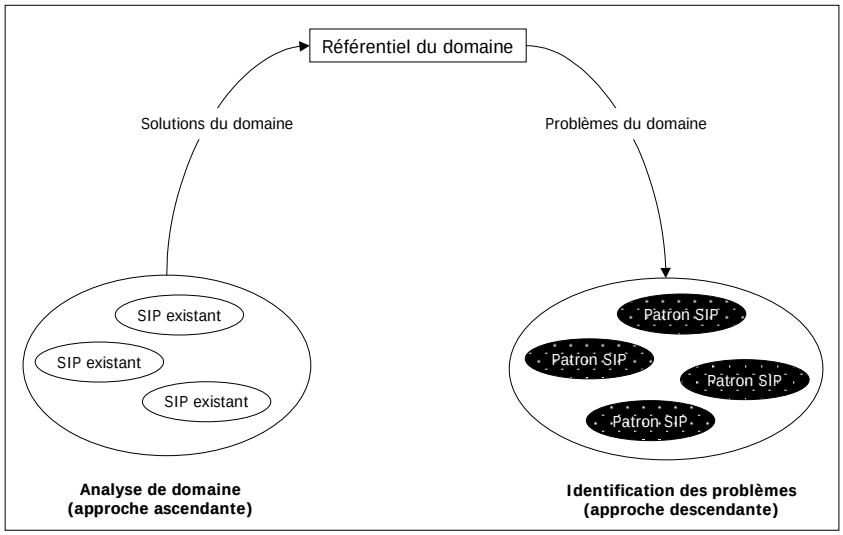


Figure 3.2 - Démarche d'identification des problèmes

Il ressort donc de l'analyse précédente que l'ingénierie de patrons revient d'abord à :
- une *analyse de domaine* pour l'identification des sources de connaissances susceptibles de contenir des problèmes réutilisables et la spécification d'un référentiel de domaine,
- une *identification de problèmes* réutilisables à partir du référentiel de domaine (et donc de patrons)

Ensuite, une fois les patrons identifiés (en terme de problèmes qu'ils soulèvent), il s'agit de les spécifier (concevoir) c'est-à-dire de *proposer une solution au problème* soulevé par chaque patron ainsi identifié.

La Figure 3.3 illustre les principales étapes du processus d'ingénierie de patrons que nous proposons.

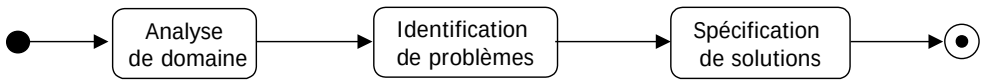


Figure 3.3 - Processus d'ingénierie de patrons

La suite de cette deuxième partie du chapitre est structurée en fonction de ces trois étapes. La Section 3.1 présente donc la démarche d'analyse de domaine permettant d'élaborer un référentiel du domaine. La Section 3.2 définit la démarche retenue pour identifier différents problèmes (et donc des patrons) réutilisables à partir de l'analyse de domaine et la Section 3.3 décrit la manière pour spécifier des patrons associés aux problèmes identifiés (des solutions).

3.1. Analyse de domaine

L'analyse de domaine constitue la première étape dans l'ingénierie de domaine (cf. chapitre II). Elle est définie dans [Hess, 90] comme l'activité d'identification et de représentation d'informations pertinentes pour un domaine donné (une classe de systèmes similaires partageant les mêmes fonctionnalités). Elle fut utilisée depuis plus d'une vingtaine d'années dans des contextes variées : pour appréhender et comprendre un domaine, pour expliquer le fonctionnement d'un domaine et plus récemment pour produire des composants réutilisables.
L'analyse de domaine a pour but d'acquérir les connaissances sur un champ d'applications (les SIP en l'occurrence) et de les structurer dans un modèle de domaine suffisamment générique pour constituer un cadre de référence pour les développements de tous les systèmes d'un domaine donné. Ce cadre de référence exprime en terme de solutions les besoins souvent pris en compte dans les différents systèmes existants.
Ainsi le référentiel obtenu constitue une source de connaissances susceptibles de contenir des besoins (des problèmes) réutilisables. Les solutions exprimées dans ce référentiel ne sont autres que les modèles du SIP devant être élaborés par réutilisation de patrons.
Un patron s'adresse à un problème rencontré dans une phase donnée de l'ingénierie de SIP. Selon la phase concernée (analyse, conception), la nature du problème traité varie. Les patrons d'analyse s'adressent à des problèmes qui apparaissent lors de l'analyse des besoins du SIP. Il s'agit donc de besoins informationnels que les utilisateurs expriment, c'est-à-dire des informations qu'ils souhaitent gérer dans le SIP. Les patrons de conception s'adressent à des problèmes qui apparaissent lors de la conception du SIP. Il s'agit donc de problèmes du domaine de la conception (orientée objet dans notre cas) que les concepteurs du SIP expriment, c'est à dire des problèmes de modélisation (ou d'abstraction) des besoins informationnels selon des thèmes communs du domaine de la conception orientée objet.
Ainsi, l'analyse de domaine revient :

- dans un premier temps à l'exploration des besoins informationnels fréquemment exprimés par les experts et les utilisateurs du SIP durant la phase d'analyse. Il s'agit de connaissances relatives aux différents concepts manipulés dans les SIP. En d'autres termes, des connaissances de domaine.
- dans un second temps à l'exploration des besoins de modélisation souvent exprimés par les concepteurs SIP pour spécifier les connaissances de domaine identifiés auparavant. Il s'agit donc d'explorer des connaissances relatifs à des techniques de spécification en ingénierie des systèmes d'information. En d'autres termes, des connaissances de développement.

L'objectif est donc d'élaborer un référentiel du domaine des SIP qui :

- propose un cadre terminologique et sémantique des connaissances de domaine. Ce cadre doit permettre d'identifier et de classifier l'ensemble des objets ou concepts gérés dans le SIP,

- identifie les problèmes de modélisation associés et,
- propose un modèle de référence qui résout les problèmes de modélisation en structurant les concepts (et notamment les relations entre ces concepts) et en explicitant leur comportement (à l'aide de connaissances de développement).

Le référentiel du domaine ainsi obtenu identifie, structure et lie les besoins informationnels rencontrés dans les projets de spécification de SIP et les problèmes de modélisation associés.

Comme énoncé précédemment, nous adoptons une approche ascendante d'analyse de domaine pour mettre en œuvre les activités d'analyse de domaine[77] et qui consiste à identifier des solutions (des produits d'ingénierie) communes. Cette analyse est menée selon deux aspects :

- un aspect théorique, en se basant sur l'étude de modèles de SIP proposés dans la littérature (approche descendante). Les contributions qui explicitent et modélisent l'ensemble des concepts gérés dans les SIP sont toutefois rares. Nous complétons cette étude par celle de divers travaux et ouvrages qui s'adressent à des parties du domaine des SIP telles que la gestion de configuration (pour la structure produit), la gestion de workflow (pour le processus SIP) ou encore la gestion électronique de documents GED (pour la documentation de produits).
- un aspect expérimental, en se basant sur l'étude des SIP existant. Cette étude est menée à deux niveaux :
  - niveau organisationnel : ceci consiste d'une part au recensement chez le partenaire industriel considéré des connaissances manipulées pour représenter et gérer les produits et d'autre part à l'analyse de la pratique industrielle de divers processus de gestion de données techniques tels que le processus de définition de produit ou le processus de modification de produits industrialisés, à travers l'identification des logiques d'action des acteurs impliqués dans ces processus et des supports d'information utilisés. Cette dernière partie a été menée en collaboration avec les partenaires sociologues du projet POSEIDON, en procédant à des enquêtes sociologiques et en "pistant" quelques instances des processus considérés[78].
  - niveau logiciel : en analysant les modèles de données et les fonctionnalités proposées dans quelques logiciels SGDT du marché ainsi que celles développées dans les applications SIP chez notre partenaire industriel. Nous dégageons l'ensemble des objets et des traitements partagés par l'ensemble des systèmes mais également les différences entre les systèmes.

La Figure 3.4 résume l'ensemble des activités ci-dessus décrites pour mettre en œuvre l'analyse de domaine.

---

[77] Cette mise en œuvre demeure aujourd'hui un axe de recherche, aussi bien en terme de démarche à suivre que de techniques à utiliser. Toutefois et de la même manière qu'en ingénierie de domaine, deux stratégies de mise en œuvre d'analyses de domaine peuvent être distinguées. Une analyse de domaine peut être centrée sur la modélisation des problèmes et des décisions qui conduisent le processus de résolution de ces problèmes, en utilisant des modèles de représentation du domaine (approche descendante). Elle peut être également centrée sur la détermination des ressemblances et des différences entre des systèmes existants d'un domaine, en terme d'objets et d'opérations existants dans ces systèmes, en se basant sur l'observation du champ d'application à travers l'analyse de différents systèmes existants du domaine (approche ascendante).

[78] Il s'agit par exemple de considérer une modification de produit déjà menée dans l'entreprise et interroger sur leur pratique, les acteurs concernés. Une description des études sociologiques menées se trouve dans [Guffond, 98] [Guffond, 99].

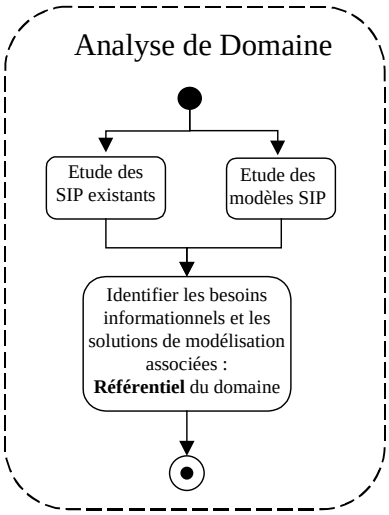


Figure 3.4 - Démarche d'analyse de domaine

Les deux chapitres suivants (chapitres IV et V) illustrent les résultats de l'analyse de domaine ainsi effectuée. Nous nous contentons ici de donner un schéma conceptuel du référentiel produit qui est un des résultats de notre analyse de domaine (cf. Figure 3.5).

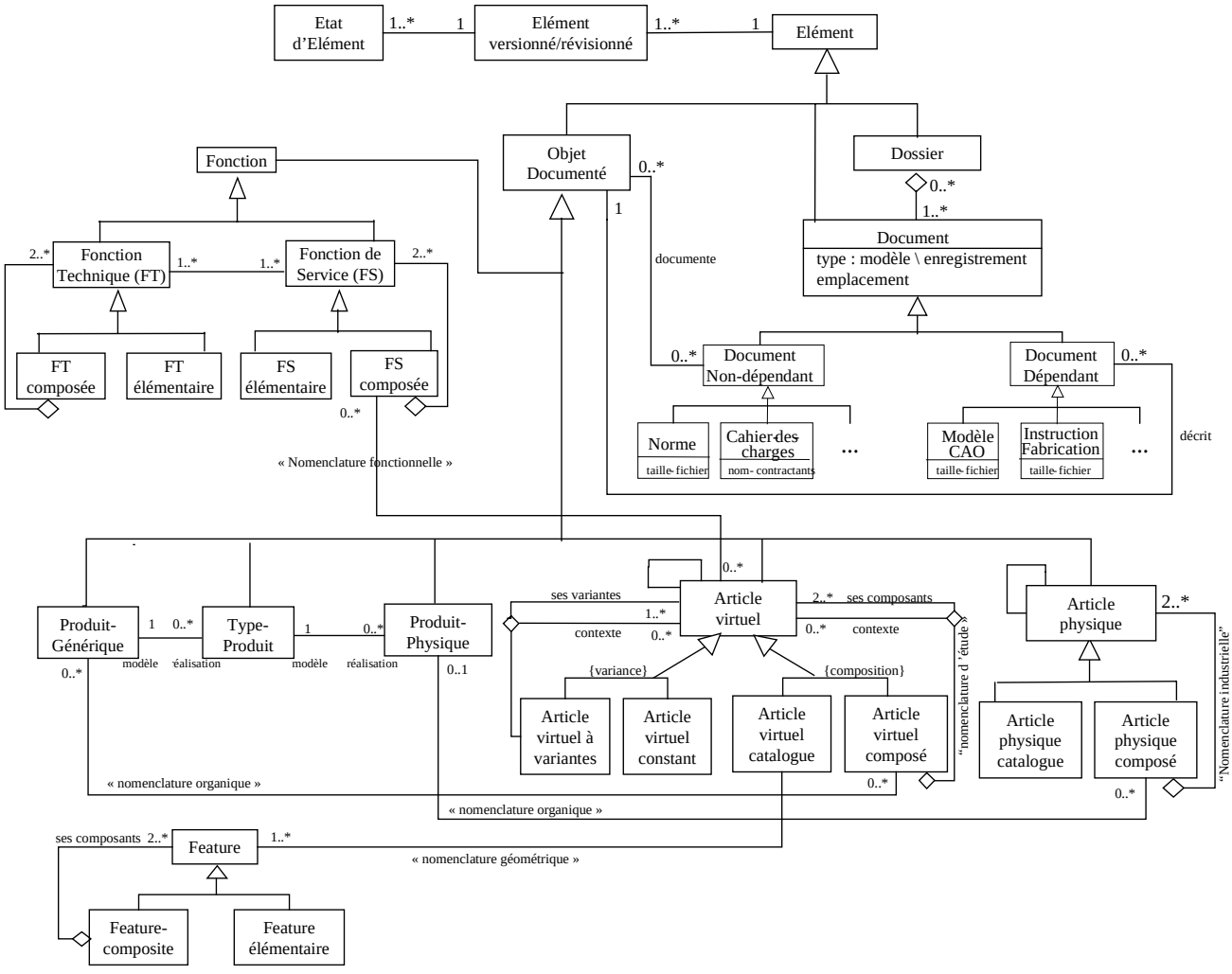


Figure 3.5 - Exemple de résultats de l'analyse de domaine

3.2. Identification de problèmes

L'objectif de l'approche à base de patrons est de mettre en évidence les divers besoins informationnels souvent rencontrés dans les projets de développement de SIP (identifiés dans l'étape précédente) et ensuite de proposer une manière de les modéliser afin d'obtenir à la fin de la phase de spécification un modèle de SIP, spécifique à l'organisation. L'analyse de domaine a produit un référentiel du domaine. Tel qu'il est, il n'a que peu d'intérêt pour un concepteur de SI qui souhaite travailler par réutilisation. Il est nécessaire d'organiser la connaissance de domaine identifiée dans le référentiel, sous forme de patrons.
L'objectif de chaque patron est de fournir un fragment du modèle SIP attendu, correspondant à un problème particulier à résoudre, selon la spécificité de l'organisation. Chaque patron doit par ailleurs constituer une unité de raisonnement modulaire : il est le plus indépendant possible des autres patrons, résout un problème standard et peut être réutilisé plusieurs fois.
Ainsi la deuxième étape de l'identification de patrons, c'est-à-dire l'identification de problèmes revient à mettre en évidence les différents fragments du modèle SIP à construire par réutilisation de façon à associer un patron à chacun de ces fragments. Le modèle SIP est alors fragmenté de tel sorte que chaque fragment résolve un problème récurrent dans l'ingénierie de SIP.

A l'issue de l'analyse de domaine, dont les résultats seront développés dans les deux chapitres suivants (chapitres IV et V), le référentiel de domaine élaboré met en évidence l'existence de besoins informationnels "permanents" qui sont souvent gérés dans les SIP et de besoins informationnels "variants" qui varient d'un SIP à un autre, selon la spécificité de l'entreprise.

Explicitement, nous distinguons dans le référentiel divers "blocs" de besoins informationnels. Chaque bloc est une constante du modèle SIP, c'est-à-dire qu'il existe dans tout modèle SIP, quelque soit la spécificité de l'entreprise. Le bloc correspond en fait à une information qui doit être gérée dans l'entreprise. Ainsi, si l'on considère le même exemple précédemment présenté pour illustrer les résultats d'analyse de domaine (Figure 3.5), les divers blocs de ce modèle de domaine sont relatifs aux niveaux de produit, aux nomenclatures de produit et aux documents techniques associés aux produit (cf. fragments encerclés dans la Figure 3.6).

Toutefois, la constitution d'un bloc est variable. Les concepts constituant un bloc ainsi que les liens entre ces concepts peuvent varier d'un SIP à l'autre, selon la spécificité de l'entreprise.
Soulignons ici que par variance, il ne s'agit pas d'une variance au niveau de la modélisation d'une même connaissance mais d'une variance au niveau de la connaissance elle même. En effet, il peut y avoir justement une variance au niveau de la modélisation, en modélisant différemment une même connaissance pour des raisons liés par exemple à la technologie informatique cible ou pour exprimer des vues différentes de la même connaissance. Ici, il s'agit plutôt d'une variance de la connaissance entre les entreprises. Bien évidemment, cette variance "métier" engendre différentes modélisations, correspondant chacune à une variété de la connaissance considérée.
Ainsi, et par rapport au même exemple considéré, le bloc "Niveaux de produit" peut être construit avec une, deux ou trois classes (voire plus) selon que l'entreprise gère un, deux ou trois niveaux de produits dans l'entreprise. Nous distinguons en effet dans le référentiel trois niveaux de produit: le produit générique, le type-produit et le produit physique (voir chapitre IV).

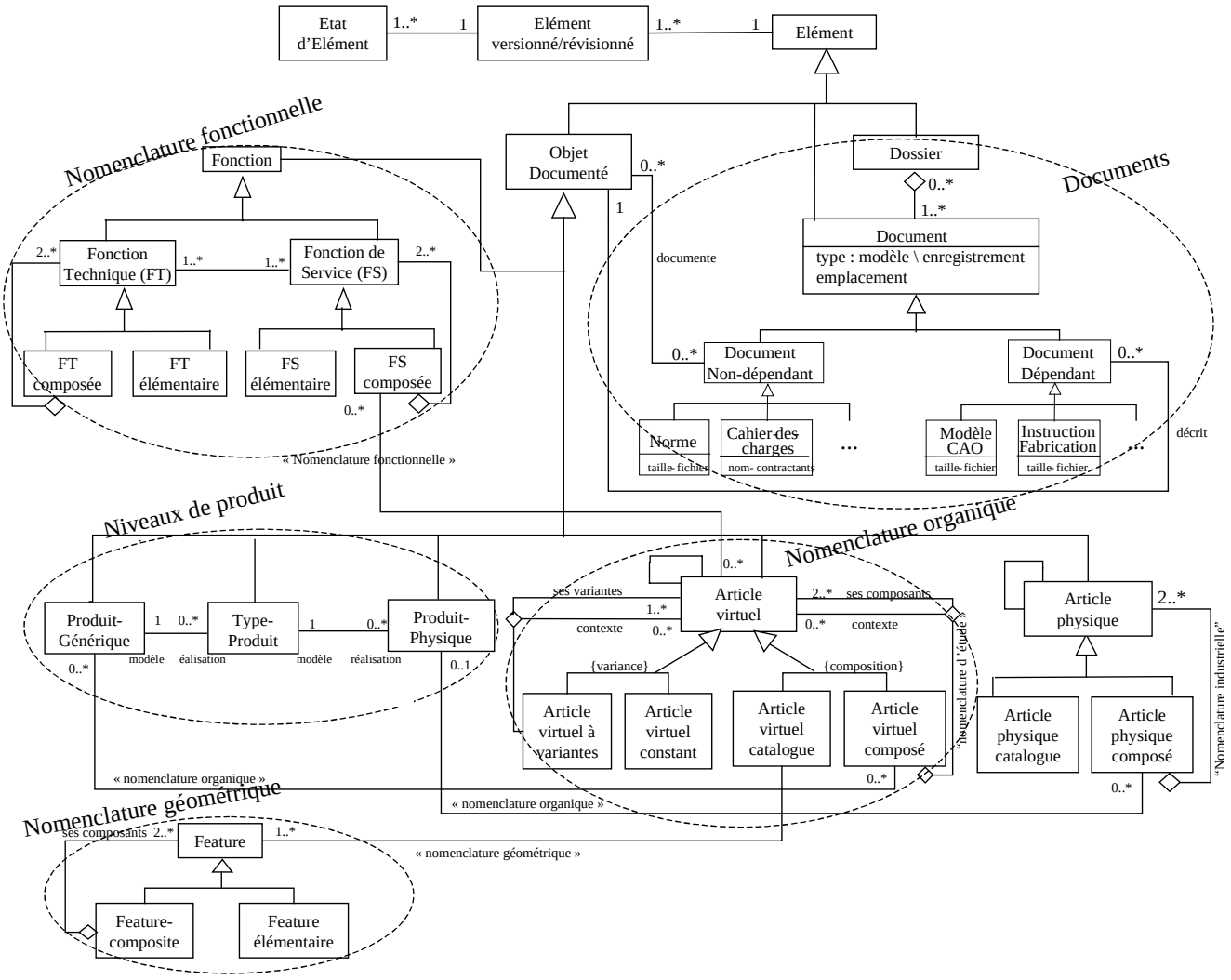


Figure 3.6 - Exemple de blocs du modèle de domaine

Le référentiel élaboré peut être vu donc comme un cadre générique, exprimant d'un côté les besoins informationnels communs à toutes les organisations et de l'autre côté la variabilité de ces besoins, selon les organisations.
La spécification des SIP par réutilisation de patrons doit tenir compte de cette variabilité et donc permettre d'élaborer le modèle d'un SIP spécifique, selon les préférences ou caractéristiques de l'organisation.

La première phase d'identification des problèmes (et donc des patrons) revient ainsi à distinguer les divers blocs dans le référentiel obtenu et ensuite à identifier, pour chaque bloc, les divers points de variabilité propres à ce bloc et qui impliquent différentes constructions du fragment de modèle associé au bloc. Une fois les blocs et les points de variabilité identifiés, nous associons à chaque bloc un patron dont l'objectif est de construire le fragment de modèle qui représente le bloc considéré. Ce patron fixe les différents points de variabilité propres au bloc considéré (selon la spécificité de l'entreprise) et dirige ensuite vers divers autres patrons permettant chacun de construire le fragment de modèle représentant le bloc considéré selon un point de variabilité fixé. La Figure 3.7 résume cette démarche.

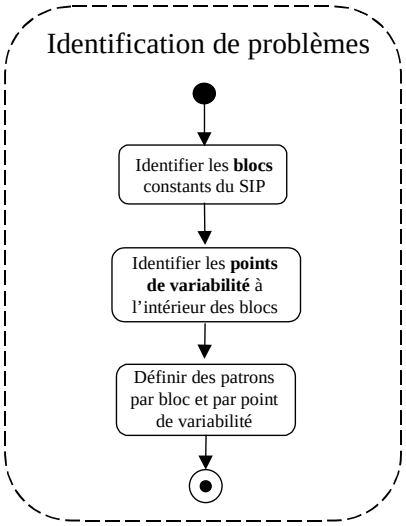


Figure 3.7 - Démarche d'identification de problèmes

Pour illustrer cette démarche, nous considérons le bloc "Niveaux de produit" identifié dans la Figure 3.6. Ce bloc correspond à une information obligatoirement représentée dans tout modèle SIP ; celle relative au produit (avec ses différents niveaux). Nous définissons ainsi un premier patron associé à ce bloc qui permet de construire le fragment de modèle représentant les niveaux de produits gérés dans le SIP. Ce patron fixe pour une entreprise donnée le nombre et le type de niveaux de produit qu'elle gère (produit générique et/ou type produit et/ou produit exemplaire) et dirige vers trois autres patrons permettant, selon le choix fait dans le patron principal, de construire un fragment de modèle représentant un, deux ou trois niveaux de produit (cf. Figure 3.8).

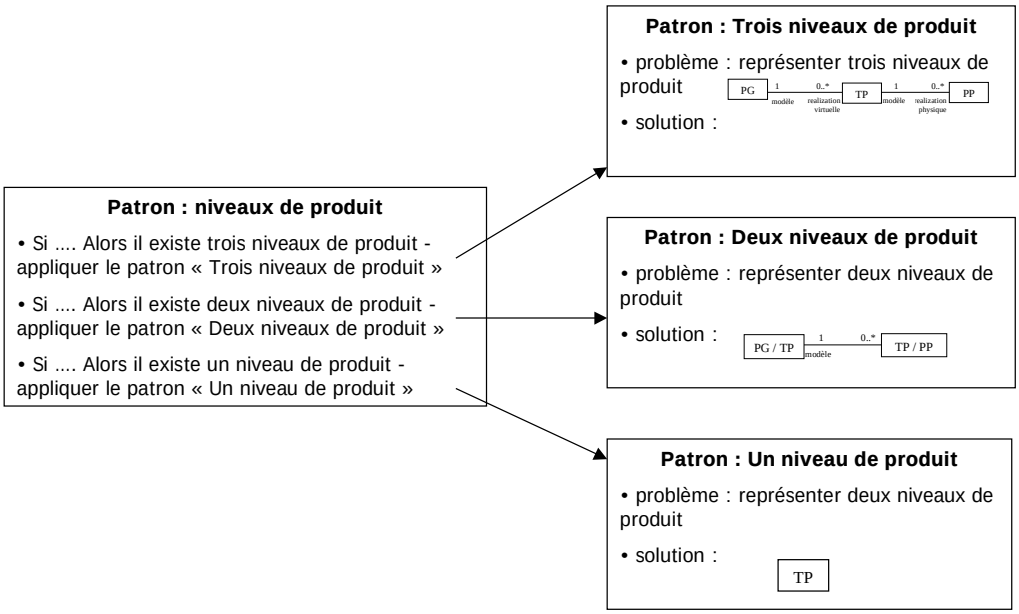


Figure 3.8 - Patrons du bloc "Niveaux de produit"

3.3. Spécification de solutions

Une fois les patrons identifiés (en terme de problèmes à résoudre), la troisième phase du processus d'ingénierie de patrons consiste à spécifier les solutions offertes ces patrons. Lorsque le problème considéré est un problème de modélisation, cette phase est basée sur l'exploitation de solutions existantes, disponibles dans les catalogues de patrons de conception proposés dans la littérature (patrons de Gamma, Coad, etc.). Dans ce cas, le problème de modélisation considéré est comparé aux problèmes traités dans les catalogues existants. Plusieurs scénarios peuvent exister lors de la spécification de la solution (cf. Figure 3.9) :

- le problème est déjà traité dans les catalogues de conception existants : la solution correspondante est alors adaptée et intégrée dans le patron considéré.
- le problème est un raffinement (un cas particulier) d'un problème déjà traité dans les catalogues de conception existants : une nouvelle solution est alors définie en se basant sur la solution existante résolvant le problème général.
- le problème est complexe et peut être ramené à des problèmes plus simples traités dans les catalogues de conception existants : le problème est alors divisé en des sous-problèmes traités dans d'autres catalogues. Les solutions proposées sont adaptées et intégrées dans la solution du patron considéré.
- le problème est nouveau : une nouvelle solution est alors proposée.

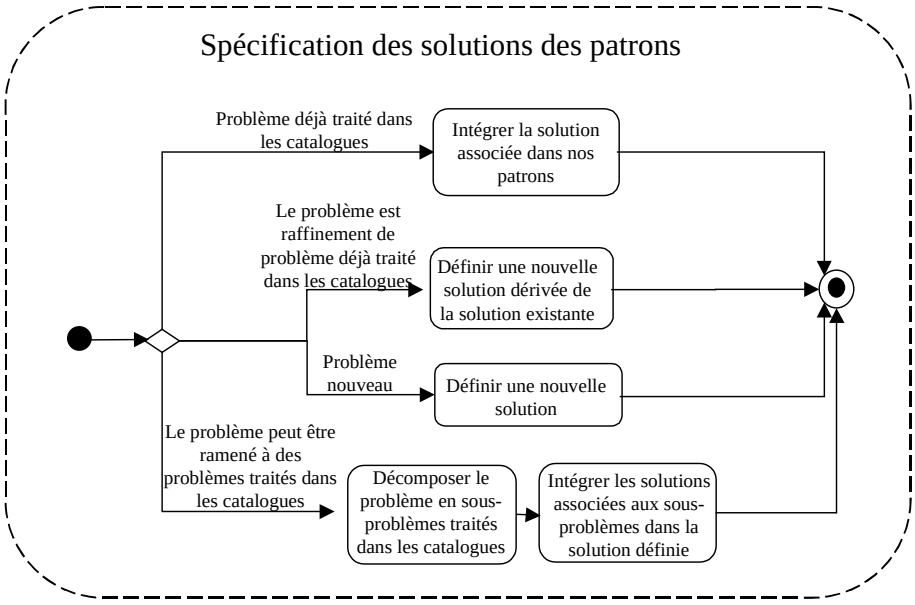


Figure 3.9 - Démarche de spécification des solutions des patrons

Pour illustrer cette démarche, nous considérons, le même exemple relatif aux niveaux de produit. Si l'on considère le patron "Deux niveaux de produit", le problème de ce patron est de représenter le concept de produit en deux niveaux d'abstraction. Ce problème est résolu par des patrons de conception proposés dans la littérature tels que le patron "Item-description"[79] de P. Coad [Coad, 92] ou encore le patron "Materialization" [Dahchour 98]. Ces patrons de

[79] Ce patron est présenté dans le chapitre II.

conception permettent de relier une classe de catégories abstraites A (modèles de voitures par exemple) à une classe d'objets plus concrets C (des voitures individuelles).

4. Conclusion

L'ingénierie de composants constitue un champ de recherche récent. Des questions essentielles restent posées quant à la manière de mettre en œuvre l'ingénierie de composants ou encore à la spécificité de cette forme d'ingénierie vis à vis de l'ingénierie traditionnelle des systèmes d'information. L'ingénierie des composants doit être basée sur des techniques favorisant leur réutilisabilité, tout en respectant les principes essentiels d'abstraction et de variabilité.

Selon la nature du composant, diverses méthodes ont été proposées. Dans un contexte de réutilisation de patrons, nous avons défini une démarche pour l'ingénierie de ces composants. Cette démarche est constituée de trois étapes (cf. Figure 3.10) :

- L'analyse de domaine : elle consiste à produire un référentiel du domaine recensant d'une part les besoins informationnels souvent rencontrés dans les projets de développement de SIP (qualifiés de "connaissances du domaine") et d'autre part les problèmes de modélisation associés à ces besoins (qualifiés de "connaissances de développement"). Cette analyse est basée sur l'étude des modèles de SIP proposés dans la littérature mais également sur l'étude des systèmes SIP mis en place.
- L'identification de problèmes (et de patrons) : le référentiel obtenu lors de l'analyse de domaine sert de base pour l'identification de divers fragments du modèle de SIP (divers problèmes) à construire par réutilisation, de façon à associer un patron à chacun de ces fragments. A l'issue de cette phase, on identifie un ensemble de patrons organisant les problèmes (besoins informationnels) souvent rencontrés dans le domaine d'ingénierie de SIP selon des besoins informationnels permanents et des besoins variants dans les SIP.
- La spécification des solutions : une fois les patrons identifiés, cette phase consiste à spécifier les solutions offertes par ces patrons aux problèmes identifiés. Elle est partiellement basée sur l'exploitation des catalogues de conception proposés dans la littérature.

Ce chapitre clôt ainsi la première partie du manuscrit, consacrée au cadre de réflexion du travail de thèse. Les deux autres parties de ce manuscrit ont pour objectif d'illustrer les étapes de la démarche d'ingénierie de patrons, définie dans le présent chapitre. Ainsi :

- la 2ème partie du manuscrit présente l'analyse de domaine. Nous décrivons le référentiel du domaine obtenu à l'issue de cette analyse. Ce référentiel identifie l'ensemble des sources de connaissances susceptibles de contenir des éléments réutilisables. Il illustre ainsi la première étape de la démarche d'ingénierie de patrons. Il initie également la deuxième étape de cette démarche en mettant en évidence d'une part les besoins informationnels "permanents" dans le SIP (les divers blocs de concepts) et d'autre part les besoins informationnels "variants" dans le SIP (les divers points de variabilité à l'intérieur de chaque bloc concept).

- La 3ème partie du manuscrit développe les patrons proposés pour résoudre les problèmes récurrents identifiés dans le référentiel. Elle illustre ainsi la suite de la deuxième étape d'ingénierie de patrons, ainsi que la troisième étape de cette démarche.

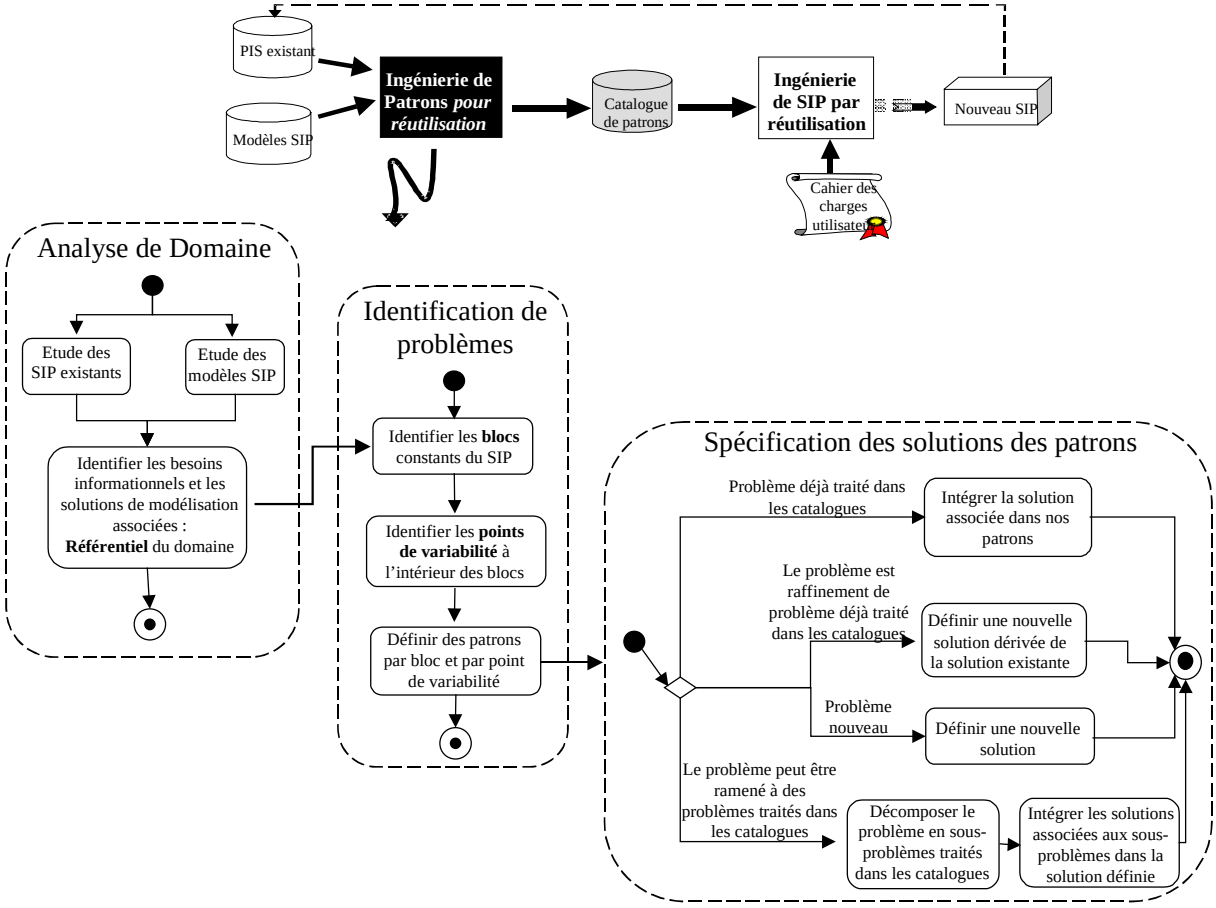


Figure 3.10 - Démarche d'ingénierie de patrons

Deuxième Partie :

Le Référentiel du Domaine

Le référentiel du domaine SIP a pour objectif d'explorer les besoins informationnels fréquemment exprimés par les utilisateurs des SIP durant la phase d'analyse des besoins ainsi que les problèmes de modélisation souvent rencontrés par les concepteurs SIP pour spécifier les besoins informationnels identifiés. Il propose ainsi de :

- délimiter et classifier l'ensemble des objets ou concepts gérés dans le SIP (les connaissances du domaine),
- revoir la terminologie utilisée pour chacun des concepts identifiés et proposer un cadre terminologique et une sémantique pour l'ensemble des concepts,
- identifier les problèmes et les techniques de modélisation associés aux différents concepts (connaissances de développement),
- proposer un modèle de référence du domaine qui résout les problèmes de modélisation en structurant les concepts (explicitant les relations entre ces concepts) et décrivant leur comportement.

En terme de connaissances de domaine, les SIP sont centrés sur les produits industriels, ils gèrent l'ensemble des informations techniques décrivant le produit à un moment donné ainsi que les divers processus agissant sur le produit et faisant évoluer celui-ci. Les concepts gérés dans un SIP s'articulent alors autour de deux axes :

- l'axe **produit** : raison d'être du SIP et relatif à la description des produits,
- l'axe **processus** : environnement du produit et comprenant les processus agissant sur le celui-ci durant son cycle de vie. Il est relatif à la description des processus du SIP et de la structure organisationnelle mise en place pour assurer ces processus.

Nous décrivons ainsi le référentiel du domaine SIP en deux chapitres, selon ces deux axes. Le chapitre IV présente les connaissances produit et le chapitre V est consacré à la description des connaissances processus.

Dans chacun de ces deux chapitres et pour chaque concept étudié, nous présentons d'abord une description globale du concept et nous proposons ensuite une structuration de ce concept. A cet effet, nous soulignons d'abord les problèmes de modélisation éventuels et présentons ensuite une modélisation adéquate pour répondre à ces problèmes. Dans un premier temps, les concepts sont modélisés indépendamment les uns des autres. Ensuite, nous proposons à la fin de chaque chapitre un modèle global explicitant en particulier les liens entre les divers concepts structurés.

Chapitre IV :

Référentiel Produit

1. Introduction

Dans cet chapitre, nous passons en revue l'ensemble des *besoins informationnels* souvent exprimés autour du produit. Nous soulignons les éventuels problèmes de modélisation associés à chacun des concepts étudiés et nous proposons une modélisation adéquate pour répondre à ces problèmes. Toutefois, avant de présenter en détail les connaissances associées au produit, nous montrons d'abord dans la Section 2 que le produit existe à différents niveaux d'abstraction, auxquels sont associées des connaissances différentes. Nous passons ensuite en revue l'ensemble des concepts associées au différents niveaux de produit (§3). Certains de ces concepts sont communs à tous les niveaux d'abstraction, d'autres sont spécifiques à certains niveaux. Enfin, nous présentons une structuration d'ensemble des concepts étudiés, en mettant en évidence les liens entre les modèles partiels (§4).

2. Le produit : un concept à différents niveaux d'abstraction

Le terme produit est un concept général dont l'emploi et la signification diffèrent selon le contexte. Ainsi, le terme produit peut faire référence soit à des objets virtuels (le produit objet de l'activité de conception qui figure dans un catalogue) soit à des objets physiques (le produit objet d'une livraison au client). Par ailleurs, nous constatons que le terme 'produit' prend différents aspects selon le métier considéré de l'entreprise. On parle de produit, d'exemplaire de produit, de produit générique, de famille de produit, de produit spécifique, de produit de base, de produit modèle, etc.. Des termes différents pour exprimer parfois la même chose ou un même terme qui n'a pas la même signification et donc utilisé pour désigner des objets différents. Ce constat amène à dire qu'il existe différents niveaux d'abstraction du concept produit, que nous nous proposons de fixer et de définir afin de les organiser entre eux. Cette distinction entre les niveaux d'abstraction du produit n'est pas due simplement à une volonté de lever une ambiguïté terminologique mais surtout à mettre en évidence que les connaissances associées à un produit ne sont finalement pas relatives au même produit mais à des niveaux différents de ce produit. Afin d'organiser les différentes connaissances attachées au produit, il est nécessaire de distinguer ces différents niveaux d'abstraction afin d'y affecter les différentes connaissances appropriées.

2.1. Les différents niveaux

A partir de l'observation des activités des entreprises industrielles, on distingue le produit virtuel et le produit physique.

La fabrication, la distribution (ainsi que l'exploitation-maintenance et la mise au rebut chez le client) concernent le **produit physique**. Il peut s'agir d'un exemplaire individualisé ou d'un lot[80] de produits ayant généralement un numéro de série l'identifiant de façon unique et le différenciant des autres exemplaires ou lots de produits.

Le développement par contre concerne le **produit virtuel**. Il s'agit d'un modèle de produit selon lequel des exemplaires ou des lots (ayant les mêmes paramètres technologiques et la

[80] Un lot est un ensemble d'exemplaires fabriqués exactement dans les mêmes conditions, parmi lesquels la distinction entre exemplaires est impossible.

même définition) vont être fabriqués et qu'on présente généralement dans les catalogues. Ce produit virtuel peut avoir:

- soit une configuration unique c'est-à-dire un modèle où tous les choix de composants sont figés. C'est une configuration réalisable, directement mise en œuvre lors de la fabrication. Les exemplaires constituent donc sa réalisation physique. On parle alors de ***type de produit***.
- soit une configuration générique c'est-à-dire un modèle défini par une ensemble d'options et de variantes[81] et à partir duquel des choix de composants sont à faire pour décliner diverses configurations réalisables. C'est le cas des entreprises qui adoptent la conception routinière ou celles dont l'offre produit est personnalisée selon les besoins des clients. Le développement ne concerne pas un type unique de produit mais un produit plus général qui peut être réutilisé plusieurs fois pour la conception de divers types de produit. Un tel produit est constitué de composants obligatoires et de composants optionnels et pour chacun de ces composants, il y a plusieurs variantes. On parle dans ce cas de ***produit générique***.

En conclusion, nous retenons trois niveaux de produit : le *produit-physique*, le *type-produit* et le *produit-générique*. Le type-produit, ainsi que le produit-générique (quand il existe) constituent des spécifications du produit à fabriquer. Ce sont des produits théoriques ou virtuels, résultats du processus d'ingénierie. Le produit-physique par contre est le résultat du processus de production; c'est un produit matériel qui peut s'écarter peu ou prou du produit théorique (selon les conditions dans lesquelles il a été fabriqué).
La Figure 4.1 illustre nos propos, sur le cas du produit "voiture" :

Produit Générique	**Type Produit**	**Produit Physique**
peugeot 206	peugeot 206-S16	ma peugeot 206-S16
Nom-PG = peugeot 206 Moteur = thermique **ou** à essence **ou** électrique Couleur = vert **ou** rouge **ou** bleu	Nom-TP = peugeot 206-S16 Moteur = à essence Nb culasses = 2 Nb soupapes = 24 Couleur = vert **ou** bleu	Nom-PP = ma peugeot 206-S16 Moteur = à essence Nb culasses = 2 Nb soupapes = 24 Couleur = bleu N°-série = S16.15.120

Figure 4.1 - **Exemple de trois niveaux de produit**

Par rapport à ces définitions, la notion de *famille de produit*[82], souvent citée dans la littérature, paraît donc comme étant l'ensemble des types de produit résultant d'un même produit générique.
Nous notons par ailleurs qu'en partant d'un produit virtuel présentant un ensemble d'options et de variantes (un produit générique), il est possible d'avoir plusieurs niveaux de raffinement dans la définition du produit virtuel avant d'arriver à un produit virtuel réalisable. Pour relever cette ambiguïté dans la définition des niveaux de produit, nous considérons que le type produit doit être un produit virtuel *réalisable* c'est-à-dire composé d'un ensemble de composants réalisables et compatibles entre eux. Pour cela, il ne doit pas y avoir dans la

[81] Nous revenons sur ce concept dans le §3.1.5.

[82] Définie dans la norme NF L00 007 B [BNAE, 92] comme étant *un ensemble de produits **résultant** d'une conception de base commune, mais comportant des **types** différents à l'intérieur desquels existent des versions et des variantes.*

composition du type-produit deux ou plusieurs variantes non interchangeables[83] d'un même composant (par exemple moteur diesel et moteur électrique) ou encore deux composants incompatibles (par exemple un moteur électrique et un réservoir de carburant). Ainsi :

- Le type-produit est un sous-ensemble de produit générique décliné de celui-ci par la mise en œuvre d'un certain choix d'options et de variantes réalisables et compatibles entre elles. Le choix entre options et variantes est contraint : le choix de certaines options ou variantes fixe souvent certaines autres options ou variantes à choisir. Pour illustrer cela, considérons le cas du produit voiture où au niveau générique un véhicule est composé de moteur dont les variantes sont : *moteur 2 cylindres / moteur 3 cylindres / moteur 4 cylindres*. Au passage à un type-produit particulier, le choix de la variante *moteur 4 cylindres* pour le composant moteur implique le choix de la variante *16 soupapes* pour le composant soupape.
- Le produit-physique est obtenu par la réalisation physique d'un type-produit avec un choix particulier entre variantes. Nous supposons que le type-produit peut présenter un ensemble de variantes interchangeables parmi lesquels un choix est à faire lors de la fabrication. A titre d'exemple, la forme du rétroviseur peut varier d'un exemplaire de voiture à l'autre (rétroviseur rectangulaire, à coins arrondis, etc.) ou encore les coloris possibles des exemplaires d'un type de voiture particulier (carrosserie bleue, carrosserie jaune, etc.). Ainsi, au passage d'un type de voiture à un exemplaire donné, il est possible de fixer l'une ou l'autre des formes de rétroviseur, sans que ce choix influe le reste de la composition de l'exemplaire (que le rétroviseur soit rectangulaire ou à coins arrondis, la liaison au toit de la voiture ne change pas par exemple).

Nous déduisons alors que :

- un produit virtuel est générique quand il contient dans sa composition au moins deux variantes non interchangeables d'un même composant ou un composant optionnel incompatible avec un ou plusieurs composants obligatoires,
- un produit virtuel est un type de produit lorsqu'il ne présente aucun choix de variantes ou contient des variantes interchangeables ou encore contient des composants optionnels compatibles avec les composants obligatoires.

2.2. Les connaissances sur les niveaux

En terme de connaissances associées aux niveaux de produit, il ressort de l'exemple présentée dans la Figure 4.1 que les connaissances associées aux trois entités Produit Générique (PG), Type Produit (TP) et Produit Physique PP diffèrent selon le niveau d'abstraction. L'entité Produit-Générique a un niveau d'abstraction supérieur à l'entité Type-Produit lui même supérieur à Produit physique. Ces connaissances sont de deux types :

- les **propriétés** des entités : une entité prend une valeur pour chacune de ses propriétés. A titre d'exemple , *"Nom-PG = peugeot 206"* est une propriété de *peugeot 206*, "*moteur = à essence"* est une propriété de *peugeot 206-S16,* de même que *"n° de série=S16.98.100"* ou *"couleur = bleu"* sont des propriétés de *ma peugeot 206-S16.* La valeur d'une propriété à un certain niveau d'abstraction peut rester visible au niveau inférieur. A titre d'exemple, la

[83] Deux objets A et A' sont interchangeables si A' peut être substitué à A sans que cela n'entraîne l'évolution d'un quelconque composé de A [Maurino, 93].

propriété *"moteur = à essence"* du type de produit *peugeot 206-S16* reste visible dans le produit physique *ma peugeot 206-S16.*

- les **contraintes** : exprimées à un niveau d'abstraction et portant sur des valeurs de propriétés de niveau inférieur. A ce titre, *"moteur = thermique ou à essence ou électrique"* exprimée pour le produit générique *peugeot 206*, est une contrainte sur la valeur de la propriété *"moteur"* du type de produit *peugeot 206-S16* (moteur = 4 cylindres). Une contrainte portant sur une propriété du niveau produit physique (par exemple la propriété *"couleur = bleu"* pour ma peugeot 206-S16) peut être exprimée au niveau du produit générique (*"couleur = bleu ou rouge ou vert"*) et restreinte au niveau du type produit (*"couleur = vert ou bleu "*).

La Figure 4.2 distingue les propriétés et les contraintes des trois niveaux de produit introduits dans l'exemple de la Figure 4.1.

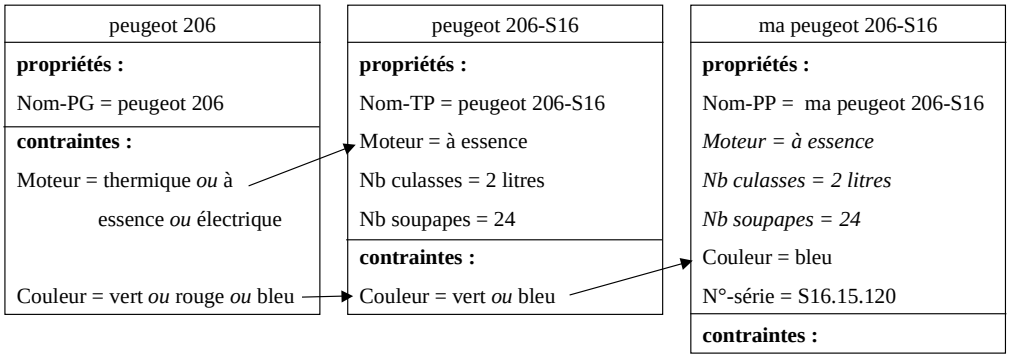


Figure 4.2 - propriétés et contraintes

En terme de **structuration**, le modèle proposé représente le concept de produit en trois niveaux d'abstraction permettant d'une part de partitionner les connaissances relatives aux différents niveaux et d'autre part de définir des relations entre les niveaux afin de permettre la propagation de propriétés entre les niveaux. Ce modèle doit pouvoir exprimer les propriétés des entités mais aussi des contraintes sur des propriétés. Ces contraintes sont exprimées à un niveau d'abstraction et portent sur des valeurs de propriétés de niveau inférieur. Deux caractéristiques fondamentales de ces propriétés et contraintes sont à prendre en compte :

– les propriétés d'un niveau doivent rester visibles au niveau inférieur,
– les contraintes sur une propriété d'un niveau inférieur peuvent être successivement raffinées à travers les niveaux supérieurs. Telle est le cas de la propriété "couleur" de *ma peugeot206-S16*, contrainte d'abord au niveau générique *peugeot 206* (vert / rouge / bleu) et raffinée ensuite au niveau type *peugeot 206-S16* (vert / bleu).

Chaque niveau d'abstraction du produit est représenté donc par une classe (cf. Figure 4.3). Le Type-Produit (resp. Produit-Physique) entretient une association avec le Produit-Générique dont il est issu exprimant que le Produit-Générique *a pour réalisation* virtuelle le Type-Produit. De même, le Produit-Physique entretient une association avec le Type-Produit dont il est issu exprimant que Produit-Physique *a pour modèle* Type-Produit.

La modélisation de niveaux d'abstraction revient donc à relier une classe de catégories abstraites A (modèles de voitures) à une classe d'objets plus concrets C (voitures individuelles). Pour cela, nous utilisons et adaptons le patron d'analyse "Item-description" de

P. Coad (voir chapitre II - §3.2.2.3). La Figure 4.3 présente le modèle obtenu. Ce modèle illustre un cas général d'un produit générique *pg1* avec un type de produit issu de celui-ci: le *tp1-pg1* ainsi que l'exemplaire physique de tp1-pg1 qui est *mon tp1-pg1*. Par rapport à l'exemple de niveaux de voiture présenté dans §2.1 (Figure 4.1), *pg1* est Peugeot 206; *tp1-pg1* est Peugeot 206-S16 et *mon tp1-pg1* est ma peugeot 206-S16. Par ailleurs, des méthodes de consultation dans les classes proposées permettent de propager une valeur d'une propriété à travers les associations entre les deux classes.

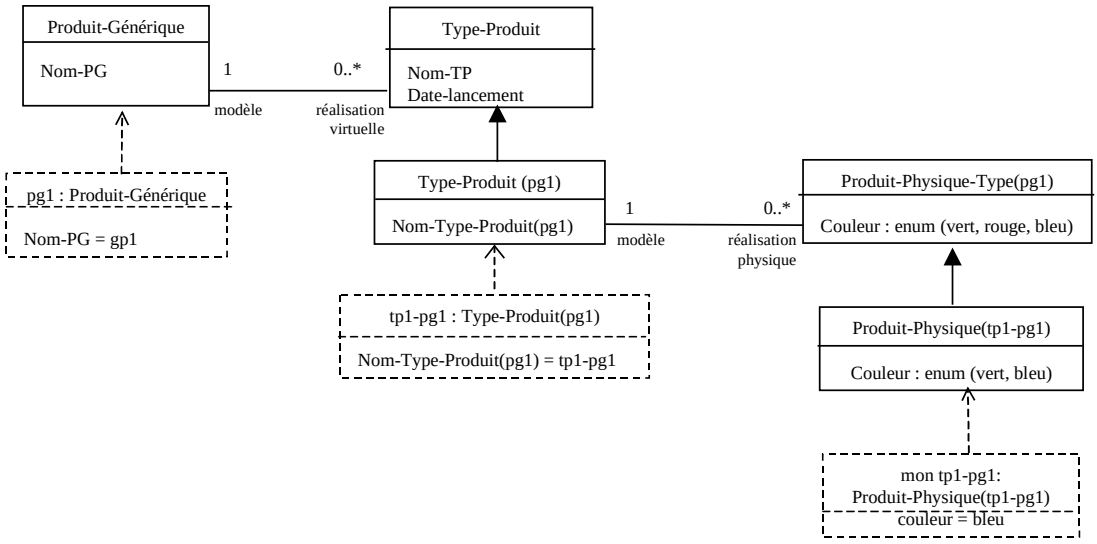


Figure 4.3 - Modèle de niveaux de produit

Par rapport à la démarche d'ingénierie de patrons, un tel modèle constitue un bloc de concepts pour représenter l'information sur les niveaux de produit. Dans le SIP, un modèle similaire doit exister pour décrire les niveaux de produit. Toutefois, la structure d'un tel modèle peut varier d'une entreprise à une autre, selon le nombre de niveaux de produit qu'elle gère (il peut contenir une, deux ou trois classes). Dans le catalogue de patrons (chapitre VI), un ensemble de patrons est proposé pour fixer les niveaux de produit gérés, puis construire le modèle associé représentant alors un, deux ou trois niveaux de produit.

Dans le paragraphe suivant, nous passons en revue l'ensemble des concepts gérés dans les SIP pour décrire le produit, à ses différents niveaux.

3. Les Concepts produit gérés dans les SIP

Les concepts relatifs au produit servent d'abord à *définir le produit* **(**à un instant donné) et ensuite à *gérer l'évolution de sa définition* dans le temps.
Un produit quelque soit son niveau (générique, type, exemplaire) possède **un cycle de vie** dans lequel il évolue et passe en conséquent par différents états. Durant tout son cycle de vie et selon l'état dans lequel il se trouve, différentes données ou informations sont attachées au produit permettant ainsi de le décrire ou de le définir : par exemple des **fonctions**, des **nomenclatures** organiques à base d'**articles**, des modèles CAO, etc. La majorité de ces informations sont consignées dans des **documents** tels que les cahiers de charges, les plans, les schémas, les fichiers CAO, etc.

En se référant aux définitions données dans le projet Esprit AIT [AIT, 97a] et par F.L. Krause [Krause, 90] (cf. §3.1.9), la description de ces informations forme le **Modèle** du produit considéré. Nous parlerons plutôt de **représentations** d'informations (une modélisation de ces informations).
Par ailleurs, un produit tout comme une information relative à celui-ci ou une représentation permettant de le décrire peuvent évoluer dans le temps, pour répondre à des besoins différents. Ils subissent des **modifications** d'ampleurs diverses et possèdent par la suite un ou plusieurs indices de **versions** ou de **révisions** qui permettent d'identifier et de distinguer ces évolutions les unes aux autres.
La suite de cette section présente chacun des concepts introduits ci-dessus, relatifs d'une part à sa description et d'autre part à ses évolutions.

3.1. Concepts pour la description du produit

3.1.1. Cycle de vie et états du produit

Tout au long de l'exécution des divers processus rattachés au produit, celui-ci change d'état au fur et à mesure de l'exécution des différentes opérations et décisions prises lors de ces processus. Ces changements déterminent ce qu'on appelle le cycle de vie, marqué par une logique de déroulement d'actions qui amènent à compléter progressivement les connaissances détenues sur le produit. CimDATA définit le cycle de vie comme *la description des phases distinctes par lesquelles passe chaque produit durant sa vie. Ceci inclut les spécifications, la conception, production, maintenance, etc.*[84] [CIMdata, 95].
Divers cycles de vie sont proposés dans la littérature. Ils se différencient par le nombre plus ou moins important des phases qu'ils proposent mais ils convergent sur un ensemble de phases minimales. G. Benchimol [Benchimol, 93] parle de courbe de vie pour les familles de produits (ce qui correspond à l'évolution des produits virtuels) et de cycle de vie pour les instances de produits (ce qui correspond à l'évolution des produits physiques). Le Tableau 4.1 présente chacun de ces deux cycles.

Etapes de la courbe de vie	Etapes du cycle de vie
1. Développement (industrialisation) 2. Introduction (entre développement et lancement) 3. Lancement 4. Point mort (entre lancement et croissance) 5. Croissance 6. Saturation 7. Déclin (rationalisation maximale de la production avant d'effectuer le retrait progressif) 8. Retrait (fin de déclin, après retour au point mort)	1. Développement 2. Fabrication 3. Distribution ou vente 4. Exploitation-maintenance (soutien logistique) 5. Mise au rebut (fin de l'exploitation),

Tableau 4.1 - Courbe de vie et cycle de vie de produit

Dans la suite de ce référentiel, on utilisera uniquement le terme *cycle de vie* pour désigner l'évolution des divers niveaux de produits. Ci-dessous, nous proposons des cycles de vie pour

[84] Traduction de : the description of the distinct phases through which each product passes during its product life. This includes such as requirements definition, concept design, production, operation, maintenance, etc.

les produits virtuels d'une part et pour les produits physiques d'autre part, en se basant sur les propositions faites dans la littérature.

Cycle de vie du produit virtuel : dès l'étape de lancement, le produit virtuel ne subit plus de transformations en matière de connaissances apportées à celui-ci et son état n'évolue pas. Les étapes de lancement, point mort, croissance, saturation et déclin, décrites dans le cycle de vie proposé par Bourdichon, correspondent à une période de stabilité informationnelle, permettant la production d'exemplaires du produit virtuel. Cette période est appelée "exploitation". En s'inspirant des états de produit définis par la recommandation RG Aéro 000 40 du BNAE [BNAE, 90], nous décrivons, dans un graphe d'état UML, le cycle de vie d'un produit virtuel comme suit :

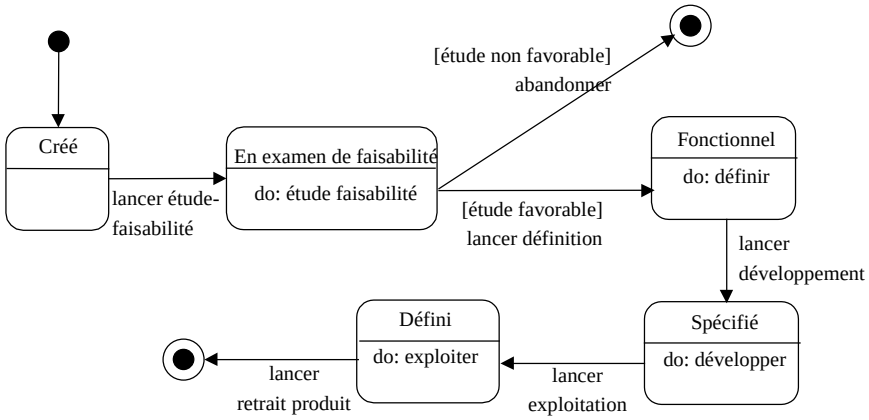


Figure 4.4 - cycle de vie du produit virtuel

Nous soulignons que le « do » dans un état donné exprime les activités que subit le produit à cet état et c'est l'achèvement du « do » qui fait passer le produit à l'état suivant.

Cycle de vie du produit physique : la Figure 4.5 illustre le cycle de vie des produits physiques.

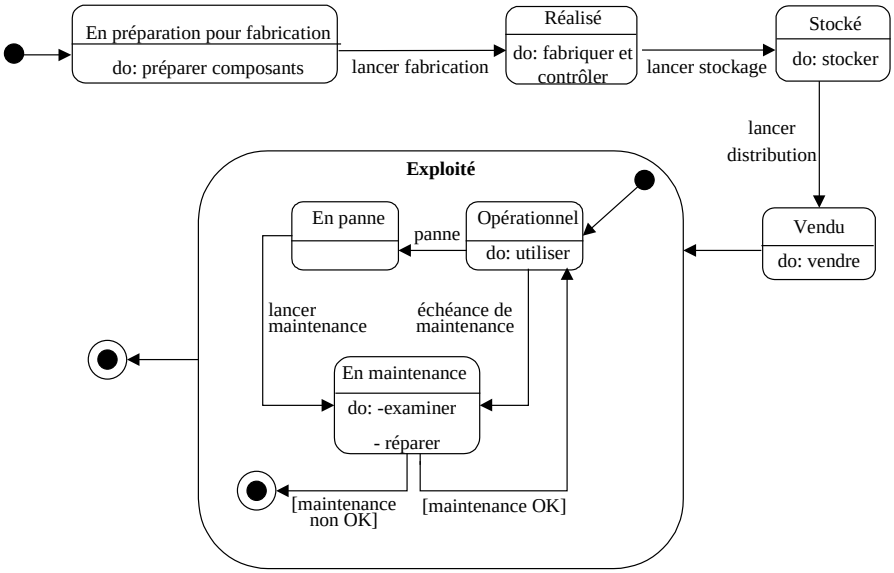


Figure 4.5 - cycle de vie du produit physique

Il est toutefois important de souligner que les cycles de vie présentés ici ne sont que des abstractions des cycles de vie des produits. En effet, selon les processus des entreprises, ces cycles de vie doivent être adaptés et en particulier enrichis.

3.1.2. Objet technique

La définition d'un produit passe par l'utilisation d'entités de définition appelées parfois *objets techniques* décrivant la structure du produit. Une structure est définie par M. Maurino [Maurino, 93] comme "une description des niveaux successifs du produit en objets techniques". Le terme le plus courant pour désigner les structures de produit est le concept de *nomenclatures*.

Par objet technique, est entendu tout élément constitutif d'un produit et qui peut être de nature variée tels que les fonctions, les composants, les entités géométriques (*features*), etc. Il n'existe pas une structure unique du produit puisqu'elle diffère selon le stade du cycle de vie du produit. En fonction de la phase du cycle de vie mais également du métier de l'entreprise ou service considéré, le produit est perçu selon différents *points de vue* qui déterminent la nature des éléments de structuration du produit. Ainsi, lors de l'expression fonctionnelle du besoin, le produit est abordé en terme de fonctions. Durant la conception détaillée, il est défini au moyen d'organes physiques, de pièces, etc. Par ailleurs, le découpage organique d'un produit par une équipe du bureau d'étude s'effectue différemment de celui effectué par une équipe de fabrication. Nous reviendrons sur le concept de point de vue dans le §3.1.9.

Diverses structures de produit peuvent donc exister dans l'entreprise. Les plus courantes sont celles préconisées dans la norme STEP (partie 214) qui propose deux découpages ou structures : le découpage fonctionnel et le découpage organique. Nous étudions de près ces deux découpages.

Concernant le découpage fonctionnel, le terme le plus souvent utilisé est celui de fonction. Une *fonction* représente une action attendue d'un produit ou d'un de ses constituants exprimée exclusivement en termes de finalité, en faisant abstraction de toute référence à des solutions [Boulet, 95].

Concernant le découpage organique, les modèles rencontrés dans la littérature pour décrire la structure des produits utilisent plusieurs termes. Les termes les plus souvent utilisés sont composant[85], pièce[86], élément[87], ensemble[88], sous-ensemble[89], système[90], sous-système[91], mécanisme[92], assemblage[93], etc.

---

[85] L'ISO [ISO, 94a] définit le composant comme un produit ne faisant pas l'objet d'une décomposition du point de vue application spécifique.

[86] L'ISO [ISO, 94c] décrit une pièce comme un article qu'il est prévu de produire ou d'utiliser dans un processus de production

[87] Défini par l'AFNOR [Afnor, 85] comme une partie constitutive d'un ensemble ou d'un sous-ensemble, quelle qu'en soit la nature ou la dimension (NF X 11-500).

[88] Défini comme un groupement de sous-ensembles assurant une ou plusieurs fonctions techniques qui le rendent apte à remplir une fonction opérationnelle [Saucier, 97].

[89] Un regroupement d'éléments associés en fonctionnement et entrant dans la composition d'un ensemble [Saucier, 97].

[90] Association de sous-systèmes constituant un tout organique complexe destiné à remplir une fonction [Saucier, 97].

[91] Une association de composants destinés à remplir une ou plusieurs fonctions opérationnelles au sein d'un système [Saucier, 97].

[92] Ensemble de pièces ou d'organes liés mécaniquement ou électromécaniquement et dont certains sont mobiles [Saucier, 97].

[93] Ensemble de pièces immobiles les unes par rapport aux autres [Saucier, 97].

Certains de ces termes correspondent à des décompositions selon des critères logiques (tel que composant, sous-système, système), d'autres à des décompositions selon des critères matériels (tel que pièce, mécanisme). Dans un contexte de gestion de données produit, parmi l'ensemble de ces termes, nous retenons le critère de gestion proposé par l'AFNOR pour décrire les produits par le biais du concept d'*article* qui représente un élément de nomenclature. Nous proposons de regrouper tous les termes présentés ci-dessus au sein d'une entité unique appelée "article". En fonction du contexte, un article pourra être une pièce, une matière première, un ensemble, etc.
Dans la suite, nous examinons davantage les concepts introduits ci-dessus. Nous étudions le concept de *nomenclature* ainsi que les éléments de base les plus utilisés dans les nomenclatures c'est à dire *fonction, article* et *feature*. Nous étudions enfin les concepts de *document, modèle de produit* et *point de vue*.

3.1.3. Nomenclature

Quelques Définitions

- La nomenclature désigne une liste ordonnée de matériaux, de composants, de sous-ensembles ou d'assemblages qui décrivent un produit. Normalement créée et maintenue comme fonction particulière de gestion de configuration de produit, elle définit la nature, le numéro de référence, le nombre de composants utilisés, ainsi que les relations entre composants et assemblages [CIMdata, 95].
- La nomenclature concrétise la relation composé-composant. Une nomenclature est l'ensemble hiérarchisé des informations nécessaires et suffisantes à la décomposition d'un produit en vue de sa fabrication [Bourdichon, 94].
- Une nomenclature est une liste de tous les sous-ensembles, les pièces et les matières premières qui vont être assemblés dans un ensemble [PDMIC, 98].

Dans ces définitions, l'accent n'est mis que sur la décomposition organique du produit. La notion de nomenclature trouve en effet ses origines dans la gestion de production. Toutefois, le concept de nomenclature peut être étendu pour désigner toute décomposition d'un objet. Nous définissons alors une nomenclature comme une *arborescence d'objets constituée par des nœuds de même nature et ayant des liens de composition*[94] *entre eux.*
Les nomenclatures décrivent le produit d'un point de vue particulier, qui peut être :

- sa fonction: la nomenclature fonctionnelle représente les niveaux successifs de décomposition du produit en *fonctions*.
- sa géométrie : la nomenclature géométrique représente la décomposition du produit en termes d'*entités géométriques* (ou features).
- sa définition : la nomenclature technique (ou d'études) est la représentation hiérarchisée des *organes matériels* constituant le produit, définie lors de la conception.
- son processus de production: la nomenclature industrielle (ou de production) est établie en fonction des étapes successives d'élaboration du produit à partir de *matières*, *composants*, *sous-ensembles*. On distingue la nomenclature de programmation (incluant les quantités correspondants à chaque variante d'un produit donné) et la nomenclature d'assemblage

[94] Connu aussi dans la littérature sous les noms de relation « meronymic », « co-subsomption », « est composant de », « est partie de ». Un lien de composition lie un objet dit composite à ses composants où le composite est une agrégation de composants qui décrivent chacun une partie de l'objet composite [Oussalah, 97a].

(donnant la constitution précise d'un produit). La structure industrielle prend en compte l'organisation de la production et elle est construite en fonction des flux matières (un sous-ensemble n'est pas identifié en tant que niveau dans la nomenclature quand il n'est pas sujet à un stockage ou que sa fabrication est synchronisée avec celle de l'article de niveau supérieur).

- son processus de mise en œuvre et de maintenance (nomenclature logistique) : la fonction SAV doit tenir les références des sous-ensembles dont l'usure ou la destruction fait partie du cycle de vie du produit et qui ont été étudiés pour être remplacés. Ainsi dans cette vision, les articles sont considérés comme étant des *éléments de soutien logistique* hiérarchisés en fonction d'un échelon de maintenance du produit. Les liens dans cette structure peuvent être du type : démontable / non démontable, démontable directement / démontable indirectement.

Selon l'entreprise, ces points de vues sont partiellement ou totalement gérés. La typologie et le nombre des nomenclatures associées y varient aussi.
Soulignons que les nomenclatures technique, industrielle et logistique sont toutes décrites à l'aide d'articles (appelés ici différemment : organes matériels, matières, composants, sous-ensemble). Ces dernières sont souvent appelées des nomenclatures organiques. Notons toutefois que la plupart des SGDT présents sur le marché supportent uniquement les nomenclatures organiques. Nous pensons qu'il est tout à fait envisageable que les SGDT gèrent également les nomenclatures fonctionnelles ou encore les nomenclatures géométriques.

En terme de **structuration**, le modèle de nomenclature doit représenter une composition récursive d'objets techniques (articles pour les nomenclatures organiques, fonctions pour les nomenclatures fonctionnelles, etc.). Il doit permettre de définir des hiérarchies d'objets simples et d'objets composites et faciliter l'ajout de nouveaux composants. Pour cela, nous utilisons et adaptons le patron de conception "Composite" d'E. Gamma [Gamma, 95].

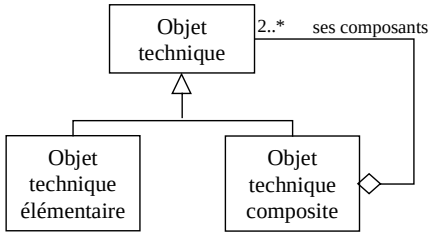


Figure 4.6 - Modèle de composition récursive d'objets techniques

Nous étudierons dans ce qui suit les entités de base constituant les nomenclatures fonctionnelle, organique et géométrique, c'est à dire *Fonction, Article* et *Feature*. Nous déclinons également les différentes structurations des nomenclatures associées.

3.1.4. Les Fonctions

Dans le cas où l'organisation industrielle adopte la conception fonctionnelle, le produit est défini en termes de besoins. C'est généralement suite à une analyse fonctionnelle[95], que le produit est défini en termes de fonctions. Plusieurs définitions sont données à ce concept

[95] Démarche qui consiste à recenser, caractériser, ordonnancer, hiérarchiser et valoriser les fonctions [Afnor, 90].

[Afnor, 90] [PDMIC, 98]. L'ensemble de ces définitions s'accorde sur le fait qu'une fonction est *une expression d'un service attendu par un article.*

Deux types de fonctions peuvent être distingués :

- des fonctions constitutives du besoin, exprimant les services attendus d'un produit. Les *fonctions de service* expriment "ce que fait" le produit pour répondre à un besoin d'utilisateurs. Elles traduisent les rapports du produit avec son environnement [Maurino, 93]. Par exemple, "dépoussiérer un local" est l'une des fonctions d'un aspirateur. Plusieurs fonctions de service peuvent être nécessaires pour satisfaire un même besoin.
- des fonctions internes au produit pour accomplir les fonctions de service. Les *fonctions techniques* expriment "comment le produit fait ce qu'il fait" ou encore comment font ses composants pour assurer les fonctions de service. Par exemple la fonction "aspirer l'air à l'aide d'une pompe" pour un aspirateur. Les fonctions techniques d'un constituant du produit peuvent être vues par le concepteur comme les fonctions de service de ce constituant.

Une fonction qu'elle soit de service ou technique peut être élémentaire ou composée. Une fonction composée nécessite une décomposition en des fonctions plus simples et de même type qu'elle. Par ailleurs, à chaque fonction de service sont associées les diverses fonctions techniques qui permettent de l'accomplir.

En terme de **structuration**, deux critères de classification interviennent donc pour décrire les fonctions : une fonction est soit de service soit technique. Dans chacun de ces cas, la fonction est soit élémentaire soit composée. Par ailleurs, une fonction composée (de type service ou technique) est composée de fonctions de même type qu'elle. La composition est alors définie pour chaque type de fonction. Par ailleurs, à chaque fonction de service sont associées une ou plusieurs fonctions techniques qui contribuent à la réalisation de la fonction de service. Le modèle proposé est alors comme suit :

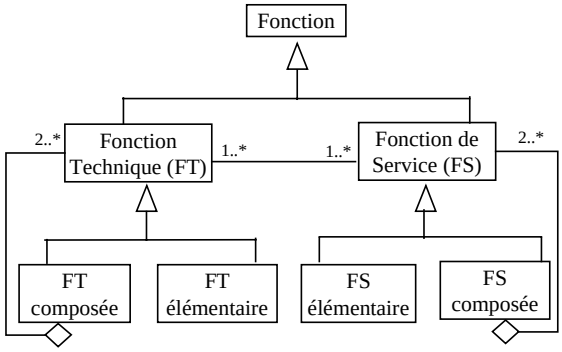


Figure 4.7 - Modèle de Fonction

Pour représenter la composition de fonctions, et de la même façon que pour la composition d'objets techniques, nous avons utilisé et adapté le patron de conception "composite" de E. Gamma [Gamma, 95].

3.1.5. Les Articles

Les diverses interprétations du concept d'article permet de structurer le produit selon plusieurs vues métiers (d'où les différentes nomenclatures). En effet, pour le métier étude, l'article est un élément de définition du produit (il est à chaque niveau de décomposition du produit

pertinent d'un point de vue technique). Pour le métier production, l'article est un élément de définition du processus de production. C'est l'ensemble des composants, matières et sous-ensembles à fabriquer ou à approvisionner. Pour la fonction SAV, l'article est un élément de définition du processus de soutien logistique. C'est l'ensemble des rechanges, regroupés en lots, des sous-ensembles et des composants démontables.
Plusieurs définitions sont données au concept d'article [BNAE, 95] [Bourdichon, 94]. Certaines d'entre elles sont parfois contradictoires. Nous retenons la définition de l'AFNOR [Afnor, 82][96] qui présente un article comme *un bien identifié en tant que tel, constituant un élément de nomenclature ou de catalogue*.
Par ailleurs, un article peut être une option ou une variante dans une nomenclature. Dans ce qui suit, nous étudions successivement la composition, la variabilité et l'aspect optionnel des articles.

1.1.1.14. Composition d'articles

Un article est un élément constitutif de produit. Il se représente donc à tous les niveaux de décomposition du produit et s'étend du produit lui-même (le niveau le plus élevé dans la décomposition d'un produit en articles) jusqu'à la pièce élémentaire (ne présente aucun intérêt à être décomposé). Un article peut être soit une pièce, un ensemble inséparable, un composé externe ou un composé interne. Les trois premiers cas correspondent à un article catalogue, géré en tant que tel (l'intérêt de gérer les articles composant un ensemble séparable ou un composé externe ne se pose pas par exemple). On distingue donc deux types d'articles : *l'article catalogue* et *l'article composé* (en interne).

En terme de **structuration**, le modèle de composition récursive d'articles doit représenter, outre la composition récursive d'articles (cf. Figure 4.6), l'incompatibilité entre certains articles. Il s'agit d'incompatibilités intrinsèques aux articles considérés, indépendamment des produits dont ils sont composants (tels que l'incompatibilité de matière). Une relation réflexive sur la classe Article est alors définie, associant à un Article l'ensemble des Articles qui ne lui sont pas compatibles.
Nous obtenons le modèle suivant :

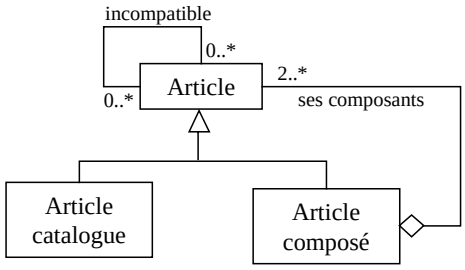


Figure 4.8 - Composition d'article

En terme de propriétés du lien de composition, diverses caractéristiques peuvent y être attachés. Nous citons en particulier la "date d'effectivité" du lien de composition et le "repère topologique". Ce dernier indique la position de l'article dans le plan d'ensemble du composé.

[96] La norme NF X60-012 : termes et définitions des éléments constitutifs et de leurs approvisionnements pour les biens durables.

Il est important dans le cas d'un produit composé de plusieurs instances d'un même article. En terme de sémantique du lien de composition, nous présentons dans l'Annexe B la sémantique du lien de composition dans les nomenclatures les plus communes.

1.1.1.15. Variabilité d'article

Les définitions du concept de "variante" proposées dans la littérature sont parfois contradictoires. Si la majorité des auteurs conviennent sur le fait que les variantes sont perçues comme des variétés de produit résultant d'une diversification à partir d'une conception de base commune[97], ils divergent sur la définition même de la variante. Alors que certains auteurs [BNAE, 90] considèrent que les variantes présentent entre elles des différences de définition, sans incident sur la fonction prévue par l'utilisateur, d'autres auteurs [Maurino, 93] considèrent que les variantes présentent entre elles des différences de définition pour satisfaire des besoins différents de l'utilisateur (et donc ils répondent à de fonctions différentes pour l'utilisateur).

Tel que nous avons défini les divers niveaux de produit, les variantes permettent de décliner divers types de produit à partir d'une conception de base ; celle du produit générique. Les variantes définies au niveau générique se présentent donc comme des alternatives d'un composant donné dans le produit pour satisfaire des fonctions différentes. Les types de produit ainsi déclinés assurent des fonctions différentes pour l'utilisateur.
Ainsi, si l'on considère le produit voiture peugeot 206, ses composants au niveau de la propulsion peuvent être un moteur essence, un moteur électrique ou un moteur thermique. Ces trois composants sont des variantes du moteur de la voiture peugeot 206. Dans cet exemple, nous avons introduit à tort un autre élément outre les trois variantes : l'élément "moteur". Les variantes dans la composition d'un produit ne peuvent donc être définis de façon indépendante. Elles sont définies par rapport à un composant du produit, que nous appelons article à variantes. Un article à variantes est un composant "abstrait" du produit, dans le sens où il n'existe pas un exemplaire physique de cet article qui peut être inclus dans l'exemplaire d'un type de produit donné. Ainsi, dans l'exemple présenté, aucun type de voiture peugeot 206 (peugeot 206-S16 ou peugeot 206-LXR ou autre) ne peut être composé d'un article qui s'appelle "moteur". Chaque type de produit est composé obligatoirement d'une variante de "moteur" (ainsi, le type de voiture peugeot 206-S16 comprend un moteur essence).
Remarquons par ailleurs que les variantes d'un article à variantes assurent une même fonction dans le produit ; celle qui est prévue pour l'article à variantes auquel elles sont associées (la fonction "propulser la voiture" dans le cas du moteur de voiture) mais assurent par ailleurs d'autres fonctions qui diffèrent d'une variante à l'autre. Ainsi, le moteur essence par exemple assure des fonctions du type "transformer l'essence en énergie mécanique" et "transmettre l'énergie mécanique aux roues" alors qu'un moteur électrique n'assure pas la fonction "transformer l'essence en énergie mécanique" mais plutôt la fonction "transformer l'énergie électrique en énergie mécanique" et assure par ailleurs d'autres fonctions différentes de celles du moteur essence.

---

[97] CIMdata par exemple définit les variantes comme étant " des articles utilisés dans la structure du produit pour indiquer un ensemble d'alternatives dans la conception qui conduisent à produire des produits différents, par exemple un voiture à moteur 4 cylindres ou une voiture à moteur 6 cylindres. Les variantes représentent un ensemble de variations qui évoluent dans des versions cohérentes avec le reste du produit" [CIMdata, 95].

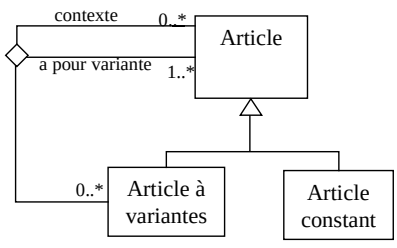
Par rapport à la typologie du concept "Fonction" présentée au §3.1.4, nous pouvons déduire que la fonction assurée par un article à variantes est également assurée par ses variantes. Par ailleurs, les variantes assurent diverses autres fonctions qui correspondent à des composantes différentes de la fonction commune associée à l'article à variantes. Ainsi le moteur électrique et le moteur à essence assurent tous les deux la fonction commune "propulser la voiture" (c'est la fonction de l'article à variantes "moteur") mais assurent des fonctions composantes de celle-ci différentes : le moteur essence assure ainsi la fonction composante "transformer l'essence en énergie mécanique" et la moteur électrique assure plutôt la fonction composante "transformer l'énergie électrique en énergie mécanique".

Par opposition à la notation "articles à variantes", nous désignons par "article constant" les articles qui ne possèdent pas de variantes (et donc qui sont des articles "concrets", ayant obligatoirement un exemplaire physique, inclus dans la composition de l'exemplaire d'un type de produit donné).

A un article à variantes sont associées donc des articles qui sont ses variantes. Ces variantes peuvent être à leur tour des articles à variante ou des articles constants. Il est possible en effet d'avoir une hiérarchie de variantes. Par exemple, la variante moteur essence peut à son tour avoir les variantes essence turbo 2l, essence 1.8 injection, essence 1.4, etc.
Soulignons enfin qu'un article n'est une variante d'un article (à variantes) que dans un contexte donné c'est à dire par rapport à un article (donc un produit) donné dans la composition duquel il entre. Il ne peut exister en soi en tant que variante. Il est toujours défini par rapport à l'article (le produit) dans la composition duquel il entre. Par exemple l'article *moteur diesel 2l* peut être une variante de l'article *moteur diesel* dans la composition des voitures peugeot 506 mais il n'est pas une variante de ce même article (moteur diesel) dans la composition des voitures peugeot 206.

En terme de **structuration**, le modèle de variabilité doit donc permettre de :
- gérer une hiérarchie de variantes de façon à pouvoir éliminer automatiquement certaines variantes lorsque les variantes de niveau supérieur ne sont pas retenues,
- exprimer le contexte dans lequel un élément est une variante d'un autre élément. Ceci permet d'associer à une variante donnée toutes les autres variantes qui lui sont compatibles.

La Figure 4.9 propose une modélisation de ces notions.



•Les variantes d'un Article à variantes sont d'autres Articles qui eux même peuvent être à variantes ou constants. Ceci permet de gérer plusieurs niveaux de variantes et facilite la gestion de ces variantes (lors de la gestion des configurations) en permettant l'élimination automatique de certaines variantes dès qu'un choix à un niveau supérieur de variantes a été effectué.

•Un article est défini comme une variante d'un certain article à variantes dans un contexte particulier. L'association créée entre Article et Article à variantes doit expliciter le contexte de telle variance, à l'aide d'une troisième branche qui explicite le contexte de la variance et qui n'est autre qu'un Article (l'article dans lequel il est composant).

Figure 4.9 - Variabilité d'articles

1.1.1.16. Article optionnel

Un article peut être obligatoire ou optionnel dans la structure d'un produit donné. Les articles optionnels sont des composants qui peuvent ou non exister dans le produit et qui conduisent à la fabrication de produits différents. Ils sont conçus pour satisfaire une ou plusieurs fonctions supplémentaires dans le produit. Un article optionnel, tout comme un article obligatoire, peut avoir des variantes et donc s'il est retenu dans la structure d'un type de produit donné ou d'un produit physique, un choix parmi ses variantes s'impose.
Soulignons enfin qu'un article n'est optionnel que dans un contexte particulier c'est à dire par rapport à un article (donc un produit) donné dont il est composant. Par exemple, le composant "climatiseur" est optionnel dans le produit peugeot 206 mais il est obligatoire dans la composition d'un véhicule frigorifique.

En terme de **structuration**, le modèle de composition avec options doit :
- exprimer le contexte dans lequel un article est optionnel. Ceci permet d'associer à une option donnée l'ensemble des options ainsi que les variantes qui lui sont compatibles, en cas de produit à options et à variantes,
- exprimer qu'un composant d'un élément composite est optionnel ou obligatoire.

La première contrainte se traduit par l'introduction de propriétés supplémentaires sur le lien de composition entre un article composite et ses composants (cf. Figure 4.10). Une propriété du lien de composition exprime le caractère optionnel du composant.
La deuxième contrainte rend nécessaire la représentation du contexte d'un composant optionnel. Ceci se traduit par l'ajout d'une troisième branche dans le lien de composition, reliant la classe Article composé à la classe Article.

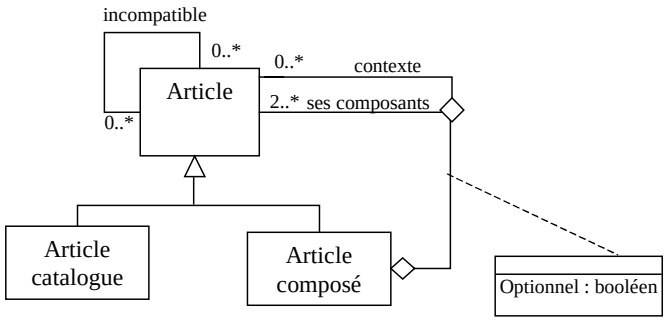


Figure 4.10 - Composition d'articles optionnels

1.1.1.17. Variabilité et Composition d'article

Deux classifications d'article existent donc ; l'une par rapport à la variabilité de l'article et l'autre par rapport à sa composition. Un problème se pose concernant l'ordre dans lequel sont exécutées ces classifications. Les règles suivantes sont à respecter :

1. Deux variantes d'un article (quand elles sont des composées) ne sont pas nécessairement décomposées des mêmes articles. La décomposition ne doit donc pas être exprimée au niveau de l'article à variantes correspondant. Elle doit plutôt être exprimée au niveau de la variante d'un article. En menant ce raisonnement d'une façon récursive (la variante peut être à son tour un article à variantes et donc la composition ne peut pas être exprimée au niveau de cette variante mais au niveau de ses variantes, et ainsi de suite), la composition

doit être définie au niveau des articles constants (quand toute la hiérarchie des variantes est déployée). Nous choisissons donc de spécialiser en premier lieu la classe Article en fonction de la caractéristique de variance. Article a pour sous-classes Article à variantes et Article constant. La classe Article constant est à son tour spécialisée en fonction du critère de composition (cf. Figure 4.11).
Cette solution paraît convenable pour construire des classifications comportant des critères indépendants faciles à ordonner, tel est le cas pour les critères de variabilité et de composition[98].

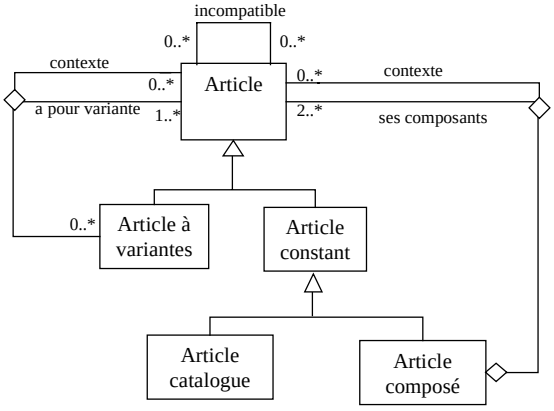


Figure 4.11 - Premier modèle d'article

2. Le schéma conceptuel présenté en Figure 4.11 ne permet pas toutefois de mettre en évidence les composants communs aux différentes variantes d'un même article. Pour illustrer ce cas, considérons l'exemple de la Figure 4.14. Cet exemple est illustré en utilisant le formalisme de FODA [Kang, 90] ci-dessous décrit :

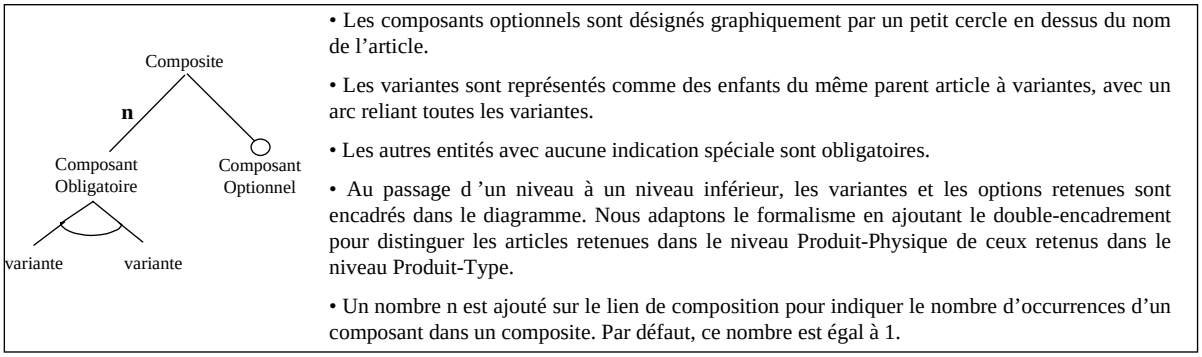


Figure 4.12 - Formalisme de FODA

[98] Il n'existe pas en effet des règles du type : lorsqu'un article est composé, il doit être constant ou encore lorsqu'un article est à variantes, il doit être composé, etc.

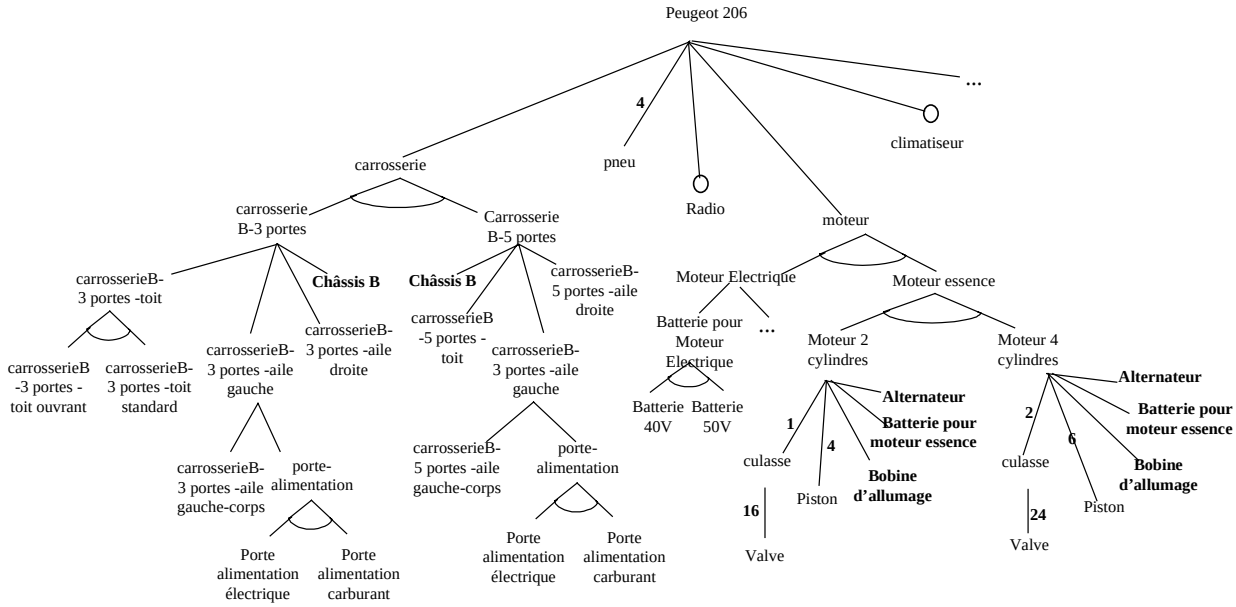


Figure 4.13 - Nomenclature générique de Peugeot 206

Sur cet exemple, le composant "châssis B" est commun aux deux variantes de "carrosserieB". Il serait donc souhaitable de l'associer directement à l'article "carrosserieB" plutôt qu'à chacune de ses variantes. Le problème est identique pour les composants "alternateur", "batterie pour moteur essence" et "bobine d'allumage" qui devaient être représentés une seule fois au niveau de l'article à variantes "moteur essence".
Pour éviter ces redondances d'informations, le schéma conceptuel doit permettre d'exprimer la décomposition d'un article à variantes (qui a différentes variantes) aussi bien au niveau de cet élément à variantes (on n'exprime à ce niveau que les composants communs à toutes ses variantes) qu'au niveau de ses variantes (on exprime à ce niveau les composants spécifiques à chaque variante). Les variantes peuvent par ailleurs "hériter" la composition de l'article à variantes auquel ils sont associés[99].
Nous notons toutefois que seuls les composants directs et communs à toutes les variantes d'un article à variantes peuvent être exprimés comme composants de ce dernier. En se rapportant au même exemple de la Figure 4.13, le composant "porte alimentation" est commun à "carrosseireB-3portes-aile gauche" et " carrosseireB-3portes-aile droite", qui sont des composants des variantes de l'article "carrosserieB" et non pas des variantes de ce dernier. On ne peut dans ce cas exprimer l'article "porte-alimentation" comme composant de "carrosserieB" ; cela fausse la décomposition du produit considéré.

Ainsi, le modèle d'article proposé est comme suit :

[99] Il s'agit en fait d'un héritage par délégation, via l'association entre les objets variante et article à variante.

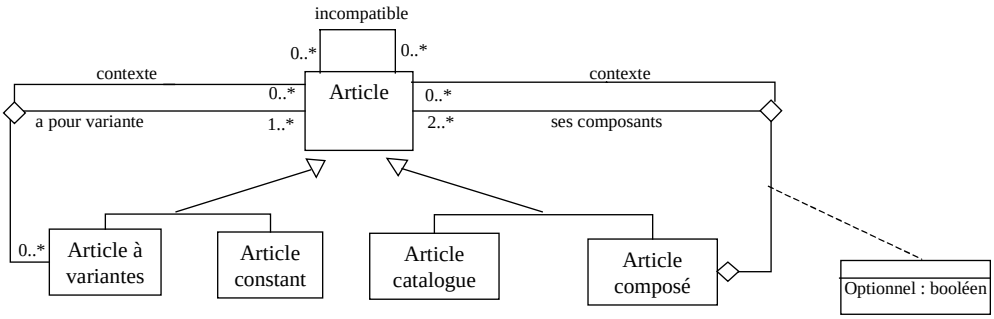


Figure 4.14- Modèle d'article

3. Le modèle d'article doit permettre aussi la déclinaison de types de produit à partir de produits génériques et de produits physiques à partir de types de produit. Le passage du produit générique aux types de produit associés et ensuite du type de produit aux produits physiques associés est effectué selon les options et les variantes. Le produit générique a une structure avec au moins deux variantes non interchangeables c'est-à-dire que le choix d'une des deux variantes contraint le choix de variantes d'autres articles à variantes et a un impact sur la structure globale du produit. Le type-produit par contre est soit composé de variantes interchangeables soit ne présente aucun choix de variantes. Ainsi,

- Le passage d'un produit-générique aux types-produit est effectué en fixant certaines variantes pour chaque article à variantes. Dans le cas particulier de variantes non interchangeables, une des variantes considérées est obligatoirement choisie. Par ailleurs, le choix d'une variante d'un article à variantes donné contraint le choix de variantes d'un autre article à variante.
- Le passage du type-produit aux produits physiques est effectué en fixant une variante parmi un ensemble de variantes interchangeables.

Pour illustrer ces mécanismes, nous considérons l'exemple de la Figure 4.13. Par un choix de certaines options et variantes dans la structure présentée, nous obtenons divers types de peugeot 206. A titre d'exemple, le type peugeot 206-S16 (cf. Figure 4.15) est composé d'une "carrosserieB-3portes", il a un "moteur essence à 4 cylindres" et peut être composé d'une "radio". Notons par ailleurs que le choix de la variante "moteur essence 4 cylindres" implique qu'on ne peut considérer que la variante "porte-alimentation-carburant" de l'article "porte-alimentation". De même, le choix d'une certaine option peut contraindre ou imposer le choix de certaines autres options ou variantes. Ainsi, si on retient l'option "climatiseur", cela peut imposer le choix de la variante "moteur électrique" de moteur, par exemple.

Remarquons enfin que nous avons "factorisé" les composants communs à l'ensemble de variantes d'un même article (présentés en gras dans la figure).

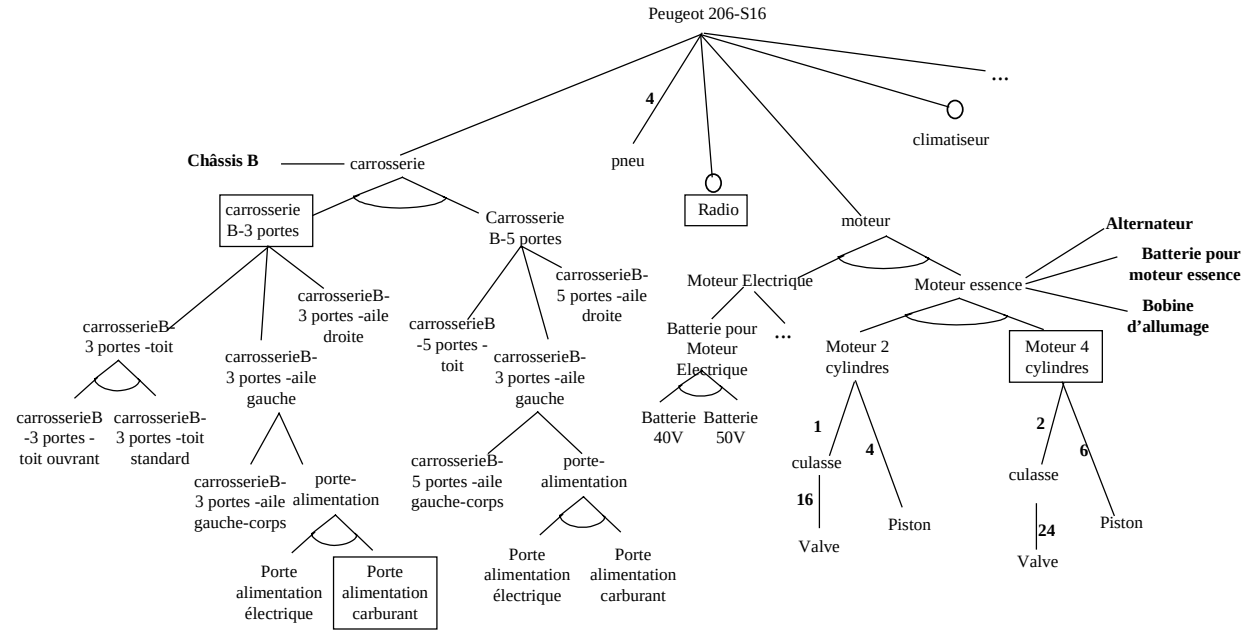


Figure 4.15 - Nomenclature type de peugeot 206-S16

Pour définir un exemplaire particulier de peugeot206-S16, toutes les options et les variantes doivent être fixées. Ainsi, ma peugeot 206-S16 (cf. Figure 4.16) est composée d'une "carosserieB-3portes" à "toit ouvrant" avec une "porte-alimentation-carburant" dans l'aile gauche, un "moteur essence à 4 cylindres", 4 "roues", une "radio", etc.

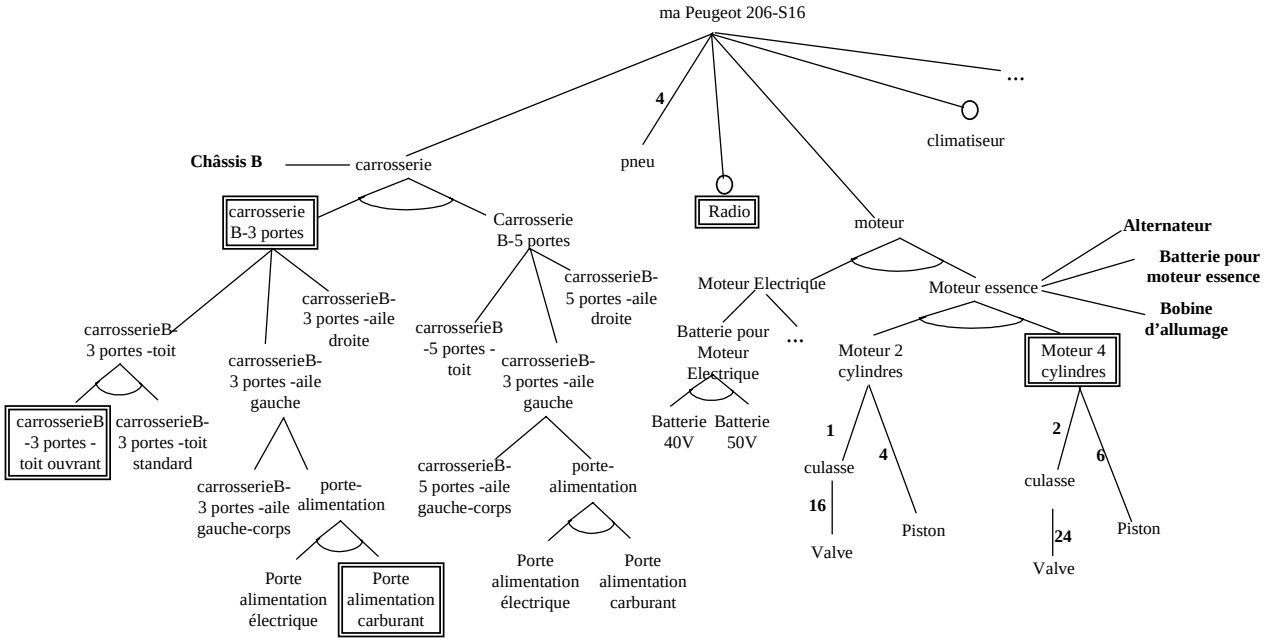


Figure 4.16 - Nomenclature physique de ma peugeot 206-S16

La modélisation proposée dans la Figure 4.14 permet d'exprimer l'ensemble des contraintes sur le choix d'options et de variantes, grâce à la notion de contexte. Deux variantes de deux articles différents (ou une variante et une option) sont compatibles entre elles ou doivent être

associées lorsqu'elles sont définies par rapport au même contexte. Par ailleurs, la propagation des propriétés et de contraintes entre les niveaux est réalisée grâce aux méthodes de consultation dans les classes associées. Ces méthodes permettent de garder visible les connaissances rattachées à un niveau de produit dans les autres niveaux.

Notons néanmoins que le modèle d'article proposé à la Figure 4.14 est valable uniquement pour la description de nomenclatures organiques de produits génériques ou types de produit (donc des produit virtuels) puisqu'un produit physique représente un choix définitif ne présentant aucune option ou variante. Il convient alors de qualifier l'article représenté dans la Figure 4.14 de "article virtuel". Pour la nomenclature organique de produits physiques, nous qualifions l'article associé de "article physique" et le modèle d'article est alors comme suit :

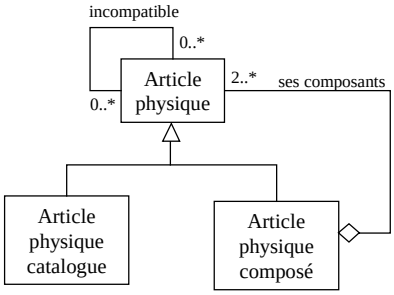


Figure 4.17 - Modèle d'article physique

3.1.6. Les Features

Divers travaux ont porté sur la modélisation des produits (mécaniques) à base de "feature". Ce concept peut constituer désormais le point commun entre les modèles de description des pièces de produits (issus du travail de conception) et les modèles de préparation à la fabrication (issus du travail d'industrialisation). Bien que la notion de feature constitue un point commun entre les métiers intervenant sur la définition du produit, elle est vue différemment par ces divers métiers. De ce fait, plusieurs définitions du concept de feature sont données dans la littérature, en fonction du métier qui les considèrent. Néanmoins quelques auteurs proposent des définitions générales de ce concept. Ainsi, A. Bernard [Bernard, 99] présente les features comme "des ensembles d'informations qui se réfèrent à des aspects de formes et à d'autres attributs du produit, afin de pouvoir raisonner sur la conception elle-même, sur les performances du produit ou sur son processus de fabrication". Le groupe GAMA[100] [Gama, 97] définit un feature comme "un groupement sémantique (atome de modélisation) caractérisé par un ensemble de paramètres, utilisé pour décrire *un objet indécomposable* utilisé dans le raisonnement relatif à une ou plusieurs activités liées à la conception et l'utilisation des produits et des systèmes de production".
Plus simplement, le concept de feature sert à définir le passage de la conception à la réalisation des pièces mécaniques en décrivant les pièces à l'aide de formes géométriques pour lesquelles un processus de réalisation est connu. Un chanfrein, un alésage, une gorge, une rainure, un plan sont des exemples de feature. Comme le souligne la définition de GAMA, les features sont rattachées à des articles élémentaires (cf. §3.1.5).

[100] Affilié à l'AFCET, le groupe GAMA regroupe des chercheurs français travaillant dans le domaine de la conception de gammes d'usinage.

Par ailleurs, plusieurs classifications d'entités sont proposées dans la littérature, selon divers critères (forme, matériau, tolérance, etc.) et orientés selon les divers métiers concernés (conception, préparation de la fabrication, etc.) [Gama, 97]. Il ne s'agit pas ici de retenir une classification particulière; chacune reflète un point de vue métier particulier et/ou présente un critère de classification intéressant. Nous étudions dans ce paragraphe, le lien que pourrait avoir le concept de feature avec les éléments de description du produit dans les SIP.

Les features se trouvent actuellement gérés dans les modeleurs CAO/FAO qui proposent des librairies de features et offrent des possibilités de définition de pièces à base de features (conception par entités[101]). Les concepteurs définissent alors la géométrie nominale[102] de la pièce par instanciation des entités proposées dans les librairies et en donnant des valeurs aux paramètres des entités. Ces modeleurs intègrent, à partir d'une représentation purement géométrique, des modules de reconnaissance de feature pour une application donnée, et permettent, pour certaines, la définition interactive de features [Bernard, 99].
Par rapport aux SIP, actuellement, la géométrie d'ensemble (objet global) des articles et des produits est référencée dans les SGDT. Elle est générée et manipulée dans les modeleurs de CAO/FAO interfacés aux SIP et donc gérée dans le SGDT en tant que document (modèle CAO/FAO).
Comme nous l'avons souligné dans le paragraphe 3.1.3, il est envisageable de gérer une nomenclature géométrique du produit dans le SIP. Il s'agit essentiellement de rattacher à chaque article élémentaire, sa nomenclature géométrique c'est à dire sa décomposition en terme de features. Cette décomposition pouvant être différente selon le point de vue souhaité (conception, fabrication, etc.). Comme pour les articles composant un produit, il s'agit uniquement de référencer les features ; leur génération et visualisation pouvant être gardées dans le modeleur associé. La nomenclature géométrique du produit global s'obtient alors par regroupement des diverses nomenclatures géométriques associées à chacun des articles élémentaires. J. Lin [Lin, 96] parle de feature-composite qui est alors composé de sous-features. Ainsi, une barre filetée est composée d'un barre et d'un filetage extérieur. On peut ainsi parler de nomenclature de feature.

En terme de **structuration**, la nomenclature géométrique est définie comme une composition récursive de features. Nous pouvons donc rattacher à chaque article (élémentaire) une décomposition en features.

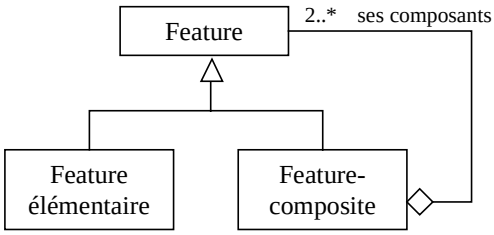


Figure 4.18 - Nomenclature géométrique d'un article élémentaire

[101] A ce titre, plusieurs approches de conception par features sont proposées dans la littérature. Elle constitue une méthode prometteuse pour intégrer CAO/gamme/FAO [Gama, 97].
[102] La forme géométrique globale de la pièce, qui est la même pour tous les intervenants [Gama, 97].

Synthèse : La notion de nomenclature constitue un bloc de concepts, souvent présent dans les modèles des SIP (chaque produit a en effet une structure et donc au moins une nomenclature). Il ressort de l'étude des principaux types d'éléments constitutifs des nomenclatures (fonction, article et feature) qu'il existe diverses constructions des nomenclatures. En effet, il existe dans chaque modèle de nomenclature une partie commune à tous les modèles ; celle relative à la composition récursive d'objets et souvent une partie additionnelle qui varie d'une nomenclature à une autre. Cette partie additionnelle dans le modèle de nomenclature apporte une dimension de variabilité aux nomenclatures, qui peut provenir :

- de la sémantique du lien de composition[103] : un lien de composition peut être partagé ou exclusif, dépendant ou indépendant, etc. Une composition peut être par ailleurs à multigraphe[104] ou à graphe simple[105].
- d'autres éléments tels que la variabilité des articles et l'existence d'articles optionnels dans les nomenclatures organiques, l'existence de fonctions externes et de fonctions internes dans les nomenclatures fonctionnelles, etc.

Dans le catalogue de patrons présenté au chapitre VI, nous proposons un ensemble de patrons permettant la construction de divers types de nomenclatures. Ces patrons résolvent entre autre les problèmes de modélisation mentionnés précédemment, relatifs aux nomenclatures organiques génériques.

3.1.7. Les Documents

Les nomenclatures qui sont attachées au produit permettent d'en donner une première description, sous forme de structures hiérarchiques décrivant la décomposition du produit. Mais cette description est généralement insuffisante. Il est donc nécessaire de compléter la description d'un produit, à tout niveau, par des documents appropriés, tels que des cahiers de charges, des plans, des modèles 3D, des relevés de mesure, etc. Le concept de document est fondamental dans les SGDT puisqu'il est le support de la majorité des informations sur le produit.

Tout au long du cycle de vie du produit, les différents acteurs de l'entreprise produisent ou utilisent des documents : cahiers de charges, résultats d'essais ou de calculs, rapports d'évaluation, dessins de définitions ou fichiers CAO (à la phase d'ingénierie), gammes de fabrication, programmes de machines à CN, fichiers FAO, instructions de fabrication, dessins de montage, comptes-rendus de fabrication, résultats de contrôle (à la phase de production), notices et manuels d'utilisation du produit, planning d'entretien (à la phase d'utilisation), etc.

Cet ensemble de documents et selon le type de l'outil de création peuvent être physiques (papier) ou électroniques (désigné par STEP sous le vocable *digital document)*. L'ensemble des informations contenues dans ces documents sont structurées de manière lisible (texte) ou perceptible (dessin, programme de MO à CN) par une ressource humaine ou matérielle. La production de documents peut être faite par divers outils (planche à dessin, XAO[106], etc.). Du fait de l'extrême diversité de ces outils, le SGDT n'a pas la capacité de se substituer à ces outils et il ne traite donc que le concept de document et non le document lui-même. Il gère la

[103] Nous présentons en Annexe B, les principales sémantiques des liens de compositions dans les nomenclatures produit gérées dans le SIP.

[104] Un composite peut référencer un même composant suivant plusieurs liens de composition.

[105] Un composite ne peut référencer un même composant que suivant un seul lien de composition.

[106] X Assisté par Ordinateur.

référence d'un document, ses liens avec les différents objets gérés dans le SGDT, l'adresse de son stockage (physique ou électronique) et sa relation avec l'outil de production de document, quand il est informatisé. Il gère aussi les différents accès au document en lecture, impression, etc.. Le stockage du fichier même du document (quand il est sous format électronique) est plutôt assuré par l'outil de production du document.

Quelques définitions

Plusieurs définitions du concept de document sont proposées dans la littérature [BNAE, 92] [Maurino, 93] [AIT, 97b] [IPDMUG, 97] [PDMIC, 98]. Nous citons en particulier les deux définitions suivantes :

- Un document est une représentation homogène de données qui communiquent une idée. Les documents physiques sont lisibles par les humains sans une assistance externe. Les documents électroniques sont lisibles à l'aide d'un outil électronique[107] [EDS, 98].
- Le terme 'document' est employé pour désigner tout type de support d'enregistrement de l'information, comme des documents écrits, des documents informatisés stockés sur disque dur, disquettes ou CD-ROM, des cassettes vidéo et audio ou des affiches [ISO, 97].

Un document est donc *un support d'enregistrement de connaissances (un conteneur) qui a pour finalité de décrire en partie un objet technique ou les processus qui lui sont attachés.*

Typologie des documents

Plusieurs typologies de documents sont distingués dans la littérature : selon l'origine des documents (interne, externe), leur aspect contractuel (document applicable, de référence, figé, de configuration, utilisateur, etc.) [BNAE, 90], le type d'information technique contenue (plan, spécification, notice, etc.) [CIMdata, 97] [Randoing, 95], leur finalité (modèle, enregistrement), etc.

Nous retenons en particulier la typologie présentée par l'ISO 9000, basée sur la finalité du document. Cette norme distingue les notions de *document* et d'*enregistrement*. Les *documents* sont ceux utilisés pour décrire ou maîtriser la façon dont les tâches doivent être réalisées et ils peuvent être révisés afin de s'adapter à une situation changeante. Les *enregistrements* par contre sont le résultat d'activités ; ils attestent de l'existence d'un fait à un moment donné et ne peuvent être révisés[108] [ISO, 97]. Nous préférons toutefois utiliser le terme *modèle* pour désigner les *documents* au sens de la norme ISO 9000 et garder le terme document pour désigner l'ensemble des documents, y compris les enregistrements, pour rester conforme aux diverses définitions du concept de document précédemment énoncées. Nous distinguons donc deux types documents selon leur finalité :

- Les modèles : ils visent à définir le produit, l'un de ses processus associés ou à justifier la définition (par exemple, une spécification, un modèle XAO, une note de calculs).
- Les enregistrements : ils visent à enregistrer l'historique des contrôles, des décisions ou des observations effectuées sur le produit.

Nous retrouvons cette typologie dans la définition donnée par l'IPDMUG (International Product Data Management Users Group) où un document comporte deux aspects : les *méta-*

[107] Traduction de : A homogeneous manifestation of data which conveys an idea. Physical documents are human readable without external assistance. Electronic documents are readable with the assistance of an electronic tool.

[108] C'est ce critère de gestion qui a dicté notre choix pour cette typologie. Les deux types de documents impliquent une gestion différente en terme d'évolution.

données qui décrivent le document et son *contenu*. Notons enfin que les documents associés au niveau produit-physique sont essentiellement de type enregistrement et que ceux associés aux niveaux produit-générique et type-produit sont plutôt du type modèle.
Une seconde typologie de document nous paraît intéressante à définir; celle basée sur la dépendance ou non du document à l'objet qu'il documente. Un document peut être de deux natures :

- un document dépendant : il sert à décrire l'objet qu'il documente, tel qu'un plan ou un modèle CAO rattachés à un produit. Ce sont des documents qui sont créés suite à la création de l'objet documenté et qui n'ont un sens qu'avec l'existence des objets qu'ils décrivent. Leur existence est alors liée à l'existence des objets qu'ils décrivent. Un document dépendant ne peut être associé qu'à l'objet qu'il décrit.
- un document non-dépendant : il est utilisé pour documenter l'objet auquel il est associé. Il sert généralement pour définir cet objet, telle qu'une norme ou un cahier de charges rattachés à l'objet. De tels documents existent indépendamment de l'existence des objets auxquels ils sont associés (la suppression de l'objet documenté n'entraîne pas la suppression du document). Un document non-dépendant peut être associé à plusieurs objets du domaine.

En terme de **structuration**, la première typologie de document (document ou enregistrement) est traduite par un attribut de la classe Document qui spécifie le type du document. Cette propriété est utile pour distinguer les documents pouvant être sujet à des modifications (les modèles) des documents qui ne le sont pas (les enregistrements). D'autres propriétés sont essentiels à gérer telles que l'adresse de stockage qui référence le document dans le SGDT au document lui-même, souvent géré et supporté dans l'outil de production correspondant. Le SGDT ne gère en effet que la concept de document (sa fiche signalétique) et non pas le document lui-même. Pour la deuxième typologie (documents non-dépendants et documents dépendants), les deux natures de documents maintiennent des liens différents avec les objets auxquels elles sont associées. Nous définissons ainsi deux classes représentant chacune de ces deux natures de documents. Par ailleurs, nous distinguons à l'intérieur de chaque nature de document, diverses désignations de documents telles qu'une norme, un cahier de charges, un plan, etc.. Nous définissons ainsi diverses sous-classes des classes Document non-dépendant et Document dépendant, portant chacune des propriétés spécifiques au document considéré (cf. Figure 4.19).

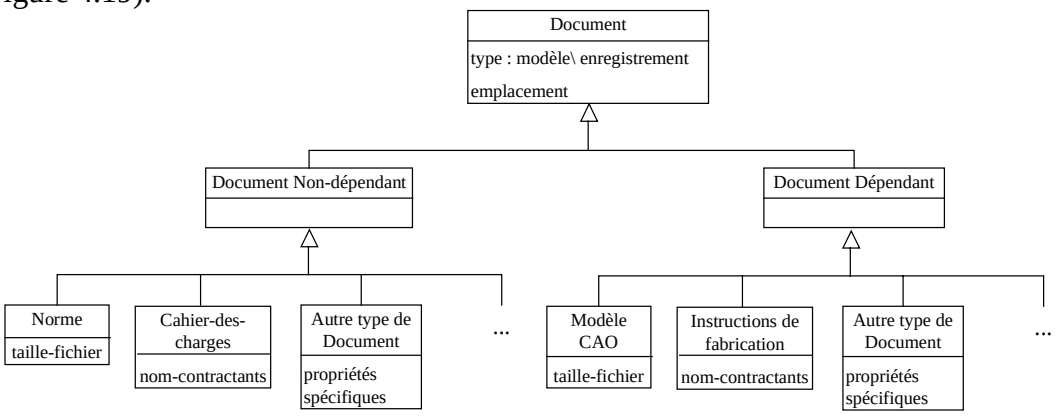


Figure 4.19 - Différentes classifications de Document

Ce modèle constitue également un bloc de concepts relatif aux documents existant dans tout modèle SIP. Sa structure diffère d'une entreprise à une autre, selon les types de documents mis en place.

3.1.8. Les dossiers

La structuration des données produit en documents, articles, fonctions, etc., offre une vision très analytique du produit. Ces différents éléments nécessitent d'être rassemblés en dossiers cohérents pour pouvoir être traités de façon naturelle en situation opérationnelle : on trouve souvent des dossiers de définition d'un ensemble, d'instruction d'une modification, etc. Un dossier (ou une liasse) est donc une vue particulière des données du produit (un rassemblement particulier). Il peut comprendre des documents mais également des nomenclatures, des propriétés, etc. [Maurino, 93]. P. Bourdichon [Bourdichon, 94] présente les dossiers comme une solution pour regrouper les informations acquises au rythme d'avancement du projet produit.
Selon les entreprises, le nombre, la typologie et le contenu des dossiers varient. A chacun des états du produit virtuel ou physique (cf. §3.1.1), sont associés divers dossiers. A titre d'exemple, le produit virtuel peut être documenté par un dossier contractuel à l'état fonctionnel, un dossier d'étude à l'état défini, un dossier industriel à l'état réalisé, etc.
Ainsi, différents types de dossiers sont renseignés dans le SGDT. Chacun décrit de façon générique le contenu d'un dossier constituant le produit d'une activité particulière, dans un domaine spécifique (fonctionnel, technique, industriel, logistique). Ces types de dossier sont définis en terme de types de documents (par type, il s'agit ici de plans de définition, de notes de calcul, de notices d'utilisation, etc.) et de types de nomenclatures (nomenclature d'étude, nomenclature fonctionnelle, etc.). Ils permettent d'homogénéiser le contenu et la présentation des dossiers relatifs aux produits, obtenus par anticipation du type de dossier.
Nous soulignons par ailleurs qu'un dossier peut inclure d'autres dossiers. Par exemple, le dossier de définition d'un produit peut être complété des dossiers de définition de chacun des articles entrant dans sa composition.

3.1.9. Modèle de Produit, Représentations et Points de Vue

Il ressort de la description faite jusqu'ici qu'à chaque état du produit, sont associées des connaissances permettant de décrire cet état. Ces connaissances sont supportées par deux types d'objets : les nomenclatures et les documents. Ce sont donc les deux formes de représentations[109] associées au produit. L'ensemble des représentations d'un produit constitue ce qui est communément connu sous le nom de modèle du produit.
Il existe plusieurs définitions du concept de modèle de produit [Krause, 90] [Harani, 97]. Nous retenons en particulier celle proposée dans le projet AIT [AIT, 97a] où "un modèle produit est défini comme une description complète et cohérente contenant toutes les informations du cycle de vie du produit"[110]. Dans cette définition, il est également souligné

[109] Une représentation est définie dans [Oussalah, 97a] comme "une spécification opérationnelle d'un modèle, très utilisée pour mettre en œuvre des types de données".

[110] Traduction de : A product model is a complete and coherent description of the product containing all the information of the life cycle of the product ….the product model may contain different descriptions of the same characteristics e.g. to fulfil the needs of different applications. The shape may e.g. be defined both by a solid (for design etc.) and by a mesh (for the analysis).

que le modèle de produit peut contenir différentes descriptions de la même caractéristique du produit pour répondre aux besoins de diverses applications. Ainsi une forme peut être décrite par un solide (pour la conception) et par un maillage (pour l'analyse).
Du fait de la diversité des métiers et des acteurs de l'entreprise manipulant les diverses connaissances sur le produit, le modèle produit doit couvrir tous les aspects le concernant. Les représentations traitent ainsi différents aspects complémentaires dans le produit. Les connaissances qui sont décrites dans les représentations sont souvent qualifiées de "multi-facettes" car une variété de représentations est indispensable pour décrire complètement un artefact donné [Katz, 90]. Toutefois chaque métier a des besoins différents en matière d'information sur le produit. Le modèle doit donc pouvoir fournir à chaque métier les seules informations dont il a besoin à travers la notion de vue ou de point de vue que nous présentons ci-après.

Vue et point de vue
Les notions de vue ou de point de vue ont été introduites par plusieurs auteurs [Carré, 89] [Harani, 97] [Rosenman, 96] pour exprimer une perception particulière du produit. Cette perception est souvent faite selon deux axes :
- la logique de déroulement des projets, définissant les phases du cycle de vie du produit,
- la structure organisationnelle de l'entreprise, traduisant les différents métiers intervenant sur le produit.

Plusieurs définitions sont proposées dans la littérature pour ces deux concepts [Muller, 97] [Carré, 89], [AIT, 97a].
Pour la notion de vue, nous citons celle proposée dans le projet AIT [AIT, 97c] qui définit ce concept comme "une perception sélective de l'entreprise mettant l'accent sur des aspects particuliers en négligeant d'autres aspects"[111].
Pour la notion de point de vue, Y. Harani [Harani, 97] précise que la notion de point de vue est définie dans le modèle de produit pour permettre la prise en compte des différentes perceptions qu'ont les concepteurs du produit.
Par ailleurs, certains auteurs ne distinguent pas entre deux concepts. Dans le projet AIT [AIT, 97a] par exemple, la vue est définie à l'aide du concept point de vue.

Cycle de vie des représentations
Tout comme un produit, une représentation possède un cycle de vie dont les grandes phases peuvent être résumées en : la création, l'évolution, le classement, la transmission, l'archivage et la recherche pour consultation.

3.2. Concepts pour l'évolution du produit

Certains des objets gérés dans les SIP (décrits dans le §3.1), après avoir été validés et utilisés dans l'entreprise, peuvent évoluer dans le temps. Nous distinguons deux types d'objets qui évoluent tout au long de l'exécution des divers processus et décisions qui leur sont associés; il s'agit des articles (et donc des produits) et des représentations. Une évolution traduit une modification apportée à ces objets et ce pour diverses raisons tels qu'une évolution du besoin,

[111] Traduction de : A view is a selective perception of the entreprise, which accentuate very much the special aspects and neglect others.

une évolution de la définition du produit pour adapter l'offre ou améliorer la qualité, une évolution des processus agissant sur le produit, etc. Une des principales fonctions des SIP est la maîtrise de ces évolutions. Ainsi, différents types d'évolution sont utilisés pour différencier les objets modifiés des objets initiaux. Cela permet de garder une trace des différentes modifications et d'associer les objets aux bons indices, en associant des indices d'évolution à chacun des objets lors de chaque nouvelle modification.
Dans la littérature, divers termes clés sont associés aux évolutions, tels que *version, révision, correction* ou encore *modification mineure* et *modification majeure*. Il n'y a pas de correspondance unique entre ces divers types d'évolution (version / révision / correction d'une part et modification majeure / mineure d'autre part). En outre, il n'existe pas de consensus sur les définitions des divers types d'évolution. Le même mécanisme (version par exemple) est utilisé différemment dans les entreprises pour désigner des types d'évolution différents (en terme d'impact). D'où l'intérêt d'analyser la sémantique des divers types d'évolution proposés, afin de positionner l'ensemble de ces concepts entre eux. Nous présentons cette analyse dans l'Annexe C. Nous retenons de cette analyse trois mécanismes d'évolution : correction, révision et version. Leur définition varie légèrement selon que l'objet évolutif est un article (et donc un produit) ou une représentation. Le Tableau 4.2 propose une définition de ces mécanismes d'évolution.

Article	**Représentation**
Version : changement de définition du produit qui n'assure pas son interchangeabilité Révision : changement de définition du produit qui assure son interchangeabilité	Correction : modification éditoriale (modification de la forme du contenu telle qu'une faute d'orthographe) sans impact sur les objets qui dépendent de la représentation. Elle peut ou non être diffusée. Révision: modification sémantique (modification du sens du contenu) qui assure l'interchangeabilité de l'objet (initial et modifié) donc sans impact sur l'ensemble des cas d'emploi de l'objet. Une diffusion de la nouvelle révision à une faible échelle suffit. Version : modification sémantique qui n'assure pas l'interchangeabilité de l'objet (initial et modifié) donc ayant un impact sur au moins un cas d'emploi de l'objet. Une diffusion de la nouvelle version à une large échelle est souvent nécessaire.

Tableau 4.2 - Version, Révision et Correction

A chacune des modifications (correction, révision, version) apportées à un article ou à une représentation, un indice est associé pour distinguer les modifications successives.
Pour les représentations, soulignons que seuls les documents du type "modèle" sont versionnables ou révisables puisque les documents du type "enregistrement" supportent des informations qui sont des traductions de faits, des données non modifiables (par exemple si la quantité livrée d'un produit mardi dernier est 100 exemplaires, on ne peut pas revenir en arrière et dire que 200 exemplaires ont été livrés ce jour là).
Soulignons également que lorsqu'une procédure de modification est engagée, la correction, la révision ou la version courante de l'objet sujet à modification reste valide tant que la modification n'est pas validée. Par rapport au cycle de vie d'un produit (cf. §3.1.1) ou d'une représentation (cf. § 3.1.9), c'est la nouvelle révision ou version qui prend l'état "créé" au démarrage d'une procédure de modification.
En terme de **structuration** des concepts d'évolution, les cycles de vie de produit et de représentation présentés respectivement au §3.1.1 et §3.1.9, ainsi que toutes les connaissances

qui leur sont attachées doivent être associés à une version (ou une révision) du produit ou de la représentation. Ceci garantit que toutes les connaissances qui typiquement changent de version en version (ou de révision en révision) soient toujours associées à l'entité concernée.
Ainsi, on peut associer à chaque entité évolutive (article et donc produit, document et nomenclature) une classe relative à la version ou à la révision de l'entité. C'est à cette classe que seront rattachées les représentations des connaissances associées, tels que par exemple les documents et les nomenclatures d'un article (cf. Figure 4.20-a).
Ce raisonnement peut être mené également pour les états des entités évolutives. Lorsqu'il est nécessaire de distinguer les divers états d'un objet (à l'intérieur d'une version ou d'une révision donnée), une classe relative à l'état de l'objet est alors associée à la version ou révision de l'objet considéré. Les représentations des connaissances alors associées à la version ou à la révision de l'objet seront attachées à l'état de l'objet versionné ou révisionné (cf. Figure 4.20-b). Ainsi, dans chacun des modèles proposés précédemment pour structurer chacun des concepts gérés dans le SIP, il est entendu que l'ensemble des éléments du modèle attachés à l'objet considéré sont en fait attachés à une version ou à une révision donnée de l'objet, voire lorsque ceci est nécessaire à un état de celui-ci.

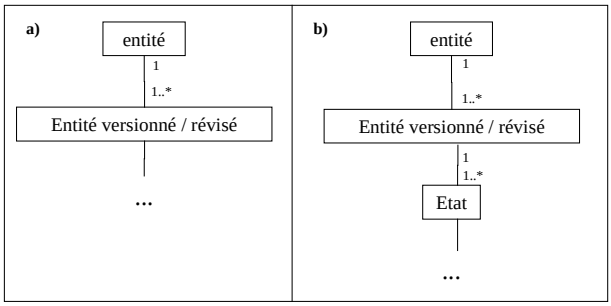


Figure 4.20 - Version et révision d'entités

4. Structuration d'ensemble des concepts produit

A partir des modélisations partielles proposées dans la section 3, nous présentons dans cette section une structuration d'ensemble de ces concepts en mettant en évidence les liens entre les modèles partiels. Lorsque des entités de divers modèles partiels se partagent des connaissances, nous définissons une super-classe à ces entités à laquelle sont attachées ces connaissances partagées.
Il ressort de la description faite dans le §2 et le §3 qu'un produit, quelque soit son niveau (générique, type et physique), est décrit par des représentations. Une représentation peut être soit un document soit une nomenclature.
Chaque niveau de produit est donc documenté par un ensemble de documents. De la même façon, un article, une fonction ou tout autre constituant du produit (objet technique) est également documenté par divers documents (tels qu'un plan de définition pour un assemblage). Nous définissons une super-classe à ces entités (produit, article, fonction) nommée Objet documenté à laquelle est associée chacune des deux natures de documents distingués : Document non-dépendant et Document dépendant ainsi que le modèle partiel associé. L'association entre Objet documenté et Document dépendant est baptisé "décrit". C'est un lien fort qui associe un document à un seul objet documenté. L'association entre Objet documenté et Document non-dépendant est baptisé "documente". Elle associe à un Document zéro ou plusieurs Objets documentés. Il est à noter que l'association reliant les

documents à l'objet associé est baptisée par certains auteurs par *lien de représentation* [Maurino, 93]. Elle porte des attributs de validité (date de création du lien, date de préemption, …).
Par ailleurs, chaque niveau de produit est décrit par diverses nomenclatures. Une nomenclature est une composition récursive d'objets techniques qui peuvent être des articles, des fonctions, des entités géométriques, etc. Une nomenclature concrétise la relation composé-composant. Elle est matérialisée dans le modèle par une association reliant l'objet composé c'est-à-dire l'objet auquel elle est associée tel qu'un produit (à un niveau donné), à la racine du graphe de la nomenclature considérée qui n'est autre qu'un objet technique. Ainsi :

- les nomenclatures organiques décrivant le produit générique ou le type de produit sont basées sur une composition récursive d'articles virtuels. Elles sont attachées au produit à travers une association entre la classe Produit-Générique et la classe Article virtuel composé, correspondant au niveau le plus haut dans la nomenclature associée. Etant donné que plusieurs nomenclatures organiques peuvent exister, chacune d'elles est désignée dans le modèle de composition récursive d'articles par un stéréotype sur le lien de composition selon la nature de cette nomenclature ("nomenclature d'étude", "nomenclature de fabrication", etc.). Il est à noter que ces nomenclatures sont seulement attachées au produit générique. Elles sont déclinées pour les types de produit associés grâce à des opérations de classes portant sur le choix d'options et de variantes.
- les nomenclatures organiques décrivant les produits physiques sont basées sur des articles physiques. Elles sont attachées au produit par une association entre les classes Produit-Physique et Article physique composé.

Pour les nomenclatures fonctionnelles des différents niveaux de produit, celles-ci ne peuvent être directement rattachées au niveau de produit considéré puisque les fonctions d'un niveau de produit varient selon les variantes d'articles qui le composent. Les fonctions (ainsi que leurs décompositions) sont plutôt rattachées aux articles qui les assurent. La nomenclature fonctionnelle d'un niveau de produit est alors déduite des nomenclatures fonctionnelles associées aux articles entrant dans sa nomenclature organique.

Nous avons par ailleurs souligné que l'ensemble des représentations associées à un niveau de produit (documents, nomenclatures) peuvent être regroupées différemment dans différents dossiers. Un dossier est alors un regroupement de représentations qui peuvent être des documents ou des nomenclatures. Pour structurer le concept de dossier, nous distinguons dans le modèle du produit une classe Dossier qui sera liée à la classe Document par un lien de composition (du type élément/collection[112]). Cette classe Dossier est composée de documents, qui documentent entre autre les nomenclatures à travers les objets documentés de type racine d'une nomenclature, tels que la classe Article composé.
Tous les objets décrits dans le référentiel évoluent et possèdent donc différentes versions\révisions et différents états. Nous créons une super-classe à Objet-documenté (qui regroupe articles, fonction, niveaux de produit et feature), Document et Dossier, appelée Elément pour désigner toutes les entités gérées dans le SIP et sujet à évolution. Nous associons à la classe Elément un indice d'évolution (classe Elément versionné\révisionné) et un état (classe Etat d'Elément). En conclusion, le modèle produit obtenu est le suivant :

[112] Dans ce type de relation, les éléments (composants) d'une collection (composé) ne requièrent pas une fonction particulière ou un agencement structurel particulier par rapport aux autres éléments de la collection ou la collection elle même. Des exemples de cette relation sont : un arbre fait partie d'une forêt, un élève fait partie d'une classe, etc [Oussalah, 97a].

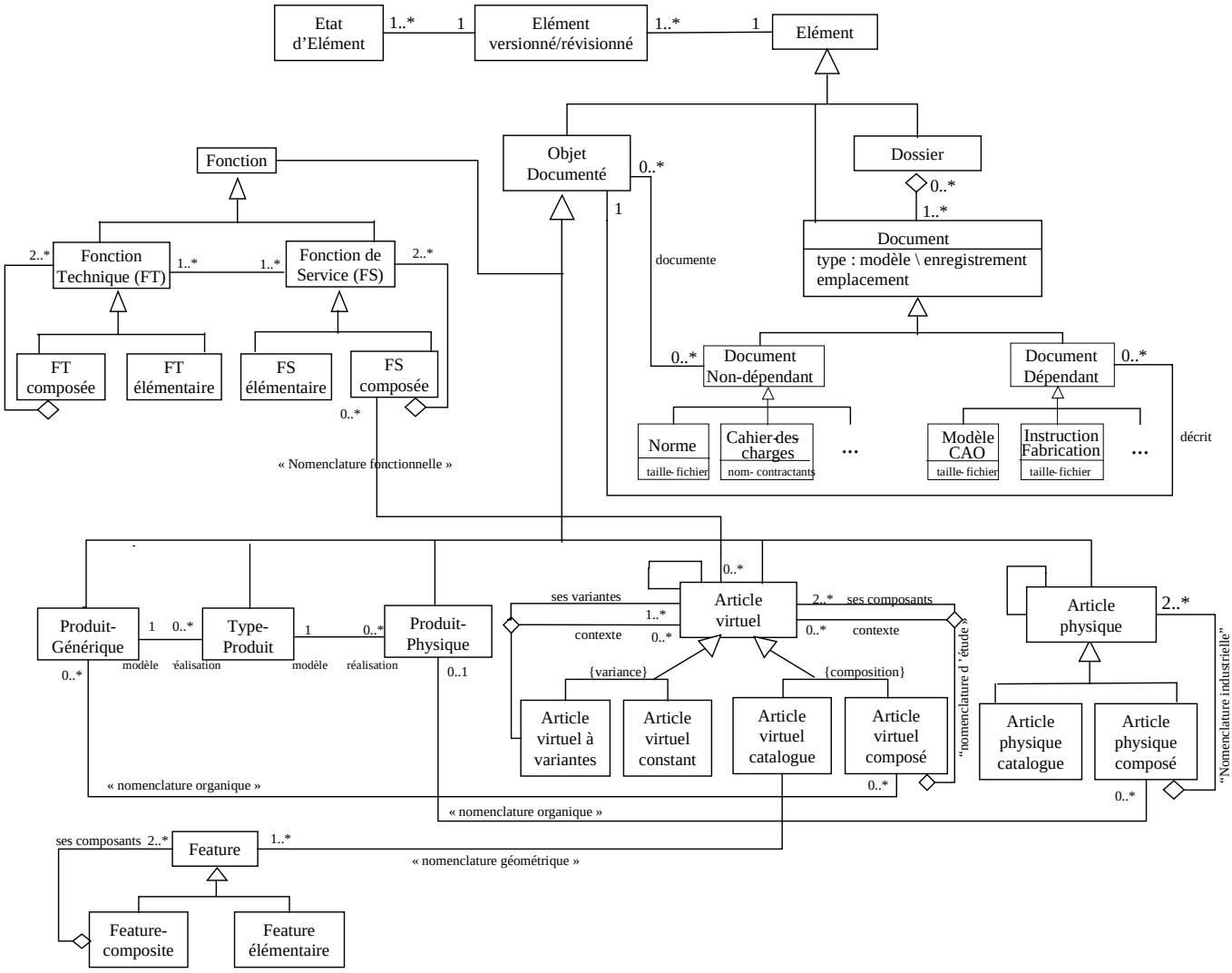


Figure 4.21 - Modèle produit du SIP

5. Conclusion

Le référentiel de domaine SIP obtenu à l'issue de cette analyse de domaine organise l'ensemble des connaissances dans un modèle de référence produit. Ce modèle a été construit en se basant sur l'hypothèse de trois niveaux de produit. Nous avons souligné l'existence de trois niveaux de produit dans l'entreprise mais certaines organisations peuvent ne pas avoir de produit générique par exemple. D'autres organisations peuvent ne pas gérer de produits physiques. Dans ce cas, il n'y a aucune raison pour distinguer des articles physiques des articles virtuels. Par ailleurs, un niveau de produit donné peut avoir une variété de cycle de vie, selon les processus de définition des entreprises. Le nombre et le type de nomenclatures gérées varie largement entre les entreprises. Par ailleurs, chaque nomenclature a diverses constructions. Le nombre et le type de documents et des dossiers gérés varie également. Ainsi la Figure 4.21 est perçue comme un modèle produit possible pour un SIP, basé sur trois niveaux de produits, quatre nomenclatures (fonctionnelle, géométrique, organique d'étude et organique industrielle) avec des documents typés dépendant et non-dépendant et des dossiers composés des quatre nomenclatures retenues. Nous montrons à l'issue de cette deuxième partie du manuscrit comment offrir une démarche pour construire un modèle produit du SIP, tenant compte de la spécificité de chaque organisation.

Chapitre V :

Référentiel Processus

1. Introduction

Nous rappelons l'objectif du référentiel du domaine SIP qui est d'explorer les besoins informationnels fréquemment exprimés par les utilisateurs du SIP durant la phase d'analyse des besoins ainsi que les problèmes de modélisation souvent rencontrés par les concepteurs SIP pour spécifier les besoins informationnels identifiés. Le chapitre précédent (Chapitre IV) s'est focalisé sur les connaissances produit. Le présent chapitre a pour objectif de passer en revue et de structurer l'ensemble des connaissances souvent exprimées par les utilisateurs et les concepteurs de SIP lors de la spécification des processus SIP. Comme pour les connaissances produit, chaque concept étudié est d'abord décrit puis structuré. Mais avant de présenter cela, nous présentons brièvement les processus SIP (Section 2).

2. Les processus SIP

Par processus SIP (cf. Chapitre I), nous désignons les processus "métiers" du système d'information produit et ce par opposition aux processus "logiciels" du système informatique associé (SGDT en l'occurrence) qui concourent à la réalisation des processus SIP. Par ailleurs, nous avons défini le système d'information produit comme étant "un dispositif organisationnel permettant de réguler la création, la circulation, l'utilisation et l'évolution du patrimoine informationnel de définition du produit". Les processus SIP permettent donc de réguler la **création**, la **circulation**, l'**utilisation** et l'**évolution** des informations produit.

Nous avons souligné dans le chapitre I, l'existence de plusieurs classifications des processus SIP [CIMdata,97] [Brude, 98] [Maurino, 93] [Randoing, 95] [Hameri, 98]. La majorité d'entre elles convergent sur un ensemble d'objectifs essentiels auxquels concourent les processus SIP. Nous avons alors distingué deux classes de processus SIP (cf. chapitre I-§2.2.2) :

1. **Le processus de définition** du produit. Ce processus vise à supporter les activités de définition (conception) de produit et à assurer en conséquence la création de l'information (dans le système qui l'a génère et sur son support approprié) et ensuite l'organisation des informations créées autour d'un modèle fédéré de produit pour faciliter l'accès aux informations. Il comprend entre autre la classification de composants, la définition des diverses configurations de produits, la mise en place du système documentaire associé, etc.
2. **Le processus d'évolution** du produit[113]. Ce processus vise à supporter l'évolution des produits au cours du temps et à assurer en conséquence la cohérence de l'information technique associée au produit (quels que soient les systèmes et les outils qui l'ont générée, et les acteurs qui la manipulent). C'est un processus déterminant dans le cycle de vie d'un produit. Dans [Guffond, 98], on souligne qu'on "aurait tort de ne le considérer que comme une adaptation marginale, une quelconque intendance qu'une conception plus efficace en amont pourrait à priori réduire". En effet, chaque produit évolue en fonction de l'élargissement et des modifications de la demande (adaptation de l'offre), de la recherche d'une plus grande fiabilité (actions correctives visant l'amélioration de la qualité), mais

[113] Connu également sous le nom de processus de modification de produit[113]. Il est encore plus connu sous le vocable anglo-saxon d'*Engineering Change Process*.

aussi en lien avec les évolutions du process de fabrication (recherche d'une meilleure productivité). Tout comme le processus de définition du produit, le processus d'évolution est vital et incontournable pour la vie du produit. Ce processus est par ailleurs riche puisque c'est autour de lui que le monde de la conception et le monde de la production se rejoignent. De ce fait, des préoccupations hétérogènes entrent en interaction.

Si l'on veut présenter une description détaillée de chacun de ces processus en termes d'étapes qui les constituent, il nous serait difficile d'en définir une car cela diffère d'une organisation à l'autre. La décomposition d'un processus peut en effet varier selon la taille de l'entreprise et la structure organisationnelle mise en place et peut évoluer dans le temps pour satisfaire des objectifs organisationnels différents. A ce titre, nous considérons le processus de modification de produits mécanique à Schneider Electric (au domaine d'activité stratégique BTP). Ce processus, se présentait initialement comme constitué des étapes suivantes[114] :

- émission de la demande de modification (par n'importe quel acteur dans l'entreprise),
- examen de la demande de modification (pour examiner la pertinence de la demande et décider de l'engagement d'une étude de la demande ou pas). A l'issue de cet examen, deux phases sont lancées en parallèle : l'enquête technique qui a pour but d'évaluer la faisabilité technique de la modification et l'enquête économique qui vise à évaluer les coûts engagés par la modification,
- examen des résultats des enquêtes et décision d'appliquer ou pas la modification.

Une telle organisation du processus de modification s'est révélée après quelques années d'application coûteuse en terme de temps et ce pour plusieurs raisons. Notamment à cause des diverses itérations nécessaires dans le processus, pour préciser ou aligner les différentes propositions provenant de métiers différents impliqués. L'aspect séquentiel des étapes du processus et l'existence de plusieurs goulots sont aussi à l'origine de ce dysfonctionnement. Une volonté pour dynamiser ce processus est née. L'idée est d'anticiper les tâches caractérisant le travail de modification et de partager au plus tôt les différents points de vue (technique, industriel et qualité) sur la modification. La nouvelle organisation de ce processus est désormais constituée de trois étapes[115] :

- création de la demande de modification et sa prise en compte par un comité de pilotage des modifications qui se charge de planifier l'ensemble des demandes de modification,
- étude de faisabilité conjointe effectué par un trinôme représentant divers métiers (technique, industriel, qualité) durant une durée fixée par le comité de pilotage,
- décision et réalisation planifiée de la modification.

Ainsi, à chaque nouveau projet de spécification de SIP, les utilisateurs du SIP sont amenés à décrire leurs processus métiers, selon la spécificité de leur entreprise. Par rapport à une démarche d'ingénierie de patrons, la représentation des processus constitue donc un bloc de concepts à capitaliser (une information souvent présente dans le SIP). Toutefois, les

[114] La description présentée est issue d'une enquête sociologique menée dans la DAS BTP par les partenaires sociologues du projet POSEIDON (Laboratoire CRISTO - Université Pierre Mendès France) pour analyser le fonctionnement du processus de modification de produit [Guffond, 98].

[115] Nous avons assisté à cette action de dynamisation en structurant le nouveau processus (ce qui a constitué une aide pour exprimer les besoins autour du nouveau processsus). Les partenaires sociologiques du projet POSEIDON ont participé par ailleurs au groupe de travail interne à Schneider [Guffond, 99]).

techniques de décomposition d'un processus varie d'une entreprise à une autre. Cela introduit ainsi une notion de variabilité à l'intérieur de ce bloc, que les patrons processus visent à fixer. Notre démarche de patrons devrait fournir une aide pour décomposer les processus, selon différents critères. L'objectif du présent chapitre est de passer en revue l'ensemble des concepts associés au processus et nécessaire pour le décrire et le décomposer.

3. Description Globale Des Processus

3.1. Définition du concept de "processus"

Bien que le concept de processus soit partagé par tous les spécialistes, sa définition, sa typologie et sa description varient d'un domaine à l'autre (informaticiens, automaticiens, gestionnaires, etc.). Les processus auxquels on s'intéresse correspondent, d'un point de vue système d'information, aux processus *organisationnels*[116] du système d'information organisationnel SIO (en l'occurrence la gestion des informations produit).

Le positionnement des processus SIP par rapport à l'ensemble des processus de l'entreprise est difficile à cause de la diversité et parfois la contradiction des typologies de processus proposées dans la littérature. Les processus SIP peuvent toutefois être rapprochés, selon la classification de [Harrington, 91], aux processus métiers (*business process*[117]) de l'entreprise, par opposition aux processus de production (*production process*[118]).

Pour décrire les processus SIP, nous nous inspirons ainsi de la littérature de modélisation et d'intégration d'entreprise, fort riche des développements qui ont été menés dans divers projets européens et internationaux (CIMOSA, GIM, PERA) mais aussi de plusieurs travaux normatifs menées par le CEN[119] (Comité Européen de Normalisation), l'ISO[120] (International Standard Organisation) et l'IFAC/IFIP[121]. Nous nous appuyons également sur la littérature de la Gestion de Données Techniques, bien que moins abondante, pour décrire l'aspect organisationnel rattaché aux SIP.

Plusieurs définitions du concept de processus ont été proposées. Nous étudions dans ce qui suit certaines d'entre elles.

En terme de travaux normatifs, l'ISO (ISO 8420, 1994) définit un processus comme "un ensemble de moyens (personnels, équipements, méthodes, etc.) et d'activités liées qui transforment des éléments entrants (inputs) en éléments sortants (outputs), tout en créant de la valeur ajoutée". M. Hammer [Hammer, 93] rejoint cette définition en présentant le processus

---

[116] Par opposition aux processus informatiques associés dans le SI informatisé.

[117] Business process : all service processes and processes that support production processes (e.g. order process, engineering change process, payroll process, manufacturing process design). A business process consists of a group of logically related tasks that use the resources of the organization to provide defined results in support of the organization's objectives… Examples: document review, product management, design systems support, engineering change management, product development, … [Harrington, 91].

[118] Production process : any process that comes into physical contact with the hardware or software that will be delivered to an external customer, up to the point the product is packaged (e.g. manufacturing computers, food preparation for mass customer consumption, oil refinement, changing iron ore into steel). It does not include the shipping and distribution processes [Harrington, 91].

[119] Développant deux pre-normes : CEN ENV 40 003 et CEN ENV 12 204.

[120] Le comité technique ISO/TC 184/SC5/WG1, ayant pour objectif d'établir un cadre de référence.

[121] Le groupe de travail l'IFAC/IFIP Task Force on Architectures for Enterprise Integration qui développe une architecture générique appelée GERAM.

comme "une suite d'activités qui, à partir d'une ou de plusieurs entrées, produit un résultat représentant une valeur pour le client". H.J. Harrington [Harrington, 91] précise qu'un processus désigne toute activité ou groupe d'activités qui prend une entrée, lui ajoute de la valeur et fournit une sortie à un client interne ou externe. Par ailleurs, un processus utilise des ressources de l'organisation pour fournir des résultats définitifs"[122]. P. Lorino [Lorino, 97] présente un processus comme un ensemble d'activités reliées entre elles par des flux d'information ou de matière, qui se combinent pour fournir un produit matériel ou immatériel important et bien défini.

Ces définitions se rejoignent ainsi sur la définition du processus comme un ensemble *d'activités* liées qui transforment, à l'aide de *ressources*, des *éléments entrants* en *éléments sortants* pour créer de la valeur ajoutée.
Nous retrouvons ces caractéristiques dans l'analyse comparative des définitions, proposée par S. Limam [Limam, 99] qui fait ressortir que la plupart des définitions se rejoignent sur un ensemble de caractéristiques communes pour un processus:

- un processus est composé d'activités/opérations liées par des relations ou enchaînés. Cet enchaînement existe à cause des flux matériels ou informationnels.
- le déclenchement d'un processus est assuré par des flux informationnels et/ou de matières. Il est défini, selon la discipline, par des termes tels que intrants (pour traduire des flux), événements (traduisant de l'information) ou composants (traduisant de la matière).
- la fin du processus est marquée par l'obtention ou la transformation d'éléments donc d'un résultat attendu du processus.

En considérant la définition de A. El Mhamedi [El Mhamedi, 97] qui présente un processus comme "une combinaison d'activités mobilisant des savoir-faire multiples se déroulant dans le temps et étant finalisé par un objectif", une notion supplémentaire d'*objectif* est alors introduite. Cette notion d'objectif se retrouve également dans la définition des processus opérationnels (business process) proposée par F. Vernadat [Vernadat, 99] où un processus est une succession de tâches qui contribuent à la réalisation des objectifs de l'entreprise. Il précise par ailleurs que de manière générale, un processus peut être défini comme un enchaînement d'activités à exécuter pour atteindre un but donné. Cet enchaînement forme ce qu'il est convenu d'appeler le flux de contrôle du processus, c'est-à-dire sa logique d'exécution.
Les activités d'un processus sont donc regroupées selon une logique de finalité. P. Lorino [Lorino, 97] souligne d'ailleurs que les activités peuvent être regroupées en un processus lorsqu'elles obéissent à la même logique de coût et de performance et qu'on peut leur trouver un résultat (output) global donc une finalité. Par ailleurs, il distingue les "objectifs" des "résultats" (output) :

- un résultat représente ce que le processus fournit concrètement. Même si le résultat est immatériel, donc informationnel, la notion de processus doit garder sa nature physique de flux d'information et aboutir à un résultat tangible,
- un objectif est un seuil de performance qu'on veut atteindre (notion plus abstraite et plus temporaire).

[122] Traduction de : any activity or group of activities that takes an input, adds value to it, and provides an output to an internal or external customer. Processes use an organization's resources to provide definitive reuslts.

Ainsi, aux caractéristiques précédemment dégagées pour les processus, s'ajoute la notion d'objectif d'un processus qui correspond à un résultat à atteindre. Dans la suite de cette section, nous détaillons les principales caractéristiques d'un processus que nous venons d'identifier, à savoir les notions de *composants* du processus (activité, opération, etc.), les *relations* entre les composants, les *entrants et sortants (ou résultats)* du processus et les *ressources* d'un processus.

3.2. Les composants du processus

3.2.1. Quels composants

Lorsqu'il s'agit de décrire les composants d'un processus, plusieurs termes sont utilisés tels que étapes, actions, tâches, activités et opérations. Certains de ces termes ont une signification générale pour désigner la décomposition de quelque chose telle que étape ou le fait de faire quelque chose tel que action. D'autres termes tels que activité, tâche ou opération ont une sémantique particulière dans la modélisation de processus. C'est cette sémantique que nous étudions afin de définir et lier l'ensemble de ces termes.

Toutes les définitions que nous avons présentées au §3.1 mais également plusieurs autres définitions issues des domaines variées tels que l'analyse des systèmes de production [Pourcel, 95] [Ouali, 97] [Megarsti, 99], la modélisation des processus de conception [Harani, 97] [Ouazzani, 97], l'ingénierie des processus en systèmes d'informations [Rolland, 96] s'appuient sur le concept d'activité comme élément de décomposition des processus. Ce concept est omniprésent dans les définitions proposées. Nous examinons ci-dessous quelques définitions du concept d'activité afin de le positionner par rapport aux autres concepts.

Dans le projet ACNOS [ElMhamedi, 97] "une activité est une combinaison d'opérations dont l'objectif est exprimé par le biais de la tâche. Chaque activité est caractérisée par une fonction qui transforme un état d'entrée, sous l'influence d'objets de contrôle et sous réserve de disponibilité de ressources nécessaires et de temps, en un état de sortie". P. Lorino [Lorino, 97] définit une activité comme un ensemble de tâches élémentaires et d'actions :

- réalisées par un individu ou une équipe,
- faisant appel à un ensemble homogène de savoir-faire,
- ayant un comportement cohérent du point de vue des coûts et des performances,
- pouvant être caractérisés globalement par des entrées et une sortie communs bien identifiés,
- et ayant effectivement ou potentiellement une importance significative pour la performance de l'entité analysée.

F. Vernadat [Vernadat, 99] présente les activités comme étant les étapes élémentaires d'un processus. Elles peuvent elles-mêmes être définies en terme d'actions plus élémentaires, appelées opérations fonctionnelles et exécutées par les entités fonctionnelles. Une activité est définie comme étant l'accomplissement d'une tâche. Elle consiste en général en une séquence d'opérations devant être exécutée en totalité par une ou plusieurs ressources et ceci dans un temps donné pour réaliser la tâche spécifiée.

A première vue, une activité a la même *structure* qu'un processus : un ensemble d'actions[123] qui transforme des entrants en des sortants à l'aide de ressources et qui a un objectif (appelé tâche). D'ailleurs, A. ElMhamedi [El Mhamedi, 97] souligne que les deux concepts d'activité et de processus dépendent du niveau d'analyse et de détail recherché dans la modélisation, c'est-à-dire qu'une activité peut être considérée à son tour comme un processus à un autre niveau d'analyse. Il considère que l'activité et le processus ont en quelque sorte une structure fractale comme le montre la Figure 5.1 ci-dessous :

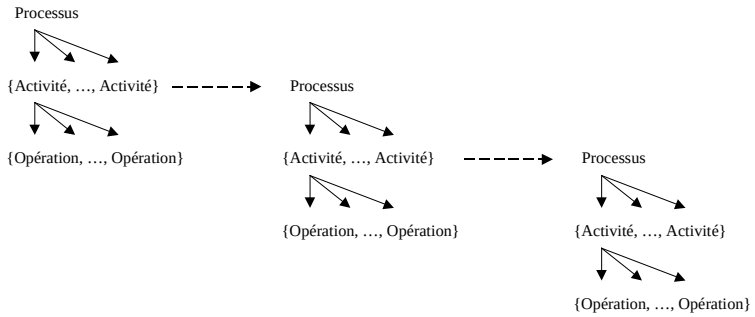


Figure 5.1 - Structure fractale des concepts d'activité et de processus (*Source : [El Mhamedi, 97]*)

C. [Foulard, 94] décrit un processus comme étant "formé de sous-processus et d'activités. C'est en fait une chaîne d'activités spécifiée au moyen de règles procédurales qui décrivent le comportement de l'entreprise en fonction des événements reçus et de l'état du système". Dans cette définition, l'auteur présente les activités comme éléments de décomposition d'un processus. Ils sont par ailleurs alignés aux sous-processus, ce qui laisse déduire qu'une activité est aussi un sous-processus (un processus de plus bas niveau).
Nous retrouvons cette idée également chez L. Hamaidi [Hamaidi, 97] qui parle de la décomposition fonctionnelle d'un processus de haut niveau en le décrivant sous forme d'un enchaînement de processus plus simples. La méthode IDEF3 définit un processus comme une enchaînement d'étapes appelées unités de comportement (UOB[124]) où une UOB peut être une activité IDEF0 ou une partie d'un processus ou un processus.

L'ensemble de ces propos fait apparaître donc une subtilité entre les concepts d'activité et de processus, du fait de la ressemblance de la structure de chacun de ces concepts. A. El Mhamedi [El Mhamedi, 97] souligne la différence entre ces deux concepts en distinguant la nature du flux impliqué dans chacun de ces concepts. L'activité concerne la transformation d'un flux d'objets (matières et/ou informations) sous l'influence d'objets de contrôle et sous réserve de disponibilité de ressources nécessaires et de temps. Elle décrit la fonctionnalité de l'entreprise. Le processus par contre concerne le flux de contrôle à exercer qui définit le comportement logique prévu de l'entreprise c'est-à-dire dans quel ordre seront exécutées les activités. Il décrit ainsi le comportement de l'entreprise. Les activités paraissent donc comme la décomposition du processus (transversal) selon les métiers ou les fonctions de l'entreprise concernés ou encore selon les différentes compétences requises pour atteindre l'objectif du processus. Elle peut correspondre donc à une décomposition selon différents acteurs ou selon différents sous-objectifs du processus (requérants chacun une compétence différente).

[123] Nous utilisons le terme action d'une façon général pour exprimer le fait de faire quelque chose.
[124] *Unit Of Behavior*.

De l'ensemble de ces réflexions, nous retenons donc que le processus est décomposable en des actions plus élémentaires dites "activités".
Certains auteurs parlent en plus du concept d'activité, du concept d'opération. Une activité peut en effet se décomposer en des actions plus élémentaires dites "opérations". Toutefois, si on explicite dans les définitions que l'activité se décompose en opérations, la limite entre ces deux notions n'est pas précise. On ne sait pas à partir de quel moment de la décomposition d'une activité on parle d'opérations. Ainsi, si on laisse entendre qu'une activité se décompose directement en opérations, cela impose deux niveaux de décomposition de processus uniquement : à un premier niveau en activités et à un deuxième niveau en opérations, ce qui ne paraît pas réaliste. Nous considérons donc les opérations comme les actions les plus élémentaires dans la décomposition d'un processus et qui ne peuvent donc être décomposées[125]. Une activité peut donc être décomposée en d'autres activités.

Si l'on veut se conformer au seul consensus établi autour du concept d'activité comme l'élément de décomposition du processus, nous considérons qu'un processus est composé d'activités. Une activité peut être par ailleurs soit décomposable et elle se décompose alors en d'autres activités, soit élémentaire. Ce dernier cas correspond à ce qui est souvent appelé "opération"[126]. Lorsqu'une activité est décomposable, nous la considérons comme un processus[127] (à un niveau d'analyse plus bas) qui se décompose alors en d'autres activités (qui peuvent être à leur tour des opérations ou des processus). Une activité, qu'elle soit de nature processus ou opération doit par ailleurs assurer un objectif, appelé parfois tâche.

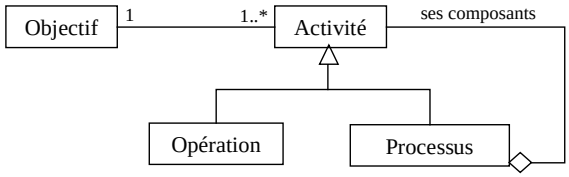


Figure 5.2 - Les composants d'un processus

En terme de typologie, différentes typologies d'activité sont proposées dans la littérature [Lorino, 97] [Vernadat, 99] : selon la nature des actions effectuées dans l'activité (activité de transformation[128] ou activité de décision[129]), selon la nature des entrants/sortants d'une

[125] Nous retrouvons cette définition dans CIMOSA où les opérations représentent le plus bas niveau de granularité dans la vue fonctionnelle des processus.

[126] Notons que dans [Hamaidi, 97], un processus est défini comme "un ensemble partiellement ordonné d'étapes formant une chaîne de traitements délimitée par un début (START) et une fin (FINISH). Une étape d'un processus est soit un sous-processus, soit une étape élémentaire appelée activité …L'activité est une étape élémentaire dans un processus. C'est l'unité d'ordonnancement en ce sens qu'elle requiert du temps et des ressources pour la durée complète de son exécution ... Une activité est un processus que l'on ne décompose pas". Cette définition est contradictoire à certaines définitions du concept "activité" avancées dans le texte; l'activité est donc vue comme une opération (telle que présentée dans certaines définitions).

[127] Dans l'autre sens, le processus peut être vu comme une activité (à haut niveau). Nous avons souligné auparavant que les notions de processus et d'activité sont caractérisés par des éléments semblables.

[128] Elle a pour objectif de transformer des objets d'entrée en objets de sortie. Elle a en général un caractère procédural et pour la plupart automatisable.

[129] Activité de nature cognitive (analyse et résolution de problèmes), souvent réalisée par des ressources humaines, dont on ne connait pas toujours le fonctionnement intime et donc difficilement automatisable.

activité, selon le type de déclenchement de l'activité (activité autonome[130] et activité commandée[131]), selon sa nature (activité de conception ou de réalisation ou de maintenance), son positionnement dans l'allocation de coût (activité primaire ou secondaire), etc.. Par rapport aux objectifs de notre étude et dans une perspective d'informatisation du SIP conçu, nous avons typé les activités de nature "opération" selon qu'elles soit informatisées à l'aide d'un SGDT ou non : une opération peut être donc de type informatisée ou manuelle.

En terme de **structuration**, nous représentons les concepts de processus, activité et opération à l'aide de diagrammes d'activité UML (cf. Figure 5.3). Toute activité qu'elle soit une opération ou un processus sera représenté par une activité UML, stéréotypé selon sa nature (opération ou processus). Un deuxième stéréotype peut être ajouté à l'activité lorsqu'elle est une opération pour indiquer si elle est de type manuelle ou informatisée.
La décomposition d'un processus donné sera représenté ainsi par un diagramme d'activités. Nous supposons qu'un processus a un état initial qui marque le début du processus et peut avoir un ou plusieurs états finaux qui correspondent chacun à une condition de fin du processus différente.

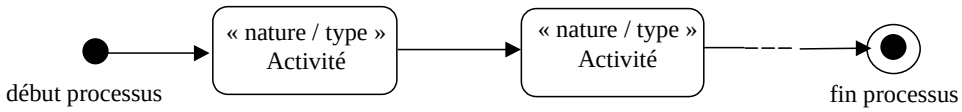


Figure 5.3 - Représentation de processus à l'aide de diagramme d'activité d'UML

3.2.2. Comment décomposer

Les composants d'un processus étant identifiés, il s'agit de déterminer les techniques de décomposition du processus en composants. Nous préconisons une décomposition du processus à plusieurs niveaux et ce selon le degré d'analyse souhaité (on peut vouloir juste voir les sous-processus d'un processus ou définir toutes les opérations du processus).
Cela revient donc à déterminer d'abord comment décomposer un processus en activités (c'est-à-dire selon quels critères) et ensuite comment déterminer la nature des composants obtenus (opération ou processus).
Concernant le premier point (critères de décomposition) et si les définitions des concepts de processus, activités, etc. sont abondantes dans la littérature, les propositions concernant la façon de décomposer les processus se font un peu rares. P. Lorino [Lorino, 97] définit cinq critères pour identifier une activité, qui sont les suivants :

1. les savoir-faire mobilisés doivent constituer un tout logique, définissant *une* compétence globale (individuelle ou collective),
2. l'ensemble des actions constituant l'activité est suffisamment homogène pour répondre à des lois de comportement économique uniques, significatives et cohérentes ; ils doivent être de même type - ayant des logiques de variation similaires (progressent ou regressent ensemble),
3. les tâches diverses composant l'activité peuvent être considérées comme ayant une sortie (output) globale, commune et unique (ayant toujours les mêmes caractéristiques),

[130] Déclenchées par les réalisateurs de l'activité eux-mêmes.
[131] Déclenchées par un ordre venant de l'extérieur de l'activité.

4. une activité doit être "intégrée" : les tâches réalisées pour transformer les inputs de l'activité en outputs doivent toutes se situer à l'intérieur des limites de l'activité (limites en termes de ressources utilisées),
5. une activité doit être reconnaissable et reconnue.

Le même auteur souligne toutefois que la décomposition du modèle d'activités ne découle pas automatiquement de la définition des activités car l'activité peut être très macroscopique ou très microscopique.

Dans son approche de décomposition des processus, [Arasti, 99] précise que la décomposition doit définir des activités équivalentes (quant à la taille et au champ d'action) et indépendantes ou autonomes (ce qui rejoint le quatrième critère préconisé par Lorino).

Nous avons souligné au paragraphe précédent que les activités paraissent, selon la définition proposée par A. El Mhamedi [El Mhamedi, 97], comme la décomposition du processus (transversal) selon les métiers ou les fonctions de l'entreprise concernés.
Nous proposons ainsi de décomposer les processus selon les différentes compétences requises pour atteindre l'objectif du processus. Par compétences requises, nous entendons dire soit des acteurs ayant des expertises différentes (des métiers différents), soit des supports différents (support SGDT ou non), soit différentes performances à réaliser pour atteindre l'objectif global du processus donc différents sous-objectifs à réaliser. Nous préconisons alors trois critères de décomposition d'un processus en activités :

- *Changement d'objectif* : quand l'objectif du processus est assez général ou assez complexe (riche), il est nécessaire de l'affiner en le décomposant. Les sous-objectifs correspondent chacun à une performance particulière à atteindre dans l'ensemble du processus. Ces sous-objectifs sont associés à diverses activités qui correspondent à la décomposition du processus initial. Nous soulignons que la présence d'un objectif dans une activité donnée signifie implicitement l'occurrence d'une action de *mesure* de satisfaction de l'objectif à la fin de l'activité et donc un point de décision[132]. La décomposition de processus selon les objectifs implique donc une décomposition selon les points de décision.
- *Changement de ressource* : un processus est assuré par des ressources ayant différents rôles[133] dans le processus. Une deuxième façon de décomposer un processus sera par changement de ressources. Par changement de ressource, nous entendons aussi bien le changement d'une ressource par une autre que le changement du rôle d'une même ressource. Le changement de ressource ou de rôle de ressource constitue en effet un bon indicateur quant au passage d'une activité à une autre.
- *Changement de type d'activité* : une opération peut être informatisée à l'aide d'un SGDT ou non. Si une activité du processus ne peut être caractérisée de façon complète (i.e. on ne peut lui affecter un unique type : informatisée ou manuelle), il convient alors de

[132] Un point de décision correspond au choix entre plusieurs transitions disjointes lors de la fin d'une activité donnée.

[133] Par rôles d'une ressource, nous entendons les autorisations ou droits d'une ressource dans l'activité à laquelle elle participe. Nous définissons 3 types de rôles pouvant être tenus par une ressource : initiateur (déclenche l'activité), exécutant (réalise l'activité), responsable (décideur de la suite du processus de niveau supérieur). Ce concept est détaillé dans le § 3.5.

décomposer cette activité en autant d'opérations que de types auxquels elle aurait pu être associée.

Reste maintenant à déterminer, une fois les activités composantes identifiées, la nature des activités obtenues. Nous avons convenu de typer une activité comme processus lorsqu'elle est décomposable et comme opération lorsqu'elle le n'est pas. En s'appuyant sur les critères de décomposition ci-dessus, nous déduisons que :

- une activité est une opération lorsque son objectif est élémentaire, lorsqu'elle est assurée par un seul acteur, jouant un rôle unique et lorsqu'elle a un unique type (informatisée ou manuelle).
- une activité est un processus lorsque son objectif est trop général ou complexe ou il lorsqu'il n'est pas possible de lui affecter un acteur *exécutant* ou lorsqu'il n'est pas possible de lui affecter un unique type (informatisée ou manuelle).

3.3. Les relations entre composants du processus

Dans les définitions de processus étudiées dans §3.1 et §3.2, le relations entre les activités d'un processus ont été souvent exprimées dans des termes du type : enchaînement (partiellement ordonné) d'activités, chaîne d'activités, séquence d'opérations, ensemble (partiellement ordonné) d'activités, ensemble d'activité (liées, inter-liées, qui se combinent), suite d'activités, groupe d'activités, etc.
Le type de relations entre les activités (enchaînement, succession, liaison, dépendance) n'est pas encore tranché. G.N. Mentzas [Mentzas, 93] distingue toutefois trois types de relations entre les activités :

- relation de succession (les sorties d'une activité sont nécessaires pour qu'une autre activité se réalise),
- partage de ressources (deux ou plusieurs activités utilisent les mêmes ressources et l'exploitation des ressources par l'une peut influencer la performance de l'autre),
- simultanéité (les sorties de deux ou plusieurs activités sont nécessaires pour réaliser les activités suivantes dans le même processus).

Nous nous intéressons en particulier à étudier les relations de successions c'est-à-dire à l'enchaînement des activités. Cet enchaînement d'activités, comme le précise S. Limam [Limam, 99], existe à cause des flux matériels ou informationnels (désignés généralement par les termes d'inputs-outputs ou entrées-sorties) et non pas à cause de liens cause-conséquence comme c'est le cas dans les procédures. C. Foulard [Foulard, 94] indique d'ailleurs que l'état de fin d'une activité (indiquant comment s'est terminée l'activité) est utilisé par les règles procédurales des processus pour enchaîner les activités au cours du temps.
Si certains auteurs considèrent que l'enchaînement des activités est prédéfini (processus déterministe), d'autres proposent une typologie de processus selon le caractère prédéterminé ou non de l'enchaînement de ses activités. Ainsi, dans [El Mhamedi, 95], on distingue :

- Les processus *structurés* : caractérisés par une connaissance quasi-parfaite à la fois de l'ensemble des activités qui la composent et de leur enchaînement. L'objectif du processus est parfaitement défini. c'est des processus automatisables.

- Les processus *semi-structurés* et *non structurés* : caractérisés par une connaissance imparfaite de l'enchaînement des activités. Deux cas se présente :
  - Lorsque l'objectif du processus est connu à priori et est parfaitement défini, on parle de processus semi-structuré. Dans ce cas, le cheminement menant à l'objectif est déterminé à fur et à mesure du déroulement du processus. C'est un processus où l'homme prend des décisions sur le choix des activités (parmi un portefeuille d'activités) et de leur enchaînement et ce selon le résultat, imprévisible, de la phase précédente.
  - Lorsque l'objectif n'est pas connu à priori, on parle de processus non structuré. Dans ce cas, l'ensemble des activités qui le composent n'est pas également connu. L'objectif se construit alors progressivement au cours du déroulement du processus et donc nécessite la création ou le développement d'activités à fur et à mesure aussi. L'homme prend des décisions sur la détermination de l'objectif et le choix du cheminement pour atteindre l'objectif (les activités qui permettent de l'atteindre) ainsi que les capacités à mobiliser.

Quant au type d'enchaînement, la méthode IDEF3 définit l'enchaînement des activités (UOB) au moyen de connecteurs de type : Et / Ou / Ou-exclusif et suivant deux modes : synchrone / asynchrone. Il permet en plus de faire des relations entre UOB de processus différents pour exprimer la précédence, l'interaction, les flux de données, dans le cas de processus synchronisés ou concurrents. Dans CIMOSA [Vernadat, 96], l'enchaînement des activités est défini au moyen des opérateurs suivants : début et fin de processus, mise en séquence, mise en parallèle, branchement conditionnel synchrone ou asynchrone, rendez-vous ou synchronisation, rebouclage.

Nous retenons deux types de succession (étiquetés par des conditions de succession) entre les activités :

- La succession du type ET : correspondant à une activité qui a un ou plusieurs suivants conditionnés par des conditions non exclusives. Le cas d'un seul suivant correspond à une séquence d'activités. Le cas de plusieurs suivants correspond à une activité dont les suivants peuvent se faire *en parallèle* quand les conditions de succession sont vérifiées.
- La succession de type OU : correspondant à une activité ayant plusieurs suivants, conditionnés par des conditions *exclusives* (disjointes). Ce cas se présente lors d'une prise de décision où, suite à la fin d'une activité donnée, il y a un choix à faire entre plusieurs activités alternatives pour la poursuite du processus et ce selon le résultat atteint lors de l'activité antérieure.

Dans le cas d'une succession de type OU, nous distinguons trois natures de transitions:

- la *poursuite* du processus 'nominal' : quand l'objectif de l'activité antérieure a été entièrement satisfait,
- la *re-direction* c'est à dire l'appel à des activités initialement imprévues dans le processus nominal ou la refonte d'une partie des activités antérieures : quand l'objectif n'a pas pu être atteint mais que les résultats peuvent être corrigés par un changement des données d'entrée ou par dérogation,
- l'*abandon* du processus de niveau supérieur : quand l'objectif n'a pas pu être atteint et que les résultats ne peuvent être corrigés.

En terme de **structuration**, la succession entre activités est modélisée sur les diagrammes d'activités d'UML par des transitions UML, stéréotypées en fonction de leur nature (poursuite, re-direction, abandon). En ce qui concerne le déclenchement des transitions entre les activités, dans les diagrammes d'activités d'UML, les transitions sont automatiques. Elles ne sont pas déclenchées par des événements externes mais c'est la fin de l'activité précédente qui déclenche la transition et fait démarrer l'activité suivante. En d'autres termes, l'événement est la fin de l'activité précédente. Lorsqu'il s'agit de transitions gardées par des conditions de succession, c'est la condition de garde qui valide la transition et la déclenche. Le nom d'une transition correspond à la condition de transition (par exemple accord, refus, …).
Dans le cas d'une succession de type ET avec plusieurs successeurs, la synchronisation entre flots de contrôle est représentée à l'aide de barres de synchronisation. Une barre permet d'ouvrir et de fermer des branches parallèles. Les transitions au départ d'une barre de synchronisation sont déclenchées simultanément. Inversement, une barre de synchronisation ne peut être franchie que lorsque toutes les transitions en entrée sur la barre ont été déclenchées. Dans le cas d'une succession de type OU, UML définit un stéréotype pour la visualisation de transitions avec des conditions exclusives. Une condition est matérialisée par un losange d'où sortent plusieurs transitions exclusives.

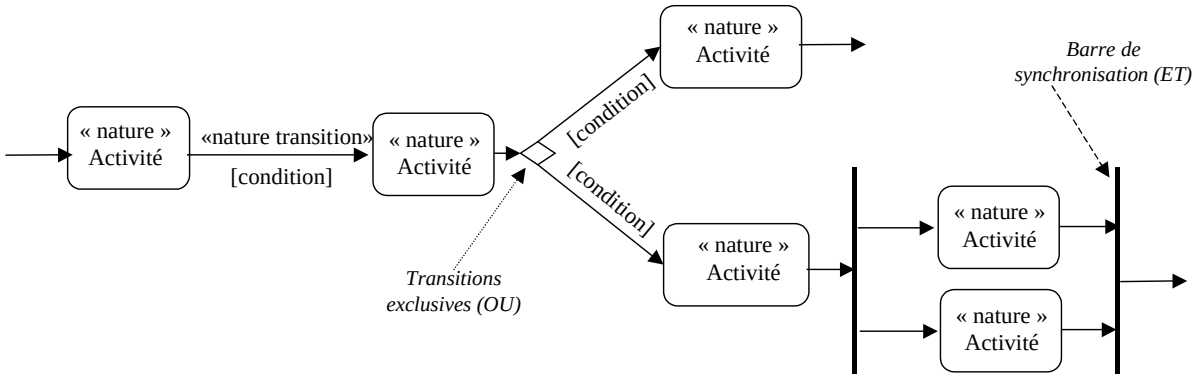


Figure 5.4 - Transitions entre activités dans UML

L'exemple suivant illustre cette modélisation sur une partie d'un processus SIP (étude de faisabilité des modifications produit) :

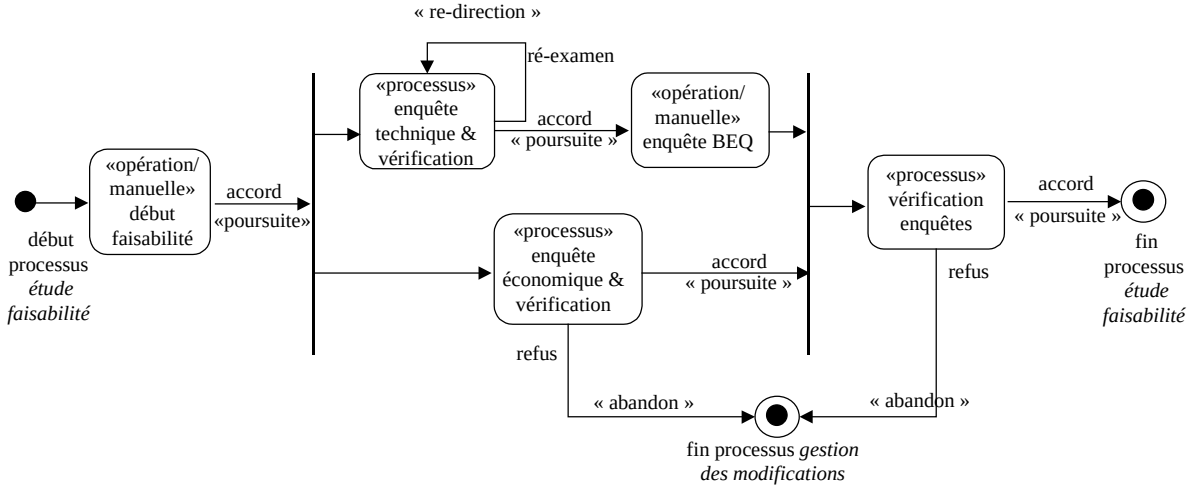


Figure 5.5 - Exemple de modélisation de transitions entre activités

3.4. Les entrants/sortants d'un processus

Dans [Foulard, 94], différents types d'entrants et de sortants sont définis. Une activité est présentée comme "réalisée au moyen de ressources (entrée ressource) au cours du temps et consiste à transformer des intrants (entrée fonctionnelle) en sortants (sortie fonctionnelle) sous certaines contraintes (entrée de contrôle) et produisant des informations de sortie (sortie de contrôle et sortie ressource)".

Nous retrouvons cette typologie dans la méthode de modélisation IEM (*Integrated Enterprise Model*) [Spur, 94] où les entrées et les sorties d'une activité sont définies sous forme d'objets qui relèvent de trois types : produit (entrées/sorties fonctionnelles) ; ordre (entrées/sorties de contrôle) et ressource (entrées/sorties ressources). Le projet ACNOS [El Mhamedi, 95] décrit également le processus en terme de flux de matière, d'information et de contrôle entre les activités.

Les entrées et les sorties de contrôle correspondent aux transitions entre activités, précédemment décrites (§3.3). Elles définissent le flot de contrôle du processus et déterminent l'enchaînement des activités dans le processus. Elles déclenchent les différentes activités du processus.
Les entrées et les sorties de ressources correspondent aux ressources mobilisées pour assurer les activités. Il peut s'agir d'acteurs humains ou de ressources matérielles qui réalisent les activités ou qui servent de support pour réaliser l'activité. Elles ne sont en aucun cas transformées par les activités. Nous décrirons ces entrées/sorties ressources d'une façon plus détaillée dans le §3.5.
Les entrées et les sorties fonctionnelles correspondent aux objets du domaine sur lesquels agit l'activité pour réaliser son objectif. Elles sont alors transformées par ces activités. Il s'agit essentiellement de représentations de produit (documents, nomenclatures) qui changent d'états à la sortie d'une activité[134].

En terme de **structuration**, il est possible de faire apparaître dans les diagrammes d'activités d'UML, les objets avec leurs états respectifs (les états sont spécifiés dans une expression entre crochets). Les flots d'objets sont alors représentés par des flèches pointillées. Une flèche relie ainsi un objet (en fait un état d'objet) à l'activité qui l'a créé ou qui le met en jeu. Lorsqu'un objet produit par une activité est utilisé immédiatement par une autre activité, le flot d'objets représente également le flot de contrôle ; il est alors inutile de représenter explicitement ce flot de contrôle (transition entre activités).

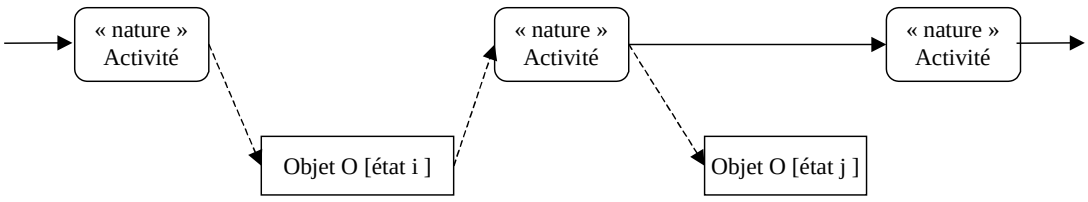


Figure 5.6 - Objets intrants et sortants dans le processus

[134] D'ailleurs dans CIMOSA [Vernadat, 96], l'exécution des activités se traduit par la transformation d'un état d'entrée en un état de sortie où les états sont définis en terme de vues d'objets de l'entreprise. Une vue d'objet est définie comme une projection d'un ou plusieurs objets d'entreprise sur une restriction de leur ensemble de propriétés à un instant donné.

L'exemple ci-dessous illustre cette modélisation sur un processus SIP (gestion des modifications de produit) :

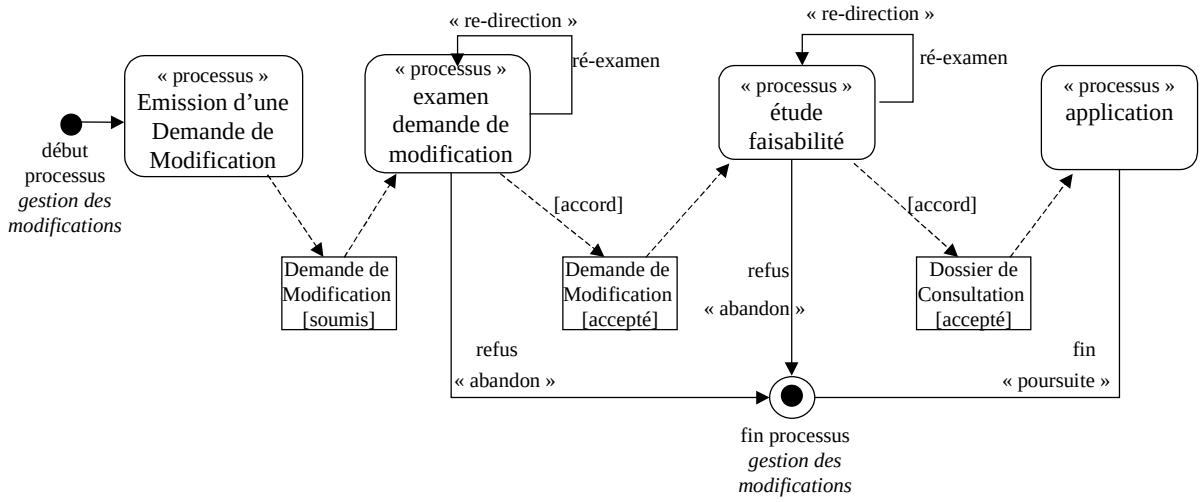


Figure 5.7 - Exemple de modélisation d'intrants\sortants de processus

Soulignons enfin que les sorties fonctionnelles d'une activité (donc les états de sortie des objets mis en jeu par l'activité) servent de base à la mesure de satisfaction de l'objectif de l'activité, c'est-à-dire l'adéquation du résultat obtenu (les sorties fonctionnelles) par rapport à l'objectif fixé de l'activité. Cette mesure sert pour la décision sur la suite du processus lorsqu'il s'agit de plusieurs transitions disjointes (succession de type OU) et influencent donc le flux de contrôle du processus dans lequel est impliquée l'activité. Les conditions de gardes placées sur les transitions vers les activités suivantes sont donc généralement fonction du résultat de la mesure de satisfaction de l'objectif de l'activité précédente.

3.5. Notion de ressources

Les ressources auxquelles on s'intéresse sont celles qui correspondent aux mécanismes d'une activité au sens de la modélisation IDEF0, c'est-à-dire les objets de l'entreprise qui contribuent à la réalisation des fonctionnalités des activités du processus. Dans CIMOSA, ces ressources sont qualifiées de ressources actives[135] ou entités fonctionnelles. Dans [Vernadat, 96], une ressource est définie comme une entité (humaine ou technique) qui peut jouer un rôle dans la réalisation d'un certain type de tâches, lorsqu'elle est disponible. Une taxonomie de ressources est également présentée. Elle distingue trois classes de ressources : les ressources humaines, les outils ou machines (tels que les outils informatiques et divers autres outils technologiques tels que les machines à commande numérique) et les applications (systèmes informatiques et logiciels tels que les systèmes de CAO, de MRP, les systèmes de GED, etc.).
Notons qu'en terme de ressources humaines, on parle plutôt d'acteurs que des personnes physiques. Un acteur correspond à un profil particulier de personnes physiques dans l'entreprise. Un profil est défini par l'association d'une personne physique, à un groupe de personnes appartenant au même métier de l'entreprise et à un niveau hiérarchique dans ce groupe. Ainsi plusieurs personnes physiques peuvent correspondre au même acteur, du

[135] Par opposition aux ressources passives ou composants tels que les chariots, les palettes, les outils, etc. c'est-à-dire celles qui sont "utilisées" par les ressources actives pour réaliser l'activité.

moment qu'ils appartiennent au même groupe et y sont au même niveau. A titre d'exemple, un chef de bureau d'étude, un méthodiste, un responsable de ligne sont des acteurs différents. Cette façon d'organiser les ressources humaines garantit une stabilité de la modélisation des processus concernés en évitant la re-définition des ressources d'un processus à chaque changement de fonction des personnes physiques de l'entreprise.

Dans la définition de ressource ci-dessus présentée, une notion de rôle est rattachée au fonctionnement des ressources. Cette notion est essentielle car une même ressource peut intervenir dans le processus selon différents rôles en fonction de l'activité considérée. Toutefois, cette notion de rôle est peu explicite dans la littérature existante. Nous nous attachons dans la suite de ce paragraphe de spécifier ce concept.
Dans le dictionnaire Robert [Robert, 93], le rôle est une action, une influence que l'on exerce ; une fonction qu'on remplit (on parle par exemple du rôle du verbe dans la phrase).
Jouer un rôle dans la réalisation d'une activité revient donc à assurer une certaine fonction (un certain ensemble d'actions conventionnelles). Par ailleurs, un rôle limite les droits que peut avoir une ressource (en terme d'actions qu'elle peut exécuter).
Pour les processus SIP, nous définissons trois types de rôles qui peuvent être tenus par une ressource dans une activité :

- Initiateur : celui qui déclenche l'activité sans la réaliser.
- Exécutant : celui qui réalise effectivement l'activité, c'est-à-dire assure la transformation des objets entrants de l'activité en objets sortants.
- Responsable : celui qui décide à la fin de l'activité de la suite du processus de niveau supérieur.

Il résulte de cette classification que :

- Le rôle d'*initiateur* apparaît dans le cas d'une activité-processus qui comporte plusieurs autres activités. Elle ne peut être affectée à une opération puisque celle-ci devrait être assurée par un seul acteur, selon un seul rôle et que ce rôle ne peut être qu'un exécutant.
- Le rôle d'*exécutant* ne peut être joué que dans le cas d'une activité élémentaire c'est-à-dire d'une opération car une activité-processus est naturellement constituée de plusieurs activités et l'on ne peut parler d'exécution de l'activité-processus puisque celle-ci est assurée par l'exécution des activités qui la composent.
- Le rôle de *responsable* apparaît d'une part pour les activités à l'issue desquelles une prise de décision est nécessaire (c'est le responsable qui assure cette prise de décision) et d'autre part pour les activités-processus pour coordonner l'ensemble des activités les composants.

Ainsi, une opération peut se voir affecter un exécutant. Une activité-processus peut se voir affecter un responsable et/ou un initiateur.

Notons toutefois que l'ensemble de ces trois rôles sont valables pour les ressources humaines. Pour les ressources matérielles du type machine ou application, seuls les rôles d'exécutant et d'initiateur sont valides[136].
Dans la suite, nous ne faisons pas la distinction entre les ressources du type applications[137] et machines puisqu'elles ne présentent aucune différence au niveau de la gestion (ils assurent les mêmes rôles) ; on les désigne sous le même vocable de ressource matérielle.

[136] Ils peuvent avoir le rôle de "responsable" dans le cas de processus entièrement automatisés. Ce cas ne se présente pas pour les processus organisationnels étudiés ici.

[137] Il est à noter que par application, on désigne des systèmes informatiques en dehors du SGDT.

Le rôle ainsi défini se présente à l'interface entre une ressource et l'activité. Il définit le type de fonction qu'assure une ressource dans une activité donné.
Toutefois, une ressource lorsqu'elle intervient dans une activité (selon un rôle donné) agit sur les entrants de l'activité c'est-à-dire les objets mis en jeu dans cette activité. Cette action peut être de quatre natures : créer, consulter, modifier ou supprimer un objet. De la même façon que pour une activité, la ressource ne peut pas faire n'importe quel type d'action sur l'objet. En effet, elle agit sur l'objet selon les autorisations dont elle dispose. Une autorisation est un droit à effectuer ou non une action sur un objet donné. Selon l'état de l'objet considéré, une même ressource peut avoir différentes autorisations. A titre d'exemple, lorsqu'un acteur donné crée une demande de modification d'un produit, cette demande est à l'état initial tant qu'elle n'est pas soumise par son demandeur. Dans cet état, le demandeur peut avoir le droit de la modifier ou de la supprimer. Un fois, cette demande prise en compte par les responsables de modification, elle ne peut qu'être consultée par ce demandeur.
Ainsi, à l'interface entre une ressource et un objet (plutôt un état d'objet), une notion d'autorisation existe pour définir le type d'action que peut faire une ressource sur un objet dans un état donné. Nous considérons que ces autorisations n'ont lieu que dans le cas où la ressource joue le rôle d'exécutant dans l'activité (car c'est dans ce seul rôle qu'une ressource exécute et donc agit sur des objets du domaine).

En terme de **structuration**, le concept de ressource sera représenté dans le diagramme d'activités d'UML par un acteur UML[138], stéréotypé selon son rôle .

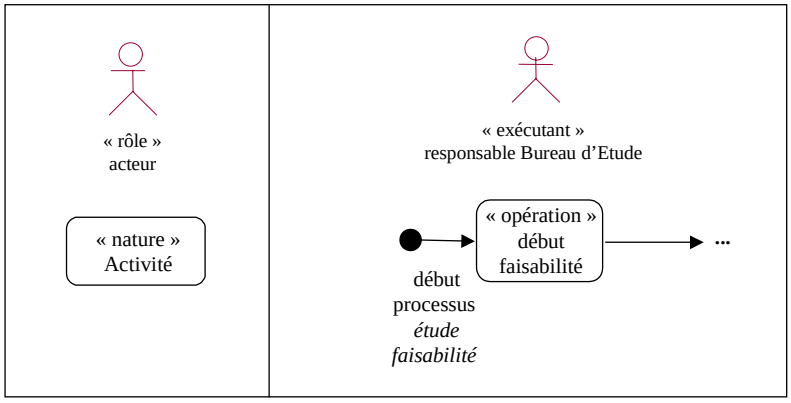


Figure 5.8 - Modélisation des ressources de processus

4. Synthèse

De l'ensemble de la description faite au §3, l'ensemble des concepts identifiés pour décrire les processus sont les suivants :

1. *Activité* : par activité, nous entendons tout type d'action effectuée dans l'entreprise par une ou plusieurs *ressources* ayant différents *rôles* et ce pour satisfaire un *objectif* donné. Une activité peut être une *opération* ou un *processus*.
2. *Ressource :* tout ou partie des moyens humains et matériels (capacités) nécessaires pour exécuter une activité. Il peut s'agir d'un acteur ou d'un groupe d'acteurs (ressource humaine) ou de ressources matérielles (machines ou applications[139]).

[138] Un acteur UML peut être aussi bien un acteur humain qu'un acteur matériel.

3. *Eléments d'entrées / de sorties* : ce sont les objets du domaine sur lesquels agit l'activité pour réaliser son objectif. Il sont transformés par ces activités. Il s'agit essentiellement de représentations de produit (documents, nomenclatures) qui changent alors d'états à la sortie d'une activité.
4. *Rôle :* fonctions d'un ressource dans l'activité à laquelle elle participe. Nous définissons 3 types de rôles pouvant être tenus par une ressource : initiateur (déclenche l'activité), exécutant (réalise l'activité) et responsable (décideur de la suite du processus de niveau supérieur).
5. *Autorisation* **:** droit d'une ressource, intervenant dans une activité, à effectuer ou non une action[140] (créer, modifier, supprimer ou consulter) sur un élément d'entrée de l'activité.
6. *Elément :* un objet du domaine sur lequel agit l'activité pour réaliser son objectif. Il est alors transformé par cette activité. Il s'agit essentiellement de représentations de produit (documents, nomenclatures). Un élément d'entrée d'une activité voit son état transformé à la fin de l'activité et devient alors un élément de sortie de l'activité. Les éléments d'entrée et de sortie peuvent être exprimés à tout niveau du processus (même s'ils sont transformés à un niveau plus bas dans la décomposition de l'activité).
7. *Objectif :* résultat que doit fournir le système sur lequel l'activité agit.
8. *Opération* : une opération est définie comme une activité élémentaire qui peut être effectuée isolément dans l'entreprise ou regroupée avec d'autres activités pour modifier l'état des objets. L'opération est non décomposable au sens où elle correspond à un objectif élémentaire, elle est assurée par un seul acteur, jouant un rôle unique et elle a un unique type : informatisée ou manuelle.
9. *Processus* : un processus est défini comme une suite partiellement ordonnée d'activités de l'entreprise réalisant un objectif et mobilisant un ou plusieurs acteurs. Un processus est une activité décomposable. Il est composé d'autres activités liées entre elles par des liens de succession étiquetés par des conditions de succession. Nous distinguons deux types de succession entre activités :

- La succession du type ET correspond à une activité qui a un ou plusieurs suivants conditionnés par des conditions non exclusives. Le cas d'un seul suivant correspond à une séquence d'activités. Le cas de plusieurs suivants correspond à une activité dont les suivants peuvent se faire *en parallèle* quand les conditions de succession sont vérifiées.
- La succession de type OU correspond à une activité ayant plusieurs suivants, conditionnés par des conditions exclusives (disjointes). Ce cas se présente lors d'une prise de décision où, suite à la fin d'une activité donnée, il y a un choix à faire entre plusieurs activités alternatives pour la poursuite du processus et ce selon le résultat atteint lors de l'activité antérieure. Dans le cas d'une succession de type OU, nous distinguons trois natures de transitions :
  - la *poursuite* du processus 'nominal' : quand l'objectif de l'activité antérieure a été entièrement satisfait,

---

[139] Hors SGDT.

[140] La ressource devra donc avoir le rôle d'exécutant dans l'activité puisqu'elle effectue des actions sur les éléments concernés par l'activité. La modélisation de la notion d'autorisation à l'interface entre activité et élément d'entrée n'introduit aucune ambiguïté quant à la ressource concernée par cette autorisation puisque seule la ressource exécutante de l'activité concernée a accès à ces autorisations (lorsqu'une activité est du type processus, celle-ci est exécutée par des ressources par le biais de ses opérations - la ressource exécutante d'une activité est alors une ressource exécutante d'une de ses opérations).

- la *re-direction* c'est à dire l'appel à des activités initialement imprévues dans le processus nominal ou la refonte d'une partie des activités antérieures : quand l'objectif n'a pas pu être atteint mais que les résultats peuvent être corrigés par un changement des données d'entrée ou par dérogation,
- *l'abandon* du processus de niveau supérieur : quand l'objectif n'a pas pu être atteint et que les résultats ne peuvent être corrigés.

Le métamodèle suivant (Figure 5.9) modélise l'ensemble concepts et des relations inter-concepts utilisés pour modéliser les processus :

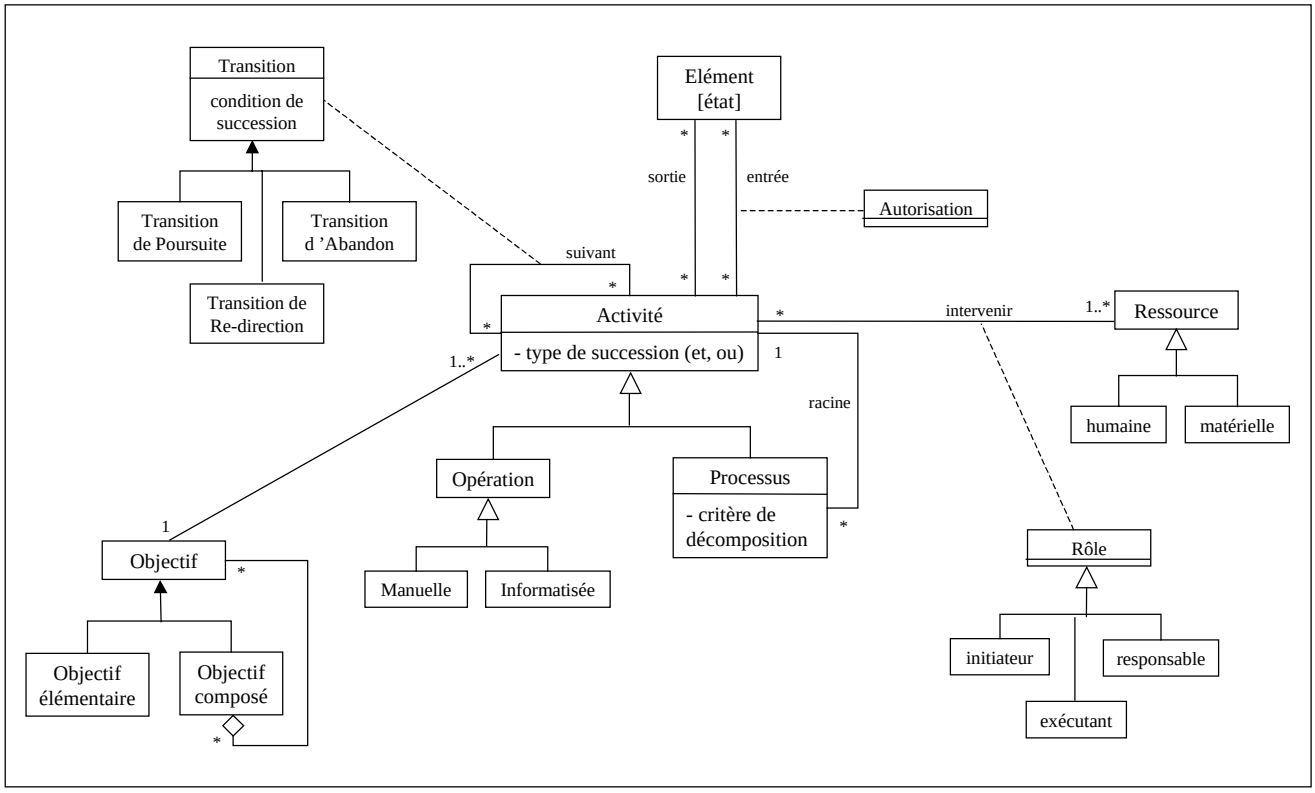


Figure 5.9 - Métamodèle du processus

Nous représentons les processus à l'aide de diagrammes d'activités UML. Toute activité qu'elle soit une opération ou un processus sera représentée par une activité UML, stéréotypée selon sa nature. La décomposition d'un processus est représentée ainsi par un diagramme d'activités. Nous supposons qu'un processus a un état initial qui marque le début du processus et peut avoir un ou plusieurs états finaux qui correspondent chacun à une condition de fin du processus différente.
Une ressource (de quelque nature qu'elle soit) sera représentée par un acteur UML, stéréotypé selon son rôle. Les transitions sont modélisées par des transitions UML stéréotypées en fonction de leur nature (poursuite, re-direction, abandon). Le nom correspond à la condition de transition (par exemple accord, refus, …).
Dans le cas de transition avec des conditions exclusives (succession de type OU), une condition est matérialisée par un losange d'où sortent plusieurs transitions exclusives.
Dans le cas de transition avec plusieurs successeurs en parallèle (succession de type ET), la synchronisation entre flots de contrôle est représentée à l'aide de barres de synchronisation. Une barre permet d'ouvrir et de fermer des branches parallèles.

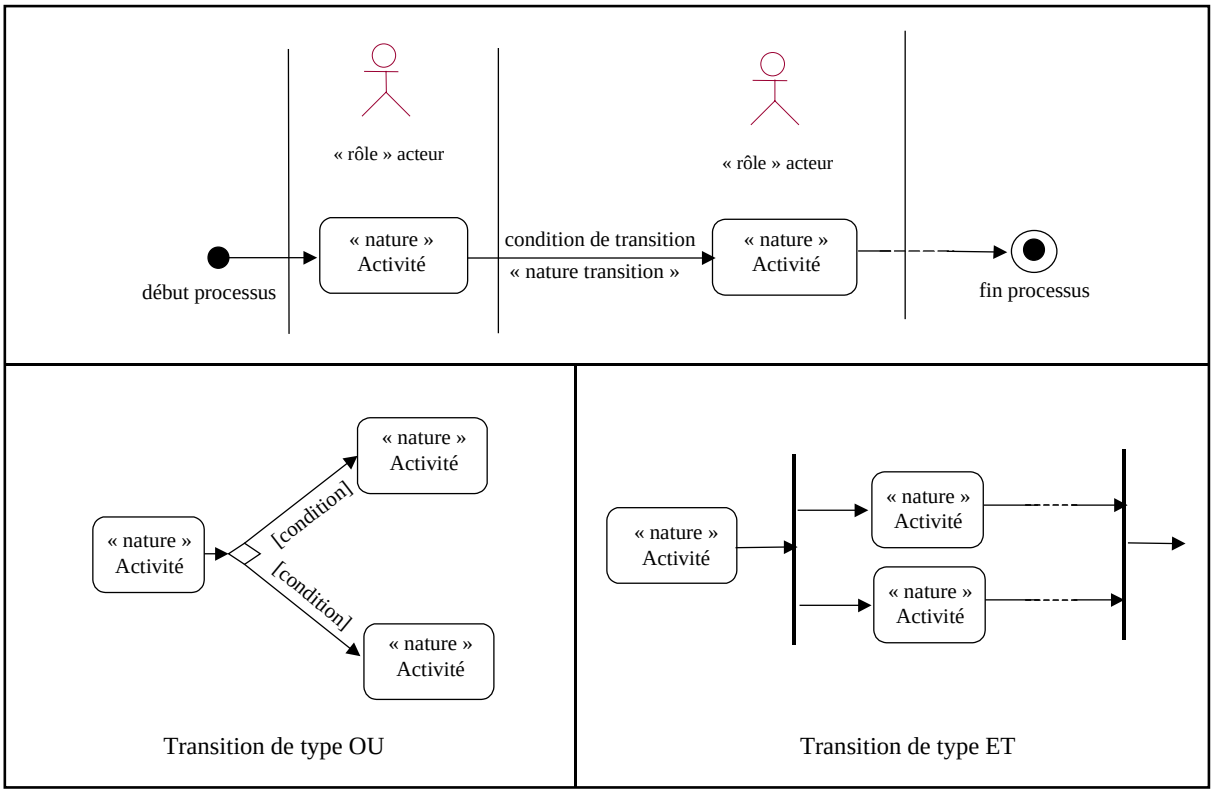


Figure 5.10 - Représentation du processus à l'aide des diagrammes d'activités d'UML

En terme de décomposition d'un processus en activités, et en s'appuyant sur les critères définis dans le §3.2.2, nous définissons dans un patron la démarche pour décomposer un processus. Ce patron est présenté au chapitre VI.

5. Conclusion

Le référentiel de domaine SIP obtenu à l'issue de cette analyse de domaine organise l'ensemble des connaissances dans un modèle de référence processus. Il ressort de cette analyse de domaine que la description de chacun des processus SIP ne peut être unique. Le processus de modification à titre d'exemple peut être composé de quatre étapes de demande, prise en compte, étude et application comme il peut être composé des étapes de demande, d'étude technique en parallèle avec une étude économique, de planification et d'application. Il en est de même pour les objets du domaine mis en jeu (documents et représentations manipulées) ainsi que l'organisation mise en place (ressources matérielles, acteurs, droits et rôles d'acteurs). La décomposition et l'organisation des processus n'est donc pas unique. Nous pouvons juste donner les concepts de base utilisés pour décrire un processus et une démarche générale de décomposition d'un processus. Aussi, nous avons préconisé deux critères de décomposition de processus (selon le changement d'objectif ou le changement d'acteurs). Certaines entreprises peuvent ne pas baser la décomposition de leurs processus sur ces deux critères mais sur d'autres (changement de la nature des objets du domaine manipulés, changement de site pour une entreprise multi-site, etc.). Comme nous l'avons déjà souligné en conclusion du chapitre IV, nous montrons à l'issue de cette deuxième partie du manuscrit comment offrir une démarche pour construire un modèle processus du SIP, tenant compte de la spécificité de chaque organisation.

Troisième Partie :

Les Patrons du domaine SIP

Le référentiel de domaine SIP obtenu à l'issue de l'analyse de domaine (2ème partie du manuscrit) organise l'ensemble des connaissances dans deux modèles : le modèle de référence produit et le modèle de référence processus. Nous avons souligné à l'issue de chacun des chapitres IV et V que les modèles de référence ainsi obtenus sont à la fois très généraux et très spécifiques. Ils sont très généraux car ils n'expriment pas des propriétés spécifiques aux organisations (tel que pour le produit une propriété du type "résistance" lorsqu'il s'agit d'industrie électronique ou pour le processus, des documents typés "demande-modification", "dossier-modification", etc.). Ainsi, ces modèles ne sont pas facilement exploitables par les entreprises. D'autre part, ces modèles sont très spécifiques car ils expriment, tel est le cas du modèle produit, un cas particulier de trois niveaux de produits, quatre nomenclatures, etc.

Il est clair donc qu'on ne peut proposer *un* modèle produit et *un* modèle processus du SIP. Nous pouvons uniquement exprimer la variabilité autour des concepts décrivant les produits et les processus. L'idée serait de garder les modèles obtenus généraux, voire plus généraux qu'ils le sont actuellement en exprimant d'un côté l'ensemble des concepts communs à toutes les entreprises et de l'autre côté la variabilité des organisations autour des concepts spécifiques et de fournir ensuite un ensemble de patrons, permettant chacun de construire un fragment du modèle SIP, selon la spécificité de l'entreprise. C'est sur la base de cette variabilité que les patrons du SIP sont construits, tel que nous l'avons décrit dans la démarche d'ingénierie de patrons proposée au chapitre III. Cette démarche d'ingénierie de patrons, rappelons-le, repose sur trois étapes :

- une étape d'analyse de domaine ayant pour objectif de recenser l'ensemble des besoins informationnels souvent exprimés par les utilisateurs du SIP (connaissances domaine) ainsi que l'ensemble des techniques de spécification détenues par les concepteurs du SIP chargés d'analyser et de spécifier les besoins des utilisateurs du futur SIP (connaissances développement). Cette analyse aboutit à un référentiel du domaine qui constitue une source de connaissances susceptibles de contenir des éléments réutilisables.
- une étape d'identification des divers problèmes souvent rencontrés lors de l'ingénierie des SIP (et donc des divers patrons) en distinguant dans le référentiel les divers blocs d'information permanents du SIP ainsi que les divers points de variabilité associés à chacun de ces blocs. Il s'agit donc d'identifier les éléments susceptibles d'êtres réutilisables et qui soient les plus indépendants possibles les uns des autres.
- une étape de spécification de patrons proposant des solutions aux problèmes identifiés à la seconde étape.

Ainsi, la partie précédente du manuscrit (2ème partie) a illustré la première étape de cette démarche (analyse de domaine) en identifiant l'ensemble des sources de connaissance

susceptibles de contenir des éléments réutilisables. Elle a également initié la deuxième étape de cette démarche (identification des éléments réutilisables) en mettant en évidence les divers blocs de concepts et les points de variabilité associés.

La présente partie (3ème partie) a pour objectif de proposer un catalogue de patrons résolvant les problèmes récurrents du SIP identifiés et d'illustrer ainsi la dernière étape du processus d'ingénierie de patrons. Elle est composée de deux chapitres :

- le chapitre VI a pour objectif d'organiser, dans des patrons, l'ensemble des connaissances identifiées aux chapitres IV et V. Ces patrons doivent intégrer d'une part les connaissances de domaine (besoins en spécification) et d'autre part les connaissances développement (les façons pour spécifier ces besoins).
- le chapitre VII présente une application des patrons proposés sur un projet industriel et propose un outil logiciel support à la démarche de spécification de SIP par réutilisation de patrons ainsi proposée.

Chapitre VI :

Le Catalogue de Patrons

1. Introduction

Le présent chapitre a pour objectif d'organiser, dans des patrons, l'ensemble des connaissances identifiées aux chapitres IV et V, classées selon les besoins informationnels "permanents" et "variants" dans le SIP.
Dans la première partie de ce chapitre, nous présentons l'architecture du catalogue de patrons développé (§2). Nous présentons d'abord une typologie des patrons qui le constituent (§2.1) et nous décrivons ensuite le formalisme retenu pour formaliser et spécifier les différents patrons du catalogue (§2.2). Nous mettrons en évidence dans cette description les divers liens qui existent entre les patrons et permettant de les organiser au sein du catalogue.
La seconde partie de ce chapitre (§3) est consacrée à la présentation des divers patrons constituant le catalogue.

2. Architecture du catalogue

Avant de présenter le catalogue de patrons, nous souhaitons préciser la sémantique du terme "catalogue". Un catalogue désigne un groupement de patrons répondant à une problématique commune. D'autres termes sont aussi utilisés pour désigner de telles collections de patrons. On parle aussi de famille, de système et de langage. Il s'agit dans tous les cas d'une collection de patrons s'adressant à un même domaine dont l'application est assurée par des règles qui permettent de les combiner. Cette collection doit être suffisamment complète et organisée pour parvenir à répondre à la problématique visée. Tout comme le concept de patron, la notion de langage, système ou catalogue de patrons a été introduite par C. Alexander qui définit un système de patrons comme "une collection de patrons formant un vocabulaire qui permet de comprendre et communiquer les idées". Les patrons de cette collection sont tissés entre eux dans un tout cohésif qui révèle les structures et les relations inhérentes à chacun de ses composants pour atteindre un objectif commun [Front, 97]. Si un patron est une solution récurrente à un problème dans un contexte donné, alors un système de patrons est une collection de solutions qui coopèrent pour résoudre un problème complexe selon une solution ordonnée pour aboutir à un but prédéfini. Les patrons tiennent rarement une place à eux tout seuls. Les systèmes de patrons sont un ensemble de patrons, classifiés par catégories, étroitement reliés entre eux par des contextes identiques. L'intérêt majeur est de faire apparaître facilement la nécessité d'autres patrons et d'exprimer les différents liens qu'il y a entre eux [Manhes, 98].
Un système de patrons bien élaboré doit permettre aux concepteurs d'avoir la liberté d'exprimer leurs problèmes et concevoir eux-mêmes les solutions qui répondent à leurs besoins particuliers dans le contexte où les patrons peuvent être appliqués.

2.1. Typologie de patrons : Trois catégories de patrons

Les patrons que nous proposons pour l'ingénierie des SIP permettent d'élaborer, par réutilisation, l'ensemble des modèles du SIP (à différents niveaux d'abstraction). Ils sont centrés sur les phases amont d'ingénierie, càd la spécification des SIP. Il s'agit donc de patrons métiers, spécifiques au domaine des SIP, qui couvrent la phase d'analyse des besoins et initialisent la phase de conception. Ils permettent de spécifier complètement le système

d'information organisationnel (SIO) associé au SIP et partiellement le système d'information informatique (SII) associé au SIO. Deux types de patrons sont ainsi mis en jeu : des patrons d'analyse et des patrons de conception.

- **Les patrons d'analyse** permettent d'identifier et de spécifier les besoins informationnels souvent exprimés par les utilisateurs du SIP, càd les objets du SIO, pour aboutir à un modèle de spécification fonctionnelle ou *modèle d'analyse* décrivant les objets du domaine (avec leur comportement intrinsèque et leurs relations). Ils mettent en jeu les connaissances de domaine (les besoins informationnels), détenues par les utilisateurs du SIP mais également les connaissances de développement, détenues par les concepteurs du SIP afin de spécifier l'ensemble des connaissances de domaine.
- **Les patrons de conception** ont pour objectif de spécifier les objets du SII associés aux objets du SIO préalablement identifiés (à l'aide des patrons d'analyse), pour aboutir à un modèle de spécification normalisé ou *modèle de conception*. Ils définissent les traitements informatiques associés aux besoins exprimés par les utilisateurs du SIP. Ceci devrait permettre de compléter la description des objets d'analyse identifiés par les patrons d'analyse (notamment en terme d'opérations de classe), de définir de nouveaux objets du SI relatifs à la conception et éventuellement de transformer certains objets d'analyse en objets de conception (telle que par exemple la transformation d'un lien entre classes dans le modèle d'analyse en une classe dans le modèle de conception). Pour ce dernier point, notons qu'il existe des patrons généraux de conception orientée objet qui résolvent ce problème.

L'analyse de domaine précédemment présentée dans la $2^{ème}$ partie du manuscrit a permis d'identifier les besoins informationnels exprimés par les utilisateurs du SIP. Deux types de connaissances de domaine sont mises en jeu : les connaissances produit (éléments de description des produits) et les connaissances processus (éléments de description des processus, en terme de workflow). L'objectif des patrons d'analyse est de capitaliser et faciliter la réutilisation de ces connaissances.

Pour les patrons de conception, il faut partir de l'analyse des traitements du SII associé au SIO. En partant de la description des processus métier, mis en évidence par les patrons d'analyse, les processus informatiques associés càd les traitements du SII peuvent être déterminés. L'analyse des traitements du SII identifiés permet de retrouver les objets du SII (parmi eux figurent évidemment les objets du SIO qui sont à compléter ou à transformer éventuellement). Il s'agit donc de partir des processus métier pour retrouver les objets du SII.
Cette démarche se retrouve dans l'analyse de domaine effectuée, qui, outre les deux dimensions produit et processus, a fait ressortir une troisième dimension, relative à l'impact des processus SIP sur les connaissances produits, en terme d'évolution de ces connaissances (états et indices de version\révision\correction). Elle met en évidence les mécanismes qui assurent cette évolution et qui sont issus de la description des processus métiers qui, rappelons le, consistent essentiellement en la définition et l'évolution des informations produit. Ces derniers sont en effet décrits en terme d'activités transformant des éléments d'entrée en éléments de sortie. Ces éléments ne sont autre que des objets du domaine décrivant le produit (connaissances produit). Lors de l'exécution des diverses activités du processus, ces objets changent d'état ou d'indice de version/révision et interagissent entre eux. L'analyse détaillée des processus SIP (en partant des processus métiers jusqu'aux processus informatiques

associés) permet donc de compléter la description de la dynamique des concepts produit, en terme d'indices d'évolution, de changement d'états et d'opérations sur les classes. Il est à noter que la dynamique des concepts produit est de deux types :

- celle liée à la structure de ces objets, traduisant le comportement intrinsèque des objets (tels que la création, la suppression de l'objet). Cette dynamique est plus ou moins standard, dans le sens où elle est indépendante des processus métiers agissant sur les objets. Elle peut être décrite avec la description des objets lors de l'élaboration du modèle d'analyse.
- celle liée aux processus métiers agissant sur les objets, traduisant le comportement des objets en réponse à l'exécution d'un processus particulier. Cette dynamique est donc spécifique au métier. Elle ne peut être décrite qu'à l'issue de l'analyse des processus métiers.

L'analyse des processus permet donc de compléter la description des objets d'analyse existants mais également de déterminer de nouveaux objets du SII.

Finalement, nous organisons les patrons SIP en trois catégories (cf. Figure 6.1) :

- des **patrons d'analyse produit** qui fournissent des fragments de modèle pour représenter les produits. Ils permettent de fixer les concepts gérés dans le SIP, càd les objets du SIO (tel que les niveaux de produit, les documents techniques, les nomenclatures, etc.) en terme de classes et de relations entre les classes.
- des **patrons d'analyse processus** qui offrent une manière pour représenter les processus métier du SIP. Ils permettent d'analyser les processus en mettant en évidence les activités qui les composent, les éléments entrants et sortants mis en jeu ainsi que les ressources employées dans chacune des activités.
- des **patrons de conception** qui offrent une démarche permettant d'identifer, à partir de la décomposition des processus métiers, les objets du SII afin d'aboutir au modèle de conception du SIP. Cette démarche est basée sur la décomposition des processus métiers jusqu'à arriver aux processus informatiques associés (donc les traitements du SII) et ensuite sur la mise en évidence des collaborations entre les objets du SII pour réaliser les processus informatiques identifiés.

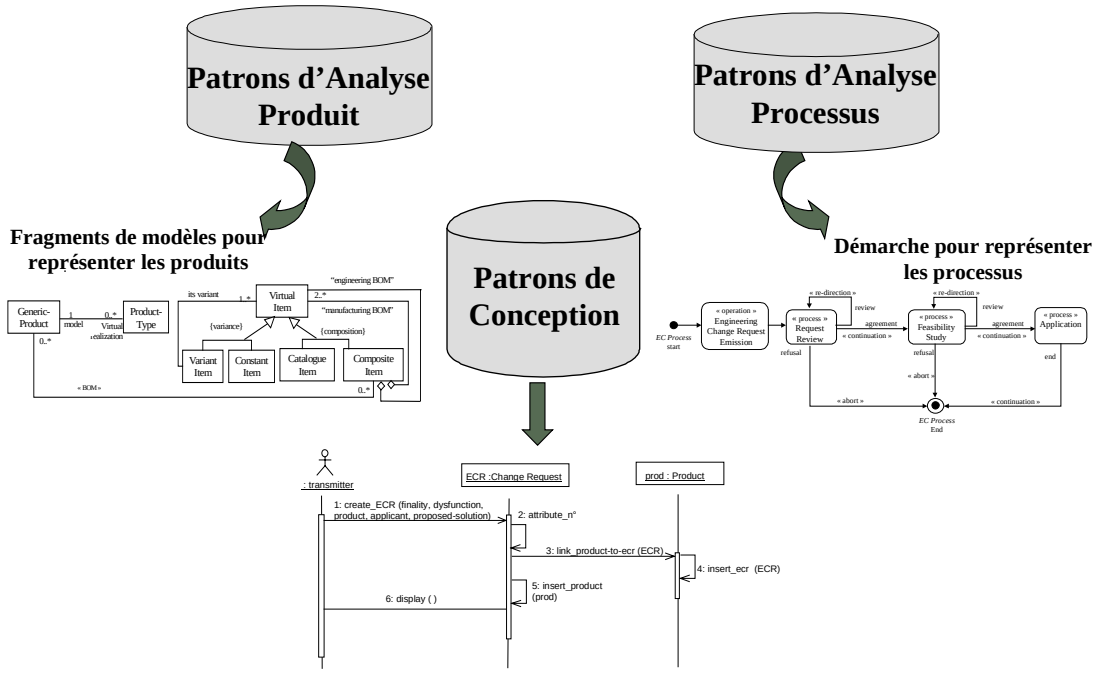


Figure 6.1 - Trois catégories de patrons

Les patrons d'analyse produit concourent surtout à capitaliser des fragments de modèle (savoir) alors que les patrons d'analyse processus et les patrons de conception concourent plutôt à capitaliser et réutiliser des fragments de démarche (savoir-faire). Nous soulignons toutefois que cette caractérisation est propre à notre catalogue et ne peut être généralisée. A titre d'exemple, les patrons de conception proposés par E. Gamma capitalisent plutôt des modèles.

Dans la suite (§3), nous présentons chacun de ces trois types de patrons mais avant cela, nous définissons le formalisme utilisé pour décrire les divers patrons (§2.2).

2.2. Formalisme des patrons

De nombreux formalismes ont été proposés dans la littérature pour représenter un patron. Un formalisme est la forme, la structure que tout auteur utilise pour représenter ses patrons [Duong, 00]. C'est ainsi que chaque auteur propose son propre formalisme pour mettre en forme ses patrons. P.Q. Duong [Duong, 00] identifie deux classes de formalismes : les formalismes *narratifs* tels que ceux de C.Alexander [Alexander, 77] et de Portland [Portland, 00] et plus récemment les formalismes *structurés* (càd composés par un ensemble de rubriques plus ou moins détaillés) tels que ceux de M. Fowler [Fowler, 97] pour les patrons d'analyse, d'E. Gamma [Gamma, 95] et de P. Coad [Coad, 96] pour les patrons de conception et plus récemment de S.W. Ambler [Ambler, 98] pour les patrons processus[141]. Les formalismes structurés offrent une meilleure représentation des patrons, qui sont souvent complexes et nécessitent une description généralement longue.

La majorité des formalismes proposés sont globalement équivalents. Ils diffèrent par le nombre et le degré de détail plus ou moins élevé des rubriques qu'ils proposent mais ils intègrent tous le triplet {problème, contexte, solution} pour exprimer un certain *problème* et

[141] Par opposition à "patrons produit" (cf. chapitre II - §3.2.2.3).

sa *solution* dans un *contexte* donné. A ce propos, C. Rolland [Rolland, 98b] précise que pour être explicite et précis, un patron doit définir le problème avec les forces qui influent le problème, une solution concrète et son contexte[142]. Et pour qu'il soit visualisable, le patron doit être exprimé à l'aide d'expressions en langue naturelle, de dessins, de modèles conceptuels, etc.

Dans le cadre des activités de l'équipe Sigma du laboratoire LSR, et partant de la volonté de définir un formalisme commun au niveau de l'équipe, un travail collectif a abouti à la définition d'un formalisme structuré de représentation de patrons. Ce travail avait pour objectif :

- d'homogénéiser les formalismes utilisés au niveau de l'équipe,
- de proposer de nouvelles rubriques facilitant la réutilisation de patrons et permettant de mieux les organiser,
- de proposer un formalisme commun pour exprimer des fragments de modèles et des fragments de démarche,
- d'extraire une sémantique commune à certains formalismes proposés dans la littérature.

Le formalisme proposé est constitué de 3 parties : **Interface**, **Réalisation** et **Relations**. Chaque partie regroupe un certain nombre de rubriques. Une rubrique est composée d'un ou de plusieurs champs de nature différente (texte, diagramme UML) ce qui explique qu'une rubrique peut être représentée de différentes manières.
Les rubriques *interface* permettent de sélectionner le patron dans un catalogue de patrons ; les rubriques *réalisation* permettent de résoudre le problème soulevé par le patron et les rubriques *relations* permettent d'organiser les patrons dans un catalogue en définissant les relations qu'entretient un patron avec d'autres patrons. Dans ce qui suit, nous détaillons les rubriques de chacune des trois parties du formalisme. Pour mieux illustrer chacune de ces parties, nous utilisons le patron "Deux Niveaux de Produit" développé dans le catalogue SIP. Ce patron a pour objectif de représenter deux niveaux d'abstraction du produit industriel.

2.2.1. Rubriques "Interface"

L'interface de patrons se compose de cinq rubriques. Elle servent à la sélection des patrons. Ces rubriques sont les suivantes :

- **Nom** : nom du patron.
- **Classification** : un ensemble de mots clés du domaine (en occurrence le SIP) qui donne intuitivement la classification du domaine. Dans le cadre des SIP, nous définissons trois classes : les patrons d'analyse produit, les patrons d'analyse processus et les patrons de conception.
- **Problème** : chaque patron adresse un problème particulier qui a besoin d'être résolu.
- **Motivation** : un scénario (un exemple) d'application du patron décrit de manière textuelle et éventuellement graphique.
- **Forces** : cette rubrique discute de la pertinence et de la force de la solution. Elle décrit les raisons pour proposer le patron, càd pour résoudre le problème soulevé avec la solution proposée. Elle exprime les contraintes à prendre en compte lors du choix de la solution au problème soulevé.

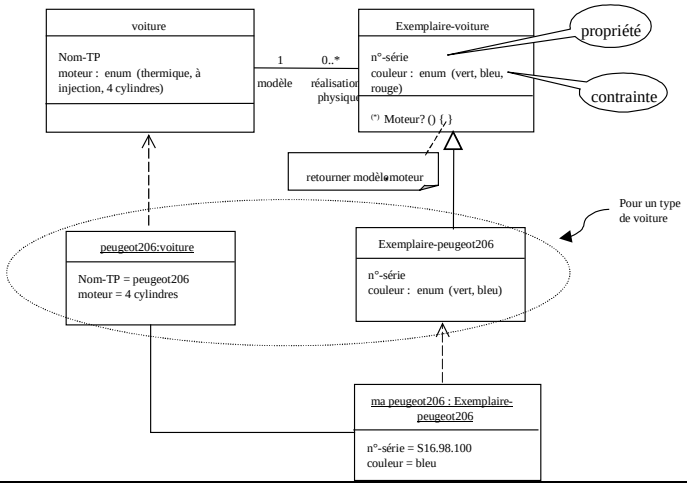
[142] Un contexte désigne une ensemble récurrent de situations dans lequel le patron est appliqué [Rolland, 98b].

- **Contexte** : pré-conditions pour l'application du patron. Il peut être décrit comme un ensemble de modèles issus ou non de l'application d'autres patrons.

L'exemple suivant illustre les rubriques "interface" du patron "Deux Niveaux de Produit" :

<table>
<tr><td>Nom</td><td>Deux Niveaux de Produit</td></tr>
<tr><td>Classification</td><td>patron d'analyse produit</td></tr>
<tr><td>Problème</td><td>Ce patron permet de représenter le concept de produit en deux niveaux d'abstraction permettant d'une part de partitionner les connaissances relatives aux différents niveaux et d'autre part de définir des relations entre les niveaux afin de permettre la propagation de propriétés entre les niveaux.</td></tr>
<tr><td>Motivation</td><td>Nous considérons l'exemple suivant pour le produit "voiture". Nous illustrons le cas des deux niveaux produit-physique et type-produit.
<table>
<tr><th>Niveau de produit</th><th>Peugeot 206 (type)</th><th>Ma peugeot 206 (exemplaire)</th></tr>
<tr><td>caractéristiques</td><td>Nom TP = peugeot 206
Moteur = 4 cylindres
Couleur = vert ou bleu</td><td>Nom EP = peugeot 206
Moteur = 4 cylindres
Couleur = bleu
N° série = 206-98-100</td></tr>
</table>
Le diagramme de classes ci-dessous organise les connaissances rattachées aux deux niveaux de produit ci-dessus décrits.

La classe "voiture" détient les propriétés de tous les types de voiture. Peugeot206 est une instance de cette classe.
La classe "exemplaire-voiture" détient les propriétés de tous les exemplaires de voitures.



</td></tr>
<tr><td>Force</td><td>Ce patron relie une classe de catégories abstraites A (modèles de voitures) à une classe d'objets plus concrets C (voitures individuelles). Sa sémantique est définie en terme des abstractions "est-un" (généralisation) et "est-de" et des correspondances classe/métaclasse.</td></tr>
<tr><td>Contexte</td><td>Ce patron n'exige aucun autre patron ou modèle pour être appliqué. Cependant, le concepteur doit déjà connaître le nombre de niveaux de produit. Ce nombre doit être de 2.</td></tr>
</table>

2.2.2. Rubriques "Réalisation"

La réalisation de patrons sert à exprimer la solution offerte par le patron pour résoudre le problème traité. Nous proposons de distinguer deux types de "solution" : la *solution-modèle* et la *solution-démarche*. Dans le chapitre II (§ 3.2.2.3), nous avons souligné qu'un patron

capitalise des connaissances de développement pour résoudre un problème de spécification (d'une connaissance de domaine). Ces connaissances de développement portent aussi bien sur des produits de développement c'est-à-dire des modèles que sur des processus de développement c'est-à-dire des démarches. Ainsi, le concept de patron est un composant orienté "modèle" ou savoir (rubrique « solution-modèle ») mais aussi orienté "démarche" ou savoir-faire (rubrique « solution-démarche »).

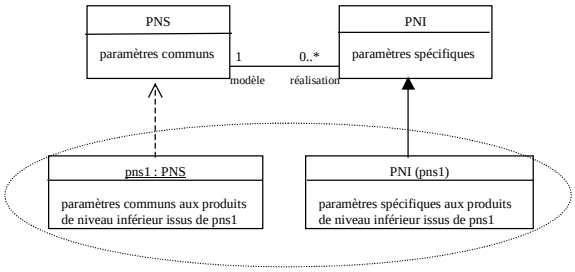
- La rubrique **Solution-modèle** exprime la solution proposée par le patron à l'aide de diagrammes UML décrivant en particulier les participants au patron ainsi que leurs collaborations. Dans le cas de diagrammes de classes, les contraintes sur les entités du modèle sont exprimées, selon le cas, soit à l'aide d'invariants de classe (contraintes qui doivent être respectées par toutes les instances d'une classe) soit à l'aide de pré et de post-conditions sur les opérations. Pour cela, nous avons choisi d'utiliser un langage pseudo-mathématique (OCL[143]) lorsqu'il s'agit de contraintes simples, et la langue naturelle lorsqu'il s'agit de contraintes plus complexes à exprimer.
- La rubrique **Solution-Démarche** exprime la démarche proposée par le patron pour résoudre le problème. Lorsque le problème posé est complexe et que la solution présente un comportement riche ou encore lorsqu'il existe différentes solutions alternatives selon les cas, le patron fait appel soit à une technique de décomposition du problème en sous-problèmes plus élémentaires résolus par d'autres patrons, soit à une technique de raffinement du problème en problèmes plus concrets résolus également par d'autres patrons. Des diagrammes d'activités UML peuvent être utilisés pour décrire la solution-démarche où une activité consiste soit à faire appel à d'autres patrons soit à exécuter certaines tâches.

Soulignons qu'un patron peut proposer un modèle en solution au problème qu'il pose ainsi que la manière d'obtenir ou d'adapter ce modèle (solution-modèle et solution-démarche). Il peut également ne pas proposer de modèle en solution au problème traité mais uniquement une démarche de spécification (solution-démarche).

L'exemple suivant illustre les rubriques "réalisation" du patron "Deux Niveaux de Produit" :

Solution-Démarche	1. Une entreprise articule ses activités autour de deux niveaux d'abstraction du produit. Deux classes abstraites interviennent : ▪ Produit Niveau Supérieur "PNS", détenant les propriétés communes des produits de niveau inférieur dérivant de celui-ci et contenant une ensemble de valeurs possibles de certaines propriétés. ▪ Produit Niveau Inférieur "PNI", détenant les propriétés spécifiques de produits de niveau inférieur associés à un produit de niveau supérieur et contenant des valeurs fixes de certaines propriétés. L'association entre ces deux classes a pour cadinalités : 1 du côté de la classe "PNS" et 0..* du côté de la classe PNI. Deux scénarios peuvent exister : "PNS" est un produit-générique & "PNI" est un type-produit ou "PNS" est un type-produit & "PNI" est un produit-physique. 2. Quand l'entreprise décide de créer un nouveau "Produit Niveau Supérieur" (par exemple le nouveau type de produit *peugeot 206*) soit *pns1*. *pns1* est représenté par une instance de la

[143] OCL (Object Contraint Language) est le langage d'expression pour UML. Il est présenté en Annexe A.

<table>
<tr><td></td><td>classe "PNS". Ceci se traduit par l'objet pns1:PNS.
3. Avec la création du nouveau produit pns1, il y aura différents produits de niveau inférieur dérivant de pns1 (par exemple différents exemplaires de peugeot 206). On crée alors une classe concrète pour tous les produits de niveau inférieur dérivant de pns1, soit la classe "PNI (pns1)". Cette classe contient les propriétés spécifiques aux produits de niveau inférieur dérivant de pns1.</td></tr>
<tr><td>Solution-Modèle</td><td> 

La création d'un nouveau PNS, soit pns1 (par exemple peugeot206), implique la création de divers produits de niveau inférieur issus de pns1 (divers exemplaires de peugeot 206).
Toute création d'un nouveau produit de niveau supérieur génère une instance de la classe "PNS" et une sous-classe de la classe "PNI".</td></tr>
</table>

2.2.3. Rubriques "Relations"

Lorsque le problème soulevé par un patron est complexe et que la solution offerte présente un comportement riche ou présente différentes alternatives, la solution-démarche du patron fait appel à d'autres patrons pour résoudre le problème traité (cf. §2.2.2 ci-dessus). Il est alors fondamental de mettre en évidence les relations entre patrons de manière à fournir un guide méthodologique modélisant la démarche même de développement (rubrique « solution-démarche »). D'ailleurs, c'est sur cette idée de collection de patrons liés (donc de catalogue ou de langage) que les démarches d'ingénierie de systèmes par réutilisation de patrons sont basées et non pas sur des patrons isolés, car c'est l'ensemble des patrons liés qui concourt à la résolution du problème global considéré (en l'occurrence, la spécification d'un SIP) et non pas chacun des patrons pris isolément[144]. G. Meszaros [Meszaros, 00] présente le concept de langage de patrons comme une collection de patrons reliés les uns aux autres dans l'objectif de résoudre un même problème ou de fournir des parties d'une solution adressant un problème plus large et partitionné[145]. C. Rolland [Rolland, 98b] précise qu'un patron n'est pas une entité isolée. En effet, lorsqu'une solution complexe ne peut être décrite dans un patron unique ou lorsqu'une seule solution est très spécifique et non partageable alors la totalité du problème et sa solution complexe doivent être décomposées en un certain nombre de problèmes et leurs solutions respectives, obtenant ainsi un langage de patrons[146]. Par ailleurs, différents patron d'un langage peuvent être combinés de diverses manières (alternatives) pour adopter différents chemins de résolution à différentes facettes du problème global.
La démarche d'ingénierie d'un SI (càd de résolution d'un problème d'ingénierie) repose donc sur la combinaison de patrons et donc sur cette notion de relations diverses entre patrons d'un catalogue.

---

[144] Par ailleurs, il est tout à fait évident qu'un problème d'ingénierie de SI, avec la complexité qu'il présente, ne peut être résolu dans un patron unique adressant la totalité du patron et proposant la totalité de la solution.

[145] Traduction de : a pattern language is a collection of patterns that are related to each other by vertue of solving the same problems or parts of a solution to a larger, partitioned problem.

[146] Soulignons toutefois que le patron doit être le plus autonome possible vis à vis des autres patrons du langage, càd qu'il puisse fournir une entité isolable et être réutilisable tout seul pour résoudre un problème élémentaire.

A ce titre, nous pouvons affirmer que les patrons constituent un guide d'ingénierie (de résolution de problèmes plus généralement) car ils permettent d'organiser hiérarchiquement (et fonctionnellement) les problèmes et les manières de les résoudre.

En termes de relations inter-patrons, nous partons du travail de D. Rieu [Rieu, 99d] qui propose un ensemble d'opérations permettant de générer de nouveaux patrons à partir de patrons existants. Le principe de base consiste à partir du fait qu'il est rare aujourd'hui qu'un problème ne puisse être solutionné (du moins en partie) par des patrons existants. C'est ainsi qu'un patron pour représenter des nomenclatures de produit est issu du patron de conception "composite" qui a pour objectif de représenter des compositions récursives d'objets. L'ensemble des opérations ainsi définies permettent de positionner à la fois le problème du patron par rapport aux problèmes existants et la solution proposée par rapport aux solutions existantes. Ces mêmes opérations peuvent être vues comme des relations entre les divers patrons impliqués. Quatre types de relations sont ainsi définies :

- Alternative : Un patron A est une alternative d'un patron B, si A a le même problème que B mais propose une solution différente. Les deux patrons ont le même contexte, la même classification mais des forces différentes.
- Raffine : un patron A "raffine" un patron B, si le problème posé par A est une spécialisation (un cas particulier) de celui posé par B. B peut résoudre les problèmes posés par A. La force et le contexte de A sont enrichis par rapport à ceux de B. On dit aussi que B "étend" A.
- Requiert : un patron A "requiert" un patron B, si l'application de B est un pre-requis à l'application de A. Ainsi, la solution-modèle du patron B fait partie du contexte du patron A et/ou le patron B paraît dans le contexte du patron A.
- Utilise : un patron A "utilise" un patron B, si une partie des problèmes posés par A peuvent être résolus en partie ou complètement par B. Ainsi, la solution-démarche de A est exprimée en "utilisant" le patron B.

De la même façon que ces relations sont utilisées pour représenter les liens entre des patrons existants (proposés dans la littérature) et de nouveaux patrons issus de ces derniers, nous pouvons utiliser ces mêmes relations pour définir les liens entre les divers patrons du catalogue de domaine SIP proposé. Dans ce catalogue, trois types de relations sont utilisées : utilise, requiert et raffine.

Ainsi, la partie du formalisme de patrons relative aux relations d'un patron peut être constituée de trois rubriques :

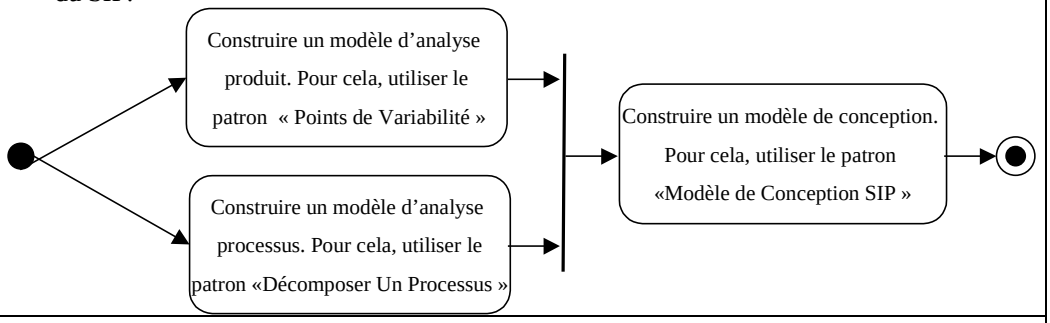
- **Utilise** : liste des patrons "utilisés" par le patron considéré.
- **Requiert** : liste des patrons "requis" par le patron considéré.
- **Raffine** : liste des patrons "raffinés" par le patron "considéré".

L'organisation des patrons par des relations devrait permettre la description de systèmes mieux construits et donc augmenter leur taux de réutilisation.
Pour illustrer les liens entre les patrons, nous utilisons les collaborations d'UML. Un patron est représenté par une collaboration et les relations entre patrons sont matérialisés par des liens entre collaborations, stérétoypés selon leur nature (utilise, requiert, raffine).

3. Zoom sur le catalogue

Nous présentons dans cette partie une vue d'ensemble des patrons proposés. Nous ne présentons pas en détail tous les patrons. Dans la plupart des cas, nous décrivons brièvement le problème, le contexte et la solution de chacun des patrons afin de donner un aperçu général des divers patrons et illustrer l'architecture globale du catalogue. Nous développons toutefois davantage quelques patrons afin de permettre la compréhension de l'ensemble du catalogue et faciliter l'introduction de divers autres patrons. La description complète de l'ensemble des patrons est proposée dans l'Annexe D.
Cette partie est organisée selon les trois types de patrons identifiés dans le paragraphe 2.1. Nous présentons successivement les patrons d'analyse produit, les patrons d'analyse processus et les patrons de conception. Toutefois et avant de passer à la description de chacun de ces trois types de patrons, nous présentons le patron "Spécifier un SIP" qui a pour objectif de guider l'utilisation du catalogue de patrons et constitue ainsi le point d'entrée de ce catalogue.

3.1. Le patron "Spécifier un SIP"

Nom	Spécifier un SIP
Problème	Guider le concepteur du SIP dans l'utilisation des divers patrons du catalogue SIP.
Motivation	Une démarche de spécification de SIP suit un ensemble plus ou moins ordonné d'étapes. Ainsi, pour construire un modèle de conception illustrant les objets informatiques mis en collaboration dans le Système d'Information Informatisé (SII) nécessite de connaître les objets métiers du Système d'Information Organisationnel (SIO) ainsi que les traitements attendus du SII. Ces derniers ne peuvent être identifiés qu'à l'issue de la description des processus métier du SIO. Par ailleurs la description des processus métiers du SIO peut nécessiter la connaissance des objets métiers mis en jeu dans le SIO.
Force	Ce patron guide l'analyse et la conception du SIP d'une façon rigoureuse.
Contexte	Ce patron ne nécessite aucun autre patron ou modèle pour être appliqué.
Solution-démarche	1. identifier les objets métiers associés au produit et gérés dans le SIO. Les structurer dans un modèle d'analyse produit. Pour cela appliquer le patron "Points de Variabilité" 2. représenter les processus gérés dans le SIO. Pour cela, appliquer le patron "Décomposer Un Processus". 3. Construire le modèle de conception représentant les objets du SII associé au SIO. Pour cela appliquer le patron "Modèle de Conception SIP". L'application de ce patron nécessite la description préalable des objets métiers associés au produit ainsi que les processus métiers du SIP. 
Solution-modèle	A l'application de la solution-démarche, on obtient le modèle d'analyse produit, le modèle d'analyse processus et le modèle de conception du SIP.
Utilise	{Points de Variabilité, Décomposer Un Processus, Modèle de Conception SIP}

Le patron "Spécifier un SIP" propose ainsi d'utiliser d'abord les patrons "Points de Variabilité" et "Décomposer Un Processus" pour construire respectivement le modèle d'analyse produit et le modèle d'analyse processus. Il n'impose pas un ordre quant à l'application de ces deux patrons. Il est néanmoins souhaitable de construire le modèle d'analyse produit en premier lieu pour pouvoir renseigner dans le modèle de décomposition du processus les objets métiers mis en jeu dans les diverses activités qui le composent. Par ailleurs, ce patron "Spécifier un SIP" suppose l'application préalable des deux patrons "Points de Variabilité" et "Décomposer un processus" pour appliquer le patron "Modèle de Conception SIP".
Dans la suite, nous décrivons donc chacun des trois patrons utilisés par le patron "Spécifier un SIP" et qui sont en fait les points d'entrée de chacun des trois catégories de patrons du catalogue SIP.

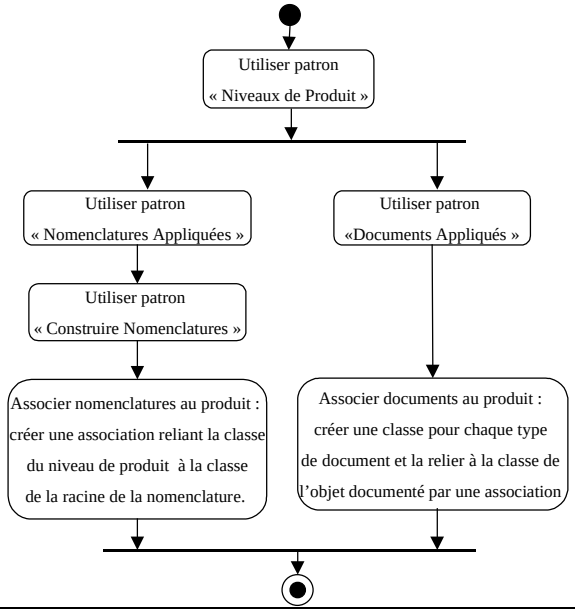
3.2. Patrons d'Analyse Produit

L'ensemble des patrons produit visent à guider un concepteur de SIP dans la représentation et la réutilisation des concepts produits, gérés dans le SIP. Ils permettent de construire le modèle statique du produit, en fixant les objets métiers gérés dans le SIP et en les structurant en terme de classes et de relations entre les classes.

3.2.1. Point d'entrée : Le patron "Points de Variabilité"

Nous avons conclu à l'issue de l'analyse de domaine sur l'existence de divers blocs de concepts gérés en permanence dans les SIP, et qui sont relatifs à divers objets produit communs à tous les SIP. Nous avons par ailleurs souligné l'existence de divers points de variabilité à l'intérieur de chacun des blocs. Ces points de variabilité expriment la variabilité autour de chacun des objets produit considérés, selon la spécificité des organisations industrielles. La première étape dans la construction du modèle produit consiste à *distinguer les divers blocs* et à *fixer les divers points de variabilité* dans chaque bloc selon la spécificité de l'entreprise. Cette étape est traitée par le patron "Points de Variabilité".

Nom	Points de Variabilité
Classification	Patron d'analyse produit.
Problème	Identifier tous les blocs du modèle SIP et fixer les points de variabilité à l'intérieur de chaque bloc selon la spécificité de l'entreprise.
Motivation	Chaque organisation gère des produits et associe diverses nomenclatures et des documents à ces produits. Toutefois, le nombre de niveaux d'abstraction de ces produits varient d'une entreprise à une autre. Certaines organisations gèrent des types de produits indépendants, d'autres gèrent en plus des produits génériques souvent appelés lignes ou gammes de produits. Dans ces deux cas, les exemplaires de produit peuvent ou non être gérés selon l'aspect critique des exemplaires produit et du service après-vente (dans l'industrie aéronautique par exemple, chaque exemplaire d'avion est géré tout au long de sa vie pour garder trace des conditions de sa fabrication, des opérations de maintenance, etc. alors que dans la production de masse tel que l'industrie de stylos, les exemplaires ne sont pas gérés). Par ailleurs, le nombre et le type de nomenclatures utilisées pour décrire le produit varient aussi d'une entreprise à l'autre. Dans certaines organisations, un type de produit a une seule nomenclature organique alors que dans d'autres organisations, on distingue la nomenclature d'étude, la nomenclature industrielle, la nomenclature logistique pour un type de produit. Il en est de même pour les documents dont le

	type (le contenu) et le nombre varient d'une entreprise à une autre.
Force	Ce patron guide la modélisation du SIP d'une façon rigoureuse et uniforme. Toutefois, il ne permet pas d'introduire de nouveaux concepts. L'apparition de nouveaux concepts exige une révision du catalogue de patrons et en particulier ce patron.
Contexte	Ce patron ne nécessite aucun autre patron ou modèle pour être appliqué.
Solution-Démarche	1. Fixer le nombre et le type de niveaux de produit (générique, type, exemplaire). Pour cela, appliquer le patron "Niveaux de Produit". 2. Pour les Nomenclatures : ▪ Fixer les types de nomenclatures appliquées. Pour cela, appliquer le patron "Nomenclatures Appliquées"; ▪ Construire les Nomenclatures décrivant votre produit. Pour cela, appliquer le patron "Construire Nomenclature"; ▪ Associer chaque Nomenclature au niveau de produit approprié. Pour cela, créer une association qui lie la classe du niveau de produit considéré à la classe qui correspond à la racine de la nomenclature considérée. 3. Pour les documents : ▪ Fixer le type de documents appliqués dans votre entreprise. Pour cela appliquer le patron "Documents Appliqués"; ▪ Associer les types de documents identifiés aux objets appropriés. Pour cela, créer une classe pour chaque type de document identifié et créer une association qui lie cette classe de document à l'objet qu'il documente (qui peut être un niveau donné de produit, un article, une fonction, un feature). Selon que le type de document est de nature "dépendant" ou "non-dépendant", associer un stéréotype à la classe associée (dépendant, non-dépendant) et baptiser l'association par "décrit" (cardinalité 1 du coté de la classe associée à l'objet documenté) ou "documente" (cardinalité 0..* du coté de la classe associée à l'objet documenté). La figure suivante illustre l'ensemble de la démarche : 
Solution-Modèle	A l'application de la solution-démarche, on obtient un modèle d'analyse produit complet.
Utilise	{Niveaux de Produit, Nomenclatures Appliquées, Construire Nomenclature, Documents Appliqués}

La résolution du problème posé par le patron "Points de Variabilité" est complexe, dans le sens où plusieurs cas sont à distinguer, en fonction desquels des solutions différentes sont proposées. Le patron considéré décompose alors le problème principal, qui concerne les points de variabilité des divers blocs, en des sous problèmes relatifs chacun aux points de variabilité d'un bloc donné. Il fait ensuite appel à un ensemble de patrons pour résoudre l'ensemble de ces sous-problèmes. Il suggère ainsi de fixer d'abord les niveaux de produits gérés dans le PIS (patron "Niveaux de Produit"). Ensuite, une fois les niveaux de produits fixés, le concepteur de SIP peut fixer les nomenclatures associées à chacun des niveaux de produit (patron "Nomenclatures Appliquées"), construire ces nomenclatures (patron "Construire Nomenclature") et ensuite les rattacher aux niveaux appropriés (activité Associer Nomenclature à Produit dans le patron "Points de Variabilité"). Enfin, le concepteur du SIP peut fixer les documents associés à chacun des niveaux de produits et éventuellement aux constituants du produit, tels que les articles et les fonctions (patron "Documents Appliquées"). Il peut ensuite rattacher les documents identifiés aux divers éléments concernés (activité *Associer Document à Produit* dans le patron "Points de Variabilité").

La Figure 6.2 illustre les patrons "utilisés" par le patron "Points de Variabilité", à l'aide de collaborations UML liées par des relations du type "utilise" et "requiert".

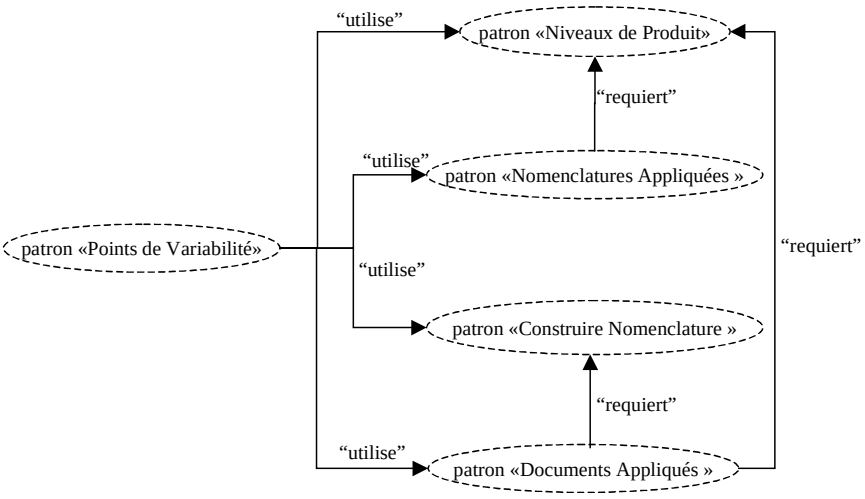


Figure 6.2 - Liens d'utilisation dans le patron "Points de Variabilité"

Remarquons que le séquencement donné dans la solution-démarche du patron "Points de Variabilité" est uniquement proposé et non imposé. Seul le lien "requiert" impose un ordre d'exécution.
Cet ordre dans la résolution du problème (d'abord fixer les niveaux, ensuite fixer les nomenclatures et les attacher aux niveaux; enfin fixer les documents et les attacher aux niveaux) découle du fait que les divers niveaux de produit impliquent l'existence de nomenclatures différentes et de documents différents. A titre d'exemple, le produit physique est structuré à l'aide de nomenclatures organiques physiques (à base d'articles physiques) et il est documenté par des rapports de maintenance par exemple alors que le produit générique est structuré à l'aide de nomenclatures organiques génériques (à base d'articles virtuels) et est documenté par des documents de spécification. Il fallait donc identifier les différents niveaux de produit gérés dans le SIP avant de fixer les nomenclatures et les documents appliquées au produit. C'est ce qui explique les relations du type requiert entre chacun des patrons

"Nomenclatures Appliquées" et "Documents Appliquées" d'une part et le patron "Niveaux de Produit" d'autre part. De la même manière, les documents ne peuvent être fixés qu'une fois les nomenclatures appliquées sont fixées et définies car un document documente, outre les niveaux produit, les objets techniques structurant les produits (tels que article, fonction, feature). Une relation du type "requiert" existe ainsi entre le patron "Documents Appliqués" et "Construire Nomenclature".

Par ailleurs, l'attachement de nomenclatures ou de documents au produit dépend du nombre et du type de niveaux de produit considérés. En effet, une même nomenclature ou un même document peuvent être rattachés différemment au produit, selon les niveaux existants du produit. Par exemple, lorsque le niveau produit-générique existe, la nomenclature organique est définie à ce niveau et elle est ensuite déclinée pour les divers types de produit grâce à des opérations sur les classes (représentant les niveaux de produit) qui propagent les différentes propriétés au travers des associations entre les classes. Il en est de même pour la nomenclature fonctionnelle, identique pour le produit générique et pour les divers types de produit qui en sont issus, et qui est rattachée au produit-générique lorsque celui-ci existe ou rattachée au type produit lorsque le niveau générique n'existe pas. L'activité de rattachement des nomenclatures (ou des documents) vient naturellement après avoir construit les fragments de modèles représentant les nomenclatures (ou les documents) considérées.
Remarquons par contre que le patron "Construire Nomenclature" peut être appliqué indépendamment des autres (il ne requiert aucune exécution préalable d'autres patrons).

Le patron "points de Variabilité" fournit donc en sortie le modèle d'analyse produit global du SIP. L'application complète de ce patron pour obtenir le modèle produit complet passe par l'application complète de tous les autres patrons produit proposés, qu'il suggère d'utiliser.
Dans la suite, nous présentons chacun des quatre patrons jusqu'ici introduits : "Niveaux de produit"; "Nomenclatures Appliquées"; "Construire Nomenclature" et "Documents Appliqués". Nous les regroupons selon le bloc de concepts auquel il réfèrent : *Niveaux de produit* (§3.2.2), *Nomenclatures* (§3.2.3) et *Documents* (§3.2.4).

3.2.2. Bloc : Niveaux de Produit

1.1.1.18. Patron "Niveaux de Produit"

Le patron "Niveaux de Produit" (cf. Annexe D - §1.2) permet d'identifier le nombre et le type de niveaux de produit gérés dans l'entreprise (produit-physique, type-produit et produit-générique) à travers un ensemble de questions destinés au concepteur du SIP. Il propose par ailleurs de représenter les niveaux de produit identifiés. Lorsqu'un seul niveau de produit est identifié, il propose dans sa solution de représenter le niveau considéré. Lorsque le nombre de niveaux identifiés est supérieur à un, il fait appel à l'un des deux patrons : "Trois Niveaux de Produit" ou "Deux Niveaux de Produit" pour représenter respectivement trois niveaux de produit (produit-générique, type-produit et produit-exemplaire) ou deux niveaux de produit (produit-générique & type-produit ou type-produit & produit-exemplaire).

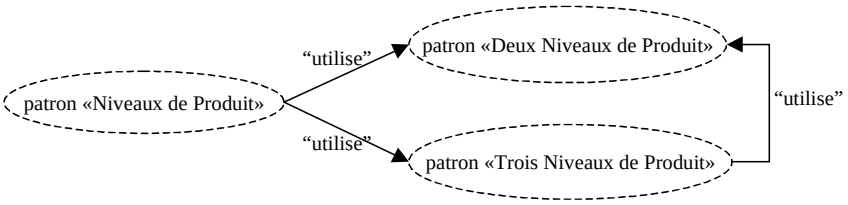
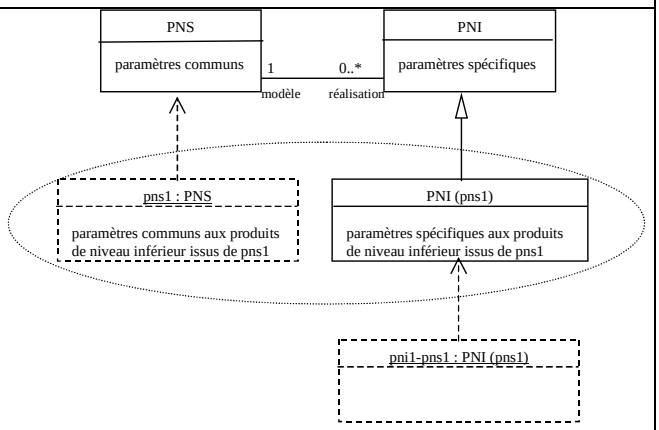


Figure 6.3- Liens d'utilisation dans le patron "Niveaux de Produit"

Dans ce qui suit, nous présentons brièvement chacun des deux patrons "utilisés" par le patron "Niveaux de Produit" c'est-à-dire les patrons "Deux Niveaux de Produit" et "Trois Niveaux de Produit". Nous mettrons en évidence le lien d'utilisation entre les patrons "Deux Niveaux de Produit" et "Trois Niveaux de Produit".

1.1.1.19. Patron "Deux Niveaux de Produit"

<table>
<tr><td>Nom</td><td>Deux Niveaux de Produit.</td></tr>
<tr><td>Problème</td><td>Ce patron permet de représenter le concept de produit en deux niveaux d'abstraction permettant d'une part de partitionner les connaissances relatives aux différents niveaux et d'autre part de définir des relations entre les niveaux afin de permettre la propagation de propriétés entre les niveaux. Ce patron peut être appliqué pour représenter aussi bien le cas des deux niveaux produit-générique & type-produit que le cas des deux niveaux type-produit & produit-physique.</td></tr>
<tr><td>Motivation</td><td>Nous considérons l'exemple suivant pour le produit "voiture". Nous illustrons le cas des deux niveaux produit-physique et type-produit.
<table>
<tr><th>Niveau de produit</th><td>Peugeot 206 (type)</td><td>Ma peugeot 206 (exemplaire)</td></tr>
<tr><td>caractéristiques</td><td>Nom TP = peugeot 206
Moteur = 4 cylindres
Couleur = vert ou bleu</td><td>Nom EP = peugeot 206
Moteur = 4 cylindres
Couleur = bleu
N° série = 206-98-100</td></tr>
</table></td></tr>
<tr><td>Force</td><td>Ce patron a les mêmes forces que le patron "Item-description" de Coad ou encore le patron "Materialization" de Dahchour. Il relie une classe de catégories abstraites A (modèles de voitures) à une classe d'objets plus concrets C (voitures individuelles). Sa sémantique est définie en terme des abstractions "est-un" (généralisation) et "est-de" et des correspondances classe/métaclasse.</td></tr>
<tr><td>Contexte</td><td>Ce patron n'exige aucun autre patron ou modèle pour être appliqué. Cependant, le concepteur doit déjà connaître le nombre de niveaux de produit. Ce nombre doit être de 2.</td></tr>
<tr><td>Solution-
Démarche</td><td>1. Une entreprise articule ses activités autour de deux niveaux d'abstraction du produit. Deux classes abstraites interviennent :
▪ Produit Niveau Supérieur "PNS", détenant les propriétés communes des produits de niveau inférieur dérivant de celui-ci et contenant une ensemble de valeurs possibles de certaines propriétés.
▪ Produit Niveau Inférieur "PNI", détenant les propriétés spécifiques de produits de niveau inférieur associés à un produit de niveau supérieur et contenant des valeurs fixes de certaines propriétés.
L'association entre ces deux classes a pour cadinalités : 1 du côté de la classe "PNS" et 0..* du côté de la classe PNI.
Deux scénarios peuvent exister : "PNS" est un produit-générique & "PNI" est un type-produit ou "PNS" est un type-produit & "PNI" est un produit-physique.
2. Quand l'entreprise décide de créer un nouveau "Produit Niveau Supérieur" (par exemple le</td></tr>
</table>

	nouveau type de produit *peugeot 206*) soit *pns1*. *pns1* est représenté par une instance de la classe "PNS". Ceci se traduit par l'objet *pns1:PNS*. 3. Avec la création du nouveau produit *pns1*, il y aura différents produits de niveau inférieur dérivant de *pns1* (par exemple différents exemplaires de *peugeot 206*). On crée alors une classe concrète pour tous les produits de niveau inférieur dérivant de *pns1*, soit la classe "PNI (pns1)". Cette classe contient les propriétés spécifiques aux produits de niveau inférieur dérivant de pns1.
Solution-Modèle	La création d'un nouveau PNS, soit pns1 (par exemple peugeot206), implique la création de divers produits de niveau inférieur issus de pns1 (divers exemplaires de peugeot 206).  Toute création d'un nouveau produit de niveau supérieur génère une instance de la classe "PNS" et une sous-classe de la classe "PNI".

Nous soulignons que la solution offerte par le patron "Deux Niveaux de Produit" a été définie en "utilisant" le patron de conception "Item-description" de P. Coad [Coad, 96].

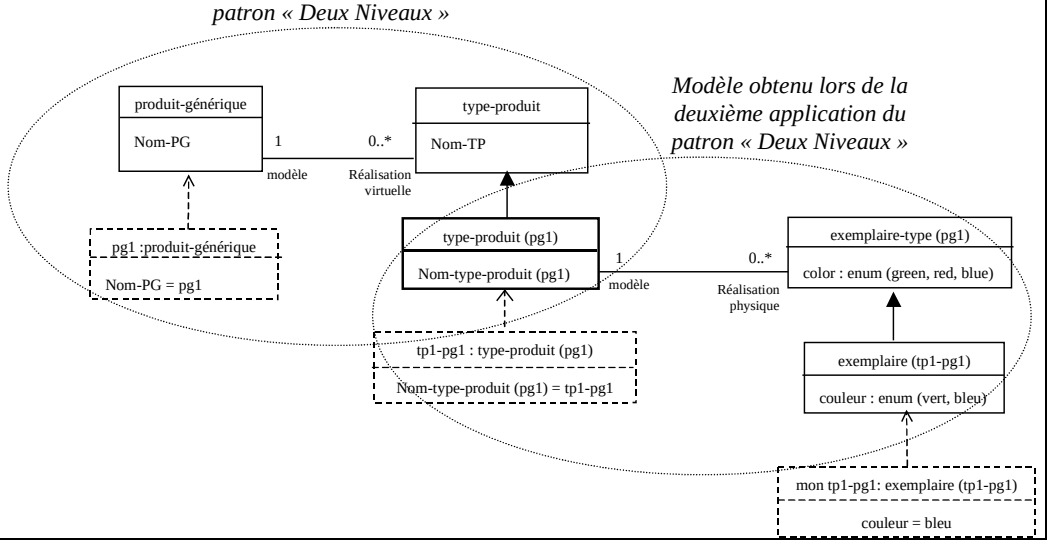
Dans ce qui suit, nous présentons le patron "Trois Niveaux de Produit".

1.1.1.20. Patron "Trois Niveaux de Produit"

Nom	Trois Niveaux de Produit.
Problème	Ce patron permet de représenter le concept de produit en trois niveaux d'abstraction (produit-générique, type-produit et produit-physique). Il permet d'une part de partitionner les connaissances relatives aux différents niveaux et d'autre part de définir des relations entre les niveaux afin de permettre la propagation de propriétés entre les niveaux.
Motivation	Nous considérons l'exemple suivant pour le produit "voiture". Nous distinguons trois niveaux de produit : produit-générique, type-produit et produit-physique.

Niveau de produit	Peugeot 206 (générique)	Peugeot 206-S16 (type)	Ma peugeot 206-S16 (exemplaire)
caractéristique	Nom-PG = peugeot 206 Moteur = 4 cylindres ou électrique Couleur = vert ou bleu ou rouge	Nom TP = peugeot 206-S16 Moteur = 4 cylindres Couleur = vert ou bleu	Nom EP = peugeot 206-S16 Moteur = 4 cylindres Couleur = bleu N° série = 206-98-100

Force	ce patron a les mêmes forces que le patron "Item-description" de Coad. Il relie une classe de catégories abstraites A (modèles de voitures) à une classe d'objets plus concrets C (voitures individuelles). Sa sémantique est définie en terme des abstractions "est-un" (généralisation) et "est-de" et des correspondances classe/métaclasse.
Contexte	Ce produit n'exige aucun autre patron ou modèle pour être appliqué. Cependant, le concepteur doit déjà connaître le nombre de niveaux de produit. Ce nombre doit être de 3.

Solution-Démarche	Une entreprise articule ses activités autour de trois niveaux d'abstraction du produit : ▪ le niveau "produit-générique", détenant les propriétés des produits génériques (tel que "nom-PG) ▪ le niveau "type-produit", détenant les propriétés des types de produits issus de divers produits génériques (tel que Nom-TP). ▪ Le niveau " produit-physique", détenant les propriétés des exemplaires de divers types de produits (tel que n° de série, date-fabrication) Le niveau "produit-générique" est plus haut que le niveau "type-prduit", lui-même plus haut que "produit-exemplaire". 1. Appliquer le patron "Deux Niveaux" avec le "produit-générique" comme classe "PNS" et le "type-produit" comme classe "PNI". - Une instance de la classe "PNS" est dans ce cas un nouveau porduit-générique (correspondant à la création d'une nouvelle ligne de produit), qu'on nomme *pg1* (par exemple le produit générique peugeot 206). - Avec la création du nouveau produit générique pg1, il y aura différents types de produits dérivant de *pg1* (par exemple différents types de *peugeot 206*). La classe concrète "type-produit (pg1)" représente l'ensemble de ces types de produits issus de *pg1*. 2. Appliquer encore une fois le patron "Deux Niveaux" avec le "type-produit (pg1)" comme classe "PNS" et les exemplaires de tous les types de produit issus de pg1 soit "exemplaire-type (pg1)" comme la classe "PNI". - Une instance de la classe PNS est dans ce cas un nouveau type de produit dans la ligne des produit pg1 (par exemple le type peugeot 206-S16 dans la ligne des produits peugeot 206), qu'on nomme *tp1-gp1*. - Avec la création du nouveau type de produit *tp1-pg1*, il y aura différents exemplaires de ce type de produit (par exemple différents exemplaires de *peugeot 206-S16*). La classe concrète "exemplaire (tp1-pg1)" représente l'ensemble de ces exemplaires de produits issus de *tp1-pg1*. 3. Assembler les deux modèles obtenus des deux phases précédente à l'aide de la classe commune entre ces deux modèle : la classe "type-produit(pg1)"
Modèle-Solution	Le modèle obtenu est le suivant : *Modèle obtenu lors de la première application du patron « Deux Niveaux »* *Modèle obtenu lors de la deuxième application du patron « Deux Niveaux »* 

Le patron "Trois Niveaux de Produit" utilise donc deux fois le patron "Deux Niveaux de Produit".
Soulignons que les patrons relatifs à la représentation des niveaux de produit ici proposés permettent de représenter plus de trois niveaux, si ce cas se présentait[147]. De la même façon que le patron "Trois Niveaux de Produit" a utilisé le patron "Deux niveaux de Produit", nous pourrions envisager d'utiliser l'un de ces deux patrons pour représenter quatre niveaux (en utilisant trois fois le patron "Deux Niveaux de Produit"), cinq niveaux (en utilisant deux fois le patron "Trois Niveaux de produit"), etc.

3.2.3. Bloc : Nomenclatures de Produit

L'objectif des patrons du bloc *Nomenclatures de produit* est de construire les fragments du modèle produit, correspondant aux diverses nomenclatures du produit (à ses différents niveaux).
Nous avons souligné à l'issue de l'analyse de domaine, la variance des types de nomenclatures gérées selon les niveaux de produit (nomenclature fonctionnelle, géométrique, organique d'étude, organique générique, etc.) et la variance de construction (de modélisation) de ces nomenclatures.

- La variance de types de nomenclatures dépend de l'entreprise considérée et des niveaux de produits gérés. En effet, et à titre d'exemple, selon qu'un produit générique est géré ou non, la nomenclature organique générique existe ou non. Par ailleurs, la typologie utilisée mais également le contenu des diverses nomenclatures varient d'une entreprise à une autre.
- La variance dans la construction des nomenclatures dépend des propriétés des divers types de nomenclatures, en termes de nature des éléments qui les constituent (les nœuds) et de la sémantique des liens de composition entre ces nœuds. Ainsi, dans certaines nomenclatures (tel que les nomenclatures organiques génériques), il existe des composants optionnels et aussi des composants qui sont des variantes d'autres composants. Par ailleurs, le lien de composition peut être partagé (cas de nomenclature organique générique), exclusif (cas de nomenclature organique physique), etc.

Si la variabilité autour des propriétés des nomenclatures est claire et exhaustive, la variabilité autour de la terminologie et le contenu des nomenclatures l'est moins et elle est peu maîtrisable. Nous nous référons donc aux propriétés des nomenclatures comme points de variabilité, à base desquels divers patrons de construction de nomenclature sont définis. Ce choix est aussi justifié par le fait que plusieurs types de nomenclatures ont les mêmes propriétés et donc la même construction. Il est alors judicieux de raisonner plutôt sur la variabilité des propriétés des nomenclatures que sur leur terminologie et leur contenu. Nous définissons donc un premier **patron "Construire Nomenclature"** qui a pour objectif d'*orienter vers divers patrons de construction de nomenclatures selon les propriétés de la nomenclature considérée*.
Toutefois et avant de construire les diverses nomenclatures, il est souhaitable de guider le concepteur du SIP dans la détermination des propriétés des nomenclatures dont il dispose, voire, le cas échéant, de guider le concepteur dans la détermination des nomenclatures gérées

[147] Dans certaines industries automobile, on parle en effet de : Tous-véhicules, Plate-forme, Véhicule, Type-véhicule, Exemplaire-véhicule.

dans l'entreprise considérée selon les niveaux de produit existants. Un patron préliminaire, appelé **patron "Nomenclatures Appliquées",** a donc pour objectif de *déterminer les nomenclatures pouvant être appliquées à chacun des niveaux de produit existants* et également de *déterminer les propriétés de chacune des nomenclatures identifiées*. Ce patron est proposé à titre informatif pour le concepteur du SIP c'est-à-dire qu'il peut l'utiliser sans toutefois appliquer intégralement la solution proposée. Il sert pour informer le concepteur des nomenclatures "pouvant être gérées" pour chaque niveau de produit sans l'obliger à considérer exactement les nomenclatures proposées (ni la terminologie utilisée). Il est également facultatif; le concepteur qui connaît préalablement les types de nomenclatures gérées et également les propriétés de ces nomenclatures peut se passer de ce patron et appliquer directement le patron "Construire Nomenclature".
Une fois les nomenclatures identifiées et construites, la dernière information relative au bloc *Nomenclatures de produit* consiste à associer chacune de ces nomenclatures aux niveaux de produit appropriés. C'est l'objectif d'une des activités du patron "Points de variabilité" (voir § 3.2.1). A ce propos, nous soulignons que le patron "Nomenclatures Appliquées" sert de support à cette activité puisqu'il précise pour chaque niveau de produit, les nomenclatures qui lui sont associées.

Nous venons ainsi d'illustrer dans ce paragraphe la façon de diviser un problème global (en l'occurrence la construction de fragments de modèle relatifs aux nomenclatures) en divers patrons et ce de façon à simplifier la résolution du problème global tout en tenant compte que chaque patron associé à chacun des sous-problèmes doit être le plus autonome possible vis à vis des autres, isolable et réutilisable seul.
Dans ce qui suit, nous décrivons brièvement le patron "Nomenclatures Appliquées" et nous développons le patron "Construire Nomenclature".

1.1.1.21. Patron "Nomenclatures Appliquées"

Nom	Nomenclatures Appliquées.
Problème	Déterminer les nomenclatures pouvant être gérées dans l'entreprise, selon les niveaux de produit présents et définir les propriétés de chacune des nomenclatures.
Solution-Démarche	▪ **Types de nomenclatures gérées** - si le niveau produit-générique existe, les nomenclatures qui peuvent lui être associées sont : nomenclature organique générique, nomenclature géométrique générique - si le niveau type-produit existe, les nomenclatures qui peuvent lui être associées sont : nomenclature organique type, nomenclature géométrique type - si le niveau produit-physique existe, les nomenclatures qui peuvent lui être associées sont : nomenclature organique physique. - la nomenclature fonctionnelle est souvent associée au niveau de produit le plus haut existant dans l'entreprise. ▪ **Attachement des Nomenclatures à chaque niveau de produit (en terme de classes)** 1. Si le niveau produit-générique existe : - au produit-générique sont attachées la nomenclature organique générique, la nomenclature géométrique générique et la nomenclature fonctionnelle - au type-produit, s'il existe, aucune nomenclature ne lui est rattachée (car ses nomenclatures sont déclinées à partir des nomenclatures du produit générique à l'aide d'opérations sur les classes associées aux niveaux de produit)

	- au produit-physique, s'il existe, est associée la nomenclature physique 2. si le niveau produit-générique n'existe pas et que le type-produit existe : - au type-produit sont attachées la nomenclature organique type, la nomenclature géométrique type et la nomenclature fonctionnelle - au produit-physique, s'il existe, est associée la nomenclature physique 3. si seul le niveau produit-physique existe, alors : - au produit-physique sont associés la nomenclature organique physique et la nomenclature fonctionnelle ▪ **Propriétés des nomenclatures** Un tableau, analogue au tableau présenté à l'Annexe B, associe à chaque type de nomenclature la nature des nœuds qui le constituent et la sémantique du lien de composition entre les nœuds.

Notons qu'il n'y a pas de solution-modèle dans ce patron. Il capitalise uniquement un fragment de démarche.

1.1.1.22. Patron "Construire Nomenclatures"

L'objectif de ce patron est de guider la construction du modèle des nomenclatures et ce selon les propriétés de la nomenclature considérée. Nous avons distingué deux types de propriétés :

- des propriétés relatives à la nature des composants constituant les nœuds de la nomenclature. Un composant peut être la variante d'un autre composant, il peut être optionnel ou obligatoire. Selon la nature des composants existants, diverses architectures[148] du modèle de nomenclature sont proposées. En fixant la variabilité autour de la nature des composants d'une nomenclature, l'ensemble des classes et des relations du modèle sont fixés.
- des propriétés relatives à la sémantique du lien de composition : un lien de composition peut être partagé, exclusif, dépendant, indépendant, etc. Selon la sémantique du lien, diverses contraintes peuvent être rattachées aux classes (la plupart des cas à la classe "composite" du lien de composition [Oussalah, 97b]). Ces contraintes se traduisent par des pré ou post conditions sur les opérations de classes[149] et par des invariants de classes[150]. Les pré et post conditions ainsi que les invariants peuvent être exprimés en langue naturelle ou par des langages plus formels (OCL en l'occurrence). En fixant la variabilité autour de la sémantique du lien de composition, la description des classes est complétée.

Ainsi, il est évident qu'il faut commencer par fixer la variabilité autour des propriétés relatives à la nature des composants de la nomenclature (pour obtenir les entités du modèle : les différentes classes et relations) puis fixer la variabilité autour des propriétés relatives à la

[148] Par architecture, nous désignons la structure du diagramme de classes en terme de classes et de relations entre les classes.

[149] Pour chaque opération, les droits et obligations de l'objet sont spécifiés par des pré conditions et post conditions. Une pré condition doit être vérifiée avant l'exécution de l'opération. Les obligations sont spécifiées par des post conditions. Celles ci doivent être vraies juste après la fin d'exécution de l'opération [Hassine, 00].

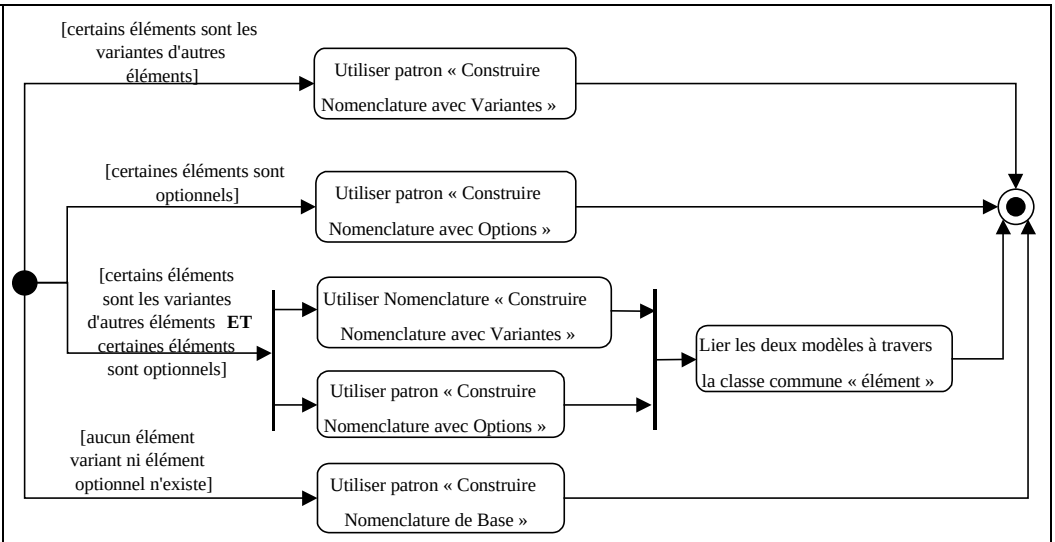
[150] Un invariant exprime une contrainte qui doit être vérifiée par toutes les instances d'une classe. Un invariant est décrit au moyen d'une expression qui doit être vraie chaque fois que l'invariant est rencontré. Les pré et post conditions doivent être vérifiées dans les intervalles de temps respectifs avant et après l'exécution d'une opération, alors qu'un invariant doit être vrai tout le temps [Hassine, 00].

sémantique du lien de composition (pour compléter le modèle en terme d'opérations de classes et de propriétés de liens éventuellement).
La solution offerte par le patron "Construire Nomenclature" consiste donc à faire appel à divers autres patrons qui assurent la construction du modèle de nomenclature selon la nature des composants de la nomenclature. Chacun de ces patrons distingue ensuite différentes constructions selon la sémantique du lien de composition pour définir entièrement la construction du modèle considéré.

Nom	Construire Nomenclature
Problème	Ce patron permet de construire une nomenclature selon différentes caractéristiques de celle-ci.
Motivation	Le produit, à ses divers niveaux peut être structuré par plusieurs nomenclatures, organiques (organique générique, organique spécifique d'étude, organique spécifique indisyrielle, organique physique, etc.), fonctionnelles, géométriques (géométrique générique, géométrique spécifique), etc. La construction de ces nomenclatures ne dépend pas de la nature des constituants de la nomenclature (article, fonction, entité géométrique), mais du rôle joué par les constituants. Certains constituants sont à variantes et donc un ensemble de variantes leur est associé, possédant chacun une composition spécifique. D'autres constituants sont optionnels dans une nomenclature. A titre d'exemple, la nomenclature organique générique ou la nomenclature géométrique générique sont définis à l'aide de constituants obligatoires, optionnels et à variantes. La nomenclature organique physique par contre n'est constituée que d'articles obligatoires.
Force	Ce patron distingue diverses constructions de la nomenclature selon la nature des nœuds constituant la nomenclature.
Contexte	Ce patron n'exige aucun autre patron ou modèle pour être appliqué[151].

Solution-Démarche	▪ Si certains éléments de la nomenclature peuvent être les variantes d'autres éléments, alors appliquer le *patron "Construire Nomenclature avec Variantes"*
	▪ Si certains éléments de la nomenclature peuvent être optionnels, alors appliquer le *patron "Construire Nomenclature avec Options"*
	▪ Si certains éléments de la nomenclature peuvent être les variantes d'autres éléments et que certains éléments peuvent être optionnels dans la nomenclature, alors :
	  ▪ Appliquer le *patron "Construire Nomenclature avec Variantes"*
	  ▪ Appliquer le *patron "Construire Nomenclature avec Options"*
	  ▪ Lier les deux modèles obtenues par application des deux patrons précédents à travers la classe communes aux deux modèles : "élément".
	▪ S'il n'existe ni éléments variantes ni éléments optionnels, alors appliquer le *patron "Construire Nomenclature de Base"*.

[151] Le concepteur n'est pas obligé par ailleurs d'avoir déjà appliquer le patron "Nomenclatures Appliquées".

	
Solution-Modèle	Le diagramme de classe obtenu représente un modèle d'une nomenclature particulière.
Utilise	{Construire Nomenclature avec Variantes, Construire Nomenclature avec Options, Construire Nomenclature de Base}

Le patron "Construire Nomenclature" fait donc appel à trois autres patrons:

- le patron "Nomenclature avec Variantes" pour construire des nomenclatures dont certains composants sont des variantes d'autres composants ;
- le patron "Nomenclature avec Options" pour construire des nomenclatures dont certains composants sont optionnels ;
- le patron "Nomenclature de Base" pour une construire une nomenclature "simple" qui correspond à une composition récursive de composants (telle que proposée dans le patron Composite d'E. Gamma).

Le modèle de nomenclature offert par ce dernier patron (qui est une composition récursive de composants) constitue la partie centrale de la nomenclature. Il se retrouve dans n'importe quel modèle de nomenclature, c'est-à-dire aussi bien dans une nomenclature simple avec unqiuement des composants obligatoires composés d'autres composants obligatoires que dans une nomenclature enrichie constituée en plus de variantes ou\et d'options. Ainsi, lors de la construction des nomenclatures enrichies dans les patrons "Nomenclature avec Variantes" et "Nomenclature avec Options", une partie de la solution consiste à construire le modèle de nomenclature simple, analogue à celui offert par le patron "Nomenclature de Base". Les patrons "Nomenclature avec Variantes" et "Nomenclature avec Options" utilisent donc le patron "Nomenclature de Base".

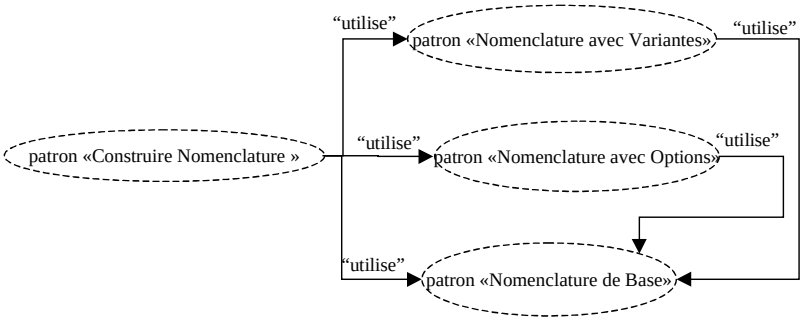
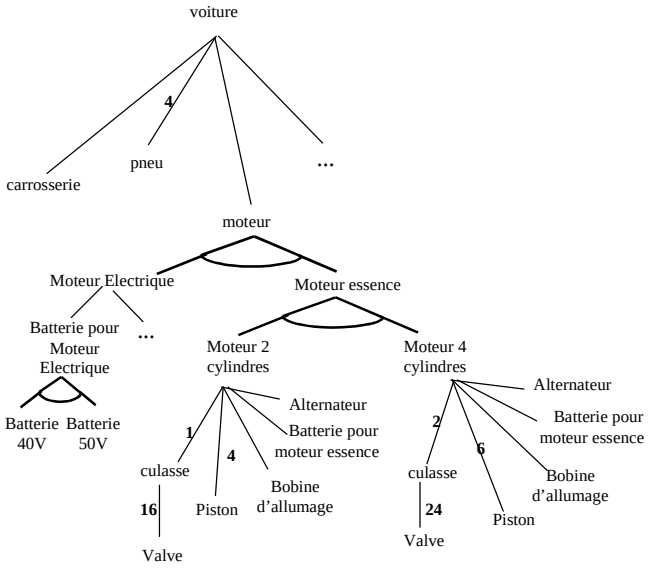


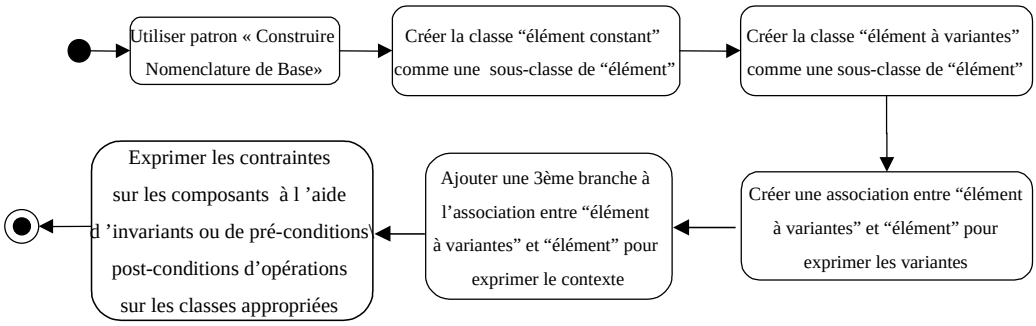
Figure 6.4 - Liens d'utilisation dans les patrons du bloc "Nomenclatures Produit"

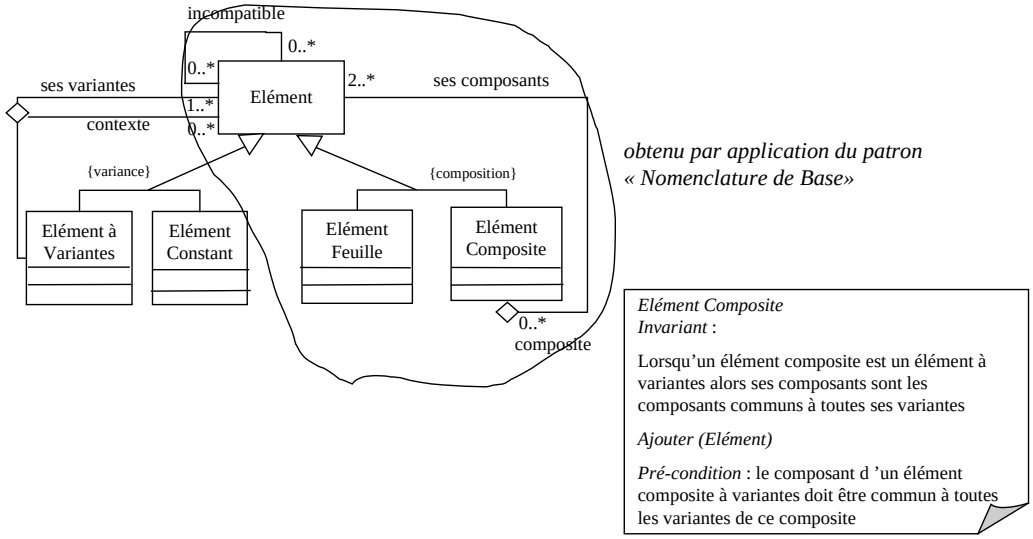
Notons enfin que c'est lors de la construction du modèle représentant une composition récursive de composants, que la variabilité autour de la sémantique de lien est fixée. Ainsi, divers patrons de construction de nomenclatures de base viennent "raffiner" le patron "Nomenclature de base", traitant chacun un problème particulier de construction de nomenclature, selon la sémantique du lien de composition.

Dans ce qui suit, nous présentons partiellement chacun des patrons introduits : "Nomenclature avec variantes"; "Nomenclature avec options" et "Nomenclature de base".

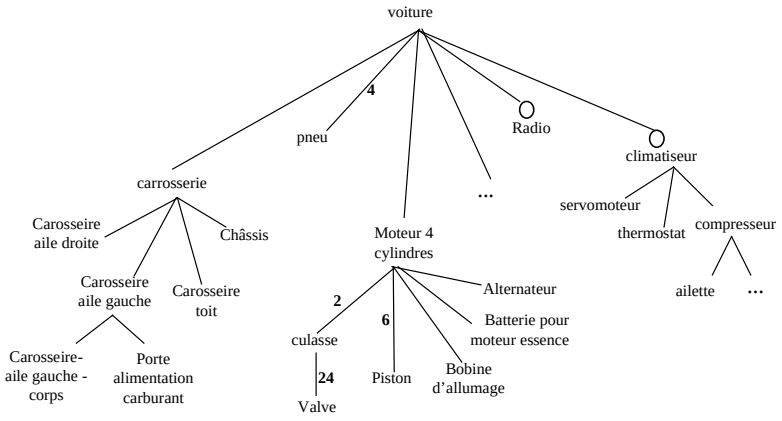
1.1.1.23. patron "Nomenclature avec Variantes"

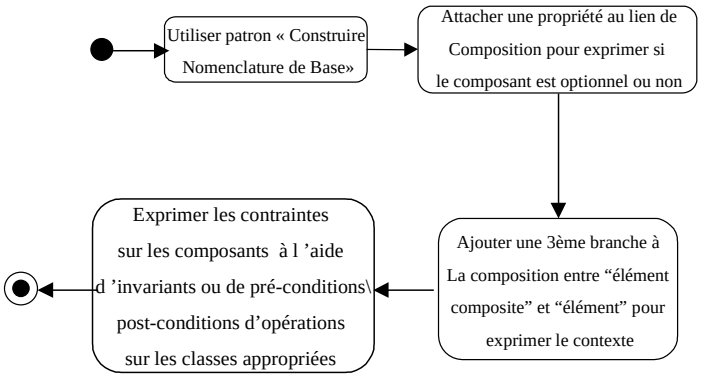
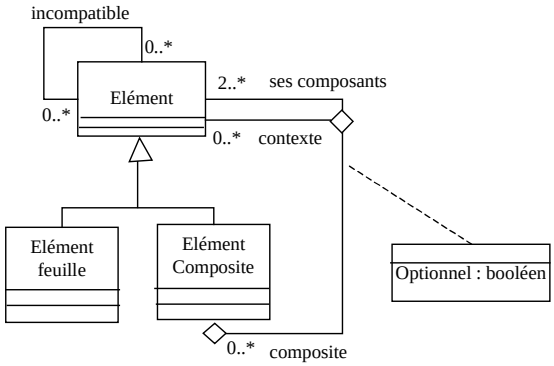
Nom	Nomenclature avec Variantes.
Problème	Ce patron permet de construire une nomenclature dont certains éléments sont les variantes d'autres éléments de la même nomenclature.
Motivation	Considérons le produit voiture. La composition du produit générique peut être en partie comme suit :  Le composant moteur est un articles à variantes. Une voiture possède soit un moteur électrique soit un moteur essence. Par ailleurs, divers types de moteur essence existent; celui-ci peut être à 2 cylindres ou à 4cylindres. Le composant moteur essence qui était une variante de moteur est elle même un article à variantes. Chacune des variantes a une composition spécifique. Ainsi, le moteur électrique est composé d'une batterie pour moteur électrique, etc. alors que les moteurs essence sont tous composés d'alternateur, de batterie pour moteur essence, de bobine d'allumage, de piston et de culasse. Selon que le moteur essence soit à 2 cylindres ou à 4 cylindres, le nombre de chacun des ces composants varie. Ainsi, dans une nomenclature à variantes, il peut exister une hiérarchie de variantes (une variante est elle même un article à variantes et possède ainsi des variantes). Chacune des variantes d'un composant donné possède sa propre composition. Toutefois, certains composants des variantes sont communs à l'ensemble des variantes (tel que le composant "alternateur" de moteur 4 cylindres et moteur 2 cylindres).
Force	Le modèle de nomenclature avec variantes permet de : - gérer une hiérarchie de variantes de façon à pouvoir éliminer automatiquement certaines variantes lorsque les variantes de niveau supérieur ne sont pas retenues.

	- exprimer le contexte dans lequel un élément est une variante d'un autre élément. Ceci permet d'associer à une variante donnée toutes les autres variantes qui lui sont compatibles. - exprimer la décomposition d'un élément à variantes (qui a différentes variantes) aussi bien au niveau de cet élément (on n'exprime alors que les composants communs à toutes ses variantes) qu'au niveau de ses variantes (on exprime les composants spécifiques à chaque variante)
Contexte	Ce patron n'exige aucun autre patron ou modèle pour être appliqué.
Solution-Démarche	1. Appliquer le patron " Nomenclature de Base". 2. Le modèle de nomenclature obtenu représente la classification des éléments selon le critère de décomposition (élément feuille ou composite). L'existence d'éléments variantes introduit une seconde classification selon leur variabilité. Ainsi, un élément peut être à variantes ou constant. Puisque un élément peut être feuille ou composite et en même temps peut être à variantes ou constant, alors la double classification devrait être complète et disjointe. Une autre branche relative à la classification selon la variabilité est alors introduite dans le modèle de nomenclature, comme suit: ▪ Créer la classe "élément constant" comme une sous-classe de "élément" ▪ Créer la classe "élément à variantes" comme une sous-classe de "élément" ▪ Puisque les variantes d'un "élément à variantes" sont d'autres "éléments" qui eux même peuvent être à variantes ou constants et feuilles ou composites, alors créer une association entre les classes "élément à variantes" et "élément" qui associe à chaque "élément à variantes" ses variantes. Cette association a comme rôle *ses variantes* du côté de la classe "élément". Les cardinalités de l'association sont de 1..* du côté de "élément" et 0..* du côté de "élément à variantes". ▪ Un élément est défini comme une variante d'un certain élément à variantes dans un contexte particulier (le composant "moteur diesel" est variante de "moteur" dans le produit "peugeot 206" mais il est un élément constant dans la composition d'un "groupe électrogène"). Ainsi, l'association précédemment créée entre "élément" et "élément à variantes" doit expliciter le contexte de la variance. Ce contexte est également un "élément" (le composite dans lequel il est composant). Ajouter une troisième branche dans l'association reliant encore une fois la classe "élément" et dont le rôle est *contexte*. La cardinalité de cette branche est de 0..*. ▪ Exprimer les contraintes éventuelles sur les composants à l'aide d'invariants ou de pré-conditions \post-conditions sur les opérations des classes appropriées. 
Solution-Modèle	Le modèle obtenu a une forme semblable à la suivante:

	
Utilise	{Nomenclature de Base}

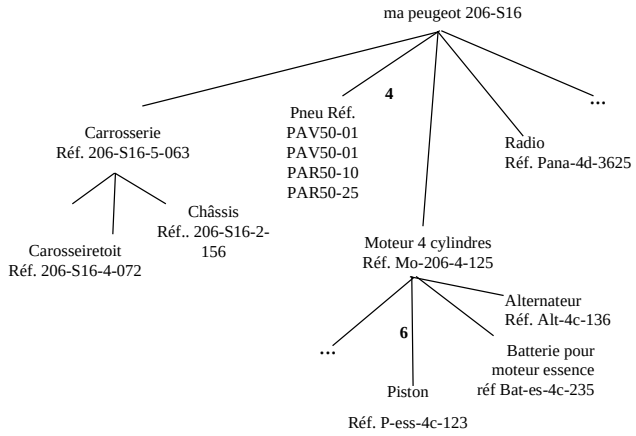
1.1.1.24. patron "Nomenclature avec Options"

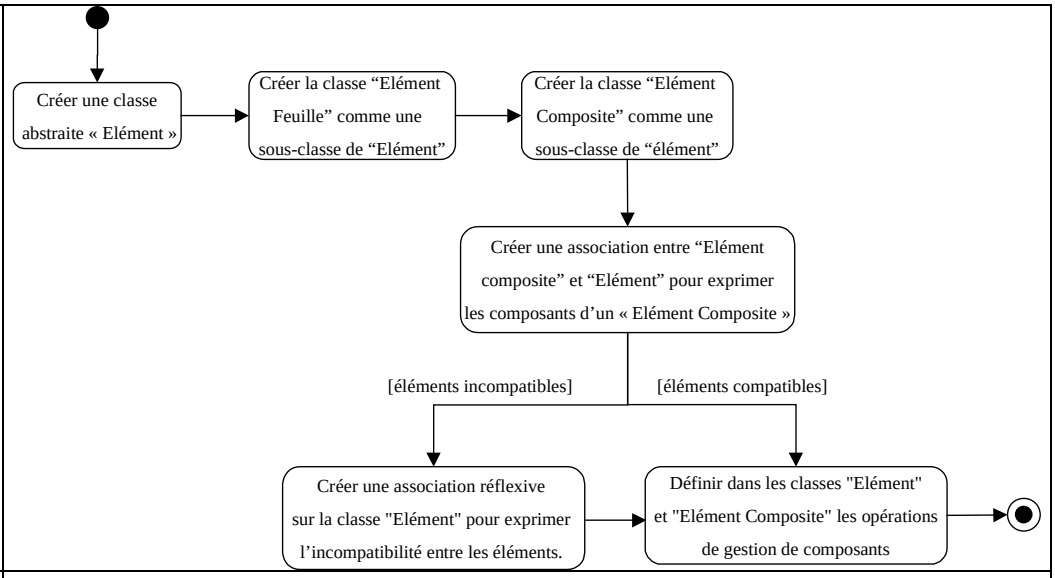
Nom	Nomenclature avec options.
Problème	Ce patron permet de construire une nomenclature dont certains éléments sont optionnels dans la nomenclature.
Motivation	Considérons le produit voiture. La composition du produit générique peut être en partie comme suit :  La voiture est composé nécessairement d'une carrosserie, d'un moteur, de pneus et d'un ensemble d'autres éléments nécessaires (volant, boite de vitesse, etc.). certains composants sont toutefois optionnels et sont mis en place dans la voiture selon le modèle de la voiture, selon les souhaits des clients, etc. Un élément optionnel peut être un composite et donc possède sa propre composition.
Force	Le modèle de nomenclature avec options permet de : - exprimer le contexte dans lequel un élément est optionnel dans une structure donnée. Ceci permet d'associer à une option donnée toutes les autres options qui lui sont compatibles ainsi que les variantes qui lui sont compatibles, en cas de produit à options et à variantes. - exprimer qu'un composant d'un élément composite est optionnel ou obligatoire.
Contexte	Ce patron n'exige aucun autre patron ou modèle pour être appliqué.
Solution-	1. Appliquer le patron "Nomenclature de Base".

Démarche	2. Le modèle de nomenclature obtenu représente la composition récursive d'éléments (élément feuille ou composite). L'existence d'éléments optionnels introduit des propriétés supplémentaires sur le lien de composition entre un élément composite et ses composants. Attacher alors une propriété au lien pour exprimer si le composant est optionnel ou non. 3. Un élément est défini comme optionnel dans uns structure dans un contexte particulier (le composant "climatiseur" est optionnel dans le produit "peugeot 206" mais il est un élément obligatoire dans la composition d'un "véhicule frigorifique"). Ainsi, le lien de composition entre "élément composite" et "élément" doit expliciter le contexte de l'option. Ce contexte est également un "élément" (le composite dans lequel il est composant). Ajouter alors une troisième branche dans le lien de composition reliant encore une fois la classe "élément composite" à "élément" et dont le rôle est *contexte*. La cardinalité de cette branche est de 0..*. 4. Exprimer les contraintes éventuelles sur les composants à l'aide d'invariants ou de pré-conditions \post-conditions sur les opérations des classes appropriées. 
Solution-Modèle	Le modèle obtenu a une forme semblable à la suivante : 
Utilise	{Nomenclature de Base}

1.1.1.25. patron "Nomenclature de Base"

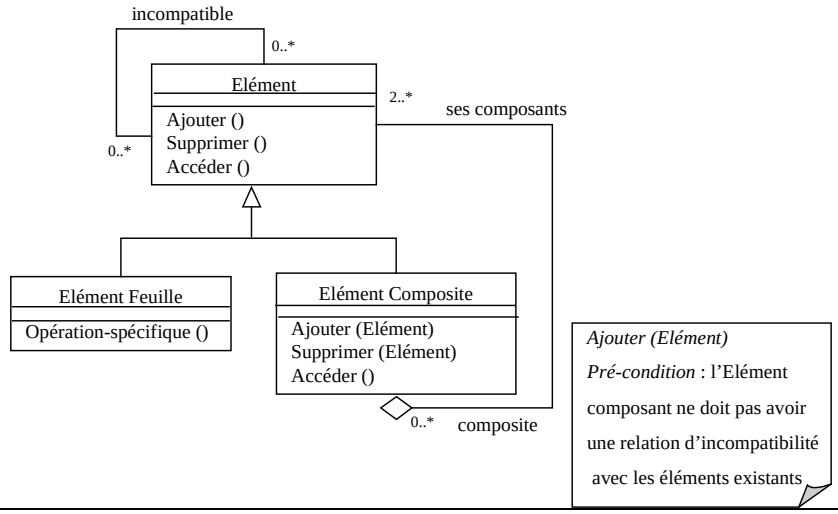
Nom	Nomenclature de Base.
Problème	Ce patron permet de construire une composition récursive d'éléments.
Motivation	Considérons un exemplaire individualisé d'une voiture, par exemple ma peugeot 206-S16. La composition de cet exemplaire est en partie comme suit :

	 L'exemplaire de voiture étant un choix définitif de composants, déjà mis en œuvre sur les lignes d'assemblage. Tous les composants sont obligatoires.
Force	Le modèle de nomenclature de base permet de définir des hiérarchies d'éléments simples et d'éléments composites et facilite l'ajout de nouveaux composants. Il a les mêmes forces que le patron de conception "Composite" d'E. Gamma.
Contexte	Ce patron n'exige aucun autre patron ou modèle pour être appliqué.
Solution-Démarche	1. Créer une classe abstraite "Elément". Cette classe définit l'interface commune des éléments composites et des éléments feuilles. 2. Créer une sous-classe de "Elément" qui s'appelle "Elément Feuille". Cette sous-classe regroupe tous les éléments qui ne sont pas décomposables. 3. Créer une sous-classe de "Elément" qui s'appelle "Elément Composite". Cette classe regroupe les éléments qui sont décomposables en d'autres éléments de même nature. 4. Créer une relation de composition entre les classes "Elément Composite" et "Elément". Cette composition associe à chaque "Elément Composite" ses composants dans la classe "Elément". Cette composition a pour cardinalité 2..*du côté de la classe "Elément" et 0..* du côté de la classe "Elément Composite". 5. Dans le cas où certains éléments ne sont pas compatibles entre eux, créer une association réflexive sur la classe "Elément" qui associe à chaque "Elément" les autres "Eléments" qui ne lui sont pas compatibles. Cette association est optionnelle. 6. Définir dans les classes "Elément" et "Elément Composite" les opérations de gestion de composants (accéder, supprimer, ajouter un composant). Dans le cas où certains éléments ne sont pas compatibles entre eux, ajouter une pré-condition à l'opération *Ajouter (Elément)* de la classe "Elément Composite".



Solution-Modèle

Le modèle obtenu est le suivant :



Le patron "Nomenclature de Base" permet de construire une composition récursive d'éléments, quelque soit la nature de l'élément considéré (article, fonction, feature, etc.). La solution offerte par ce patron a été définie en adaptant le patron Composite d'E. Gamma [Gamma, 95]. Soulignons par ailleurs l'intérêt de la rubrique solution-démarche qui permet d'indiquer les cas particuliers de construction de la solution-modèle. Dans la solution-modèle du patron "Nomenclature de Base", l'association réflexive sur la classe "Elément" ainsi que la pré-condition sur l'opération *Ajouter (Elément)* de la classe "Elément Composite" sont optionnelles (valables uniquement lorsque certains éléments sont incompatibles entre eux). C'est la solution-démarche qui permet d'exprimer ces cas particuliers.

Le patron "Nomenclature de Base" ne traite pas les différentes sémantiques que peut présenter le lien de composition. Il permet de résoudre le problème en général. Pour tenir compte des diverses sémantiques du lien de composition, plusieurs patrons doivent être définis pour traiter chacun des cas particuliers de la sémantique du lien de composition (lien partagé, lien exclusif, lien dépendant, lien pré-dominant, etc.). Ces patrons constituent donc des "raffinements" du patron "Nomenclature de Base"; le problème traité par chacun de ces patrons est une spécialisation (un cas particulier) du problème traité par le patron

"Nomenclature de Base". Ainsi, nous pouvons définir autant de patrons de raffinement que de sémantiques existantes du lien de composition (cf. Figure 6.5).

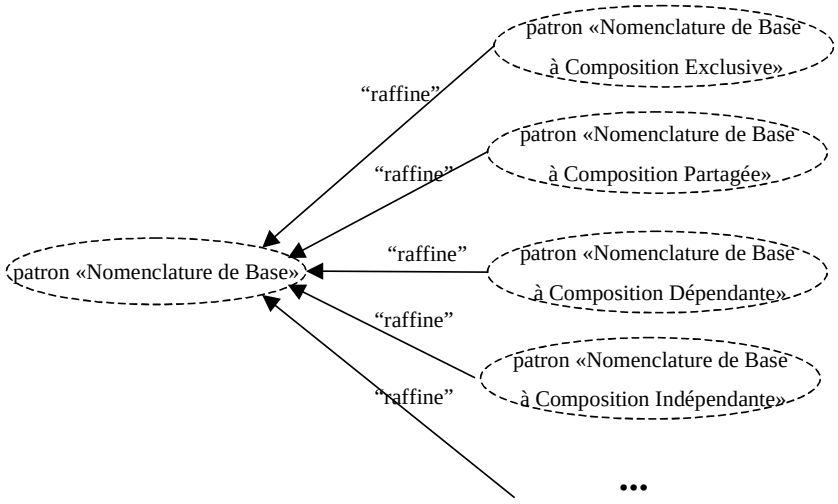
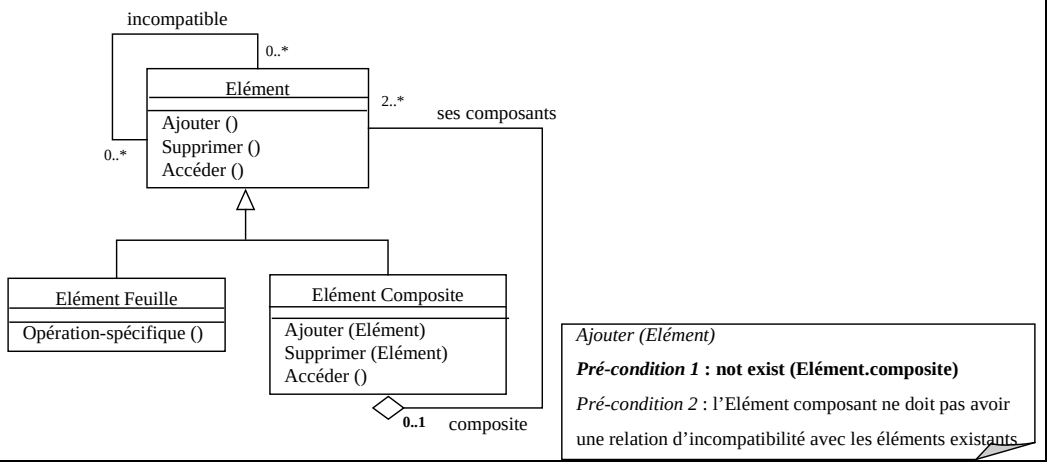


Figure 6.5 - Liens de raffinement entre les patrons de construction de nomenclatures de base

Pour d'illustrer nos propos, nous décrivons un des patrons de raffinement; ayant pour objectif de construire une nomenclature de base avec un lien de composition exclusif.

Nom	Nomenclature de Base avec lien de composition exclusive.
Problème	Ce patron permet de construire une composition récursive et exclusive d'éléments.
Motivation	Dans la nomenclature d'un exemplaire particulier d'une voiture (*ma peugeot 206-S16* par exemple), le *piston réf. P-ess-4c-123* ne peut pas appartenir en même temps au moteur d'un autre exemplaire de voiture (sa peugeot 206-S16).
Force	Le modèle de nomenclature permet de définir des hiérarchies d'éléments simples et d'éléments composites liés entre eux par des liens de composition exclusives.
Contexte	Ce patron n'exige aucun autre patron ou modèle pour être appliqué.
Solution-Démarche	1. Construire une composition récursive d'éléments. Pour cela, appliquer le patron "Nomenclature de Base". 2. Restreindre la cardinalité de 0..* à 0..1 pour le rôle *composite* du lien de composition. 3. Ajouter à l'opération *Ajouter (Elément)* de la classe "composite", les informations suivantes: - pré-condition : not exist (Elément.composite) - post-condition : …
Solution-Modèle	

3.2.4. Bloc : Document

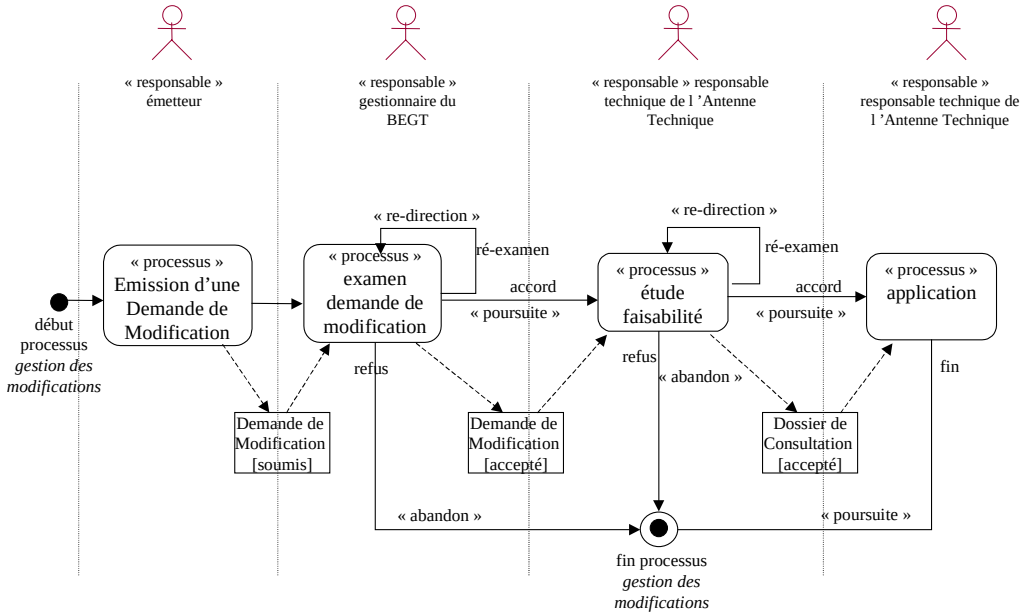
Les patrons du bloc *Document* servent à déterminer les divers documents et les dossiers documentant le produit (à ses divers niveaux) et à les associer au niveau de produit approprié dans le modèle produit. Le patron "Documents Appliqués" concourt à la réalisation du premier objectif. Il joue le même rôle dans le bloc *Document* que celui joué par le patron "Nomenclatures Appliquées" dans le bloc *Nomenclatures*. Il permet d'informer le concepteur du SIP sur les types documents ou les dossiers pouvant être associés à chacun des niveaux de produit gérés dans l'entreprise. Il permet par ailleurs de caractériser chacun des documents identifiés selon qu'il soient des documents dépendants ou non-dépendants. Ce patron facilite l'activité "Associer Documents au Produit" dans le patron "Points de Variabilité" puisqu'il identifie pour chaque document, l'objet documenté qui lui associé et caractérise également le lien entre le document et l'objet documenté (lien de type "décrit" ou "documente"). Le patron "Documents Appliqués" est présenté en Annexe D (§1.11).

La description des patrons d'analyse produit est ainsi terminée. Nous présentons dans ce qui suit les patrons d'analyse processus.

3.3. Patrons d'Analyse Processus

Les patrons processus ont pour objectif de traiter des problèmes de représentation des processus SIP.
Nous avons souligné lors de l'analyse de domaine que l'organisation des processus SIP (processus de définition de produit, processus de gestion de configuration, processus de modification produit, etc.) varie entre les différentes entreprises, en terme d'étapes qui composent un processus, de structure des ressources employées, d'objets du domaine mis en jeu, etc.
Contrairement aux connaissances produit où la variabilité autour des divers concepts est fixée, la variabilité sur l'organisation des processus ne peut être explicitée. Il ne s'agit pas donc ici de dresser les divers points de variabilité qui impliquent diverses constructions du modèle d'un processus donné (définition de produit, modification de produit, etc.). L'objectif est de fournir au concepteur du SIP une démarche générique qui lui permet de représenter n'importe quel processus SIP. Cette démarche doit aider le concepteur à décomposer un processus en une succession d'activités, à n niveaux où n est le nombre de niveaux de décomposition désiré par le concepteur. Cette démarche doit permettre également de renseigner le processus en terme de ressources impliquées, d'entrants\sortants mis en jeu, etc.
Lorsqu'on observe la manière avec laquelle les processus sont décomposés en activités dans les entreprises (décomposition à un niveau), nous remarquons la diversité des critères de décomposition. Cela peut être selon le changement de site d'entreprise dans lequel les activités sont assurées, selon le changement d'objectif des diverses activités composants le processus, selon le changement des acteurs impliqués, etc.
Nous soulignons par ailleurs qu'à chaque niveau de décomposition, le critère de décomposition peut varier (car on ne peut plus continuer avec le même critère, utilisé dans les niveaux de décomposition précédents). Toutefois, après n niveaux de décomposition et indépendamment de l'ordre des n critères considérés, la décomposition obtenue est souvent la même.

Une variabilité existe donc autour des critères de décomposition. Toutefois, même si ces critères sont différents, la manière d'aborder un problème de décomposition d'un processus, selon un critère donné (n'importe lequel) reste la même. C'est cette manière que nous capitalisons dans le patron "Décomposer un Processus" proposé.
Le patron "Décomposer Un Processus"[152] permet donc de décomposer un processus à n niveaux de décomposition, selon différents critères. Nous préconisons deux critères mais le patron permet de considérer d'autres critères de décomposition, spécifiques aux entreprises.

Nom	Décomposer un Processus.
Problème	Décomposer un processus global en sous-processus afin : - De représenter la répartition des tâches entre les acteurs concernés, - De représenter les points de décision et de synchronisation, - Distinguer les activités manuelles des activités informatisées.
Motivation	Prenons l'exemple du processus de ***Gestion des Modifications de produits*** chez notre partenaire industriel (Schneider Electric). Un tel processus de gestion de modification de produit peut être découpé en activités de manière à le rendre modulaire, à mieux cerner les actions nécessaires pour le mener et, ainsi, à mieux définir les acteurs responsables de chacune de ces actions. Nous préconisons trois critères de décomposition : le changement d'objectif, le changement d'acteur et le changement de type d'activité. Ainsi, nous pouvons décomposer le processus de gestion des modifications de la manière suivante : 1. Dans un premier temps, selon *l'objectif* en décomposant l'objectif global qui est « gérer processus de modification ». Le processus *Gestion des Modifications* est alors décomposé en les activités suivantes : - *Emission Demande de Modification* :. - *Examen demande :* le processus est initialisé par une émission d'une demande de modification qui est soumise à un gestionnaire du BEGT (Bureau d'Etudes Gestion Technique) pour examiner la pertinence de la demande et décide de la continuité du processus ou pas. - *Etude faisabilité :* si la demande est acceptée lors de l'étape précédente, elle est soumise aux responsables des Antennes Techniques (AT) des usines de production, pour une étude de faisabilité portant sur différents critères (techniques, financiers, …). - *Application :* si la demande est jugée faisable, elle est mise en application par les membres de l'AT.  *Figure a : Première décomposition du processus de gestion de modifications de produits*

[152] Nous soulignons que le patron "Décomposer un Processus" est issu d'une réflexion commune au niveau du projet POSEIDON, menée notamment avec le laboratoire DIAM de l'IUSPIM (Université d'Aix-Marseille III).

Remarque : Pour ne pas surcharger le modèle, nous n'avons illustré que quelques entrants/sortants dans la figure a.. Il convient de ne faire apparaître les entrants/sortants qu'à un niveau plus bas de décomposition, au niveau des opérations qui les transforment effectivement. Dans la suite de cet exemple, nous ne faisons pas figurer les entrants/sortants.

2. Les activités ainsi obtenues peuvent, lorsqu'elles sont de type 'processus', être découpées à leur tour en d'autres activités par affinement de leurs objectifs. Ainsi, le sous-processus "Etude faisabilité", dont l'objectif est d'étudier la faisabilité de la modification, peut être découpé comme suit :

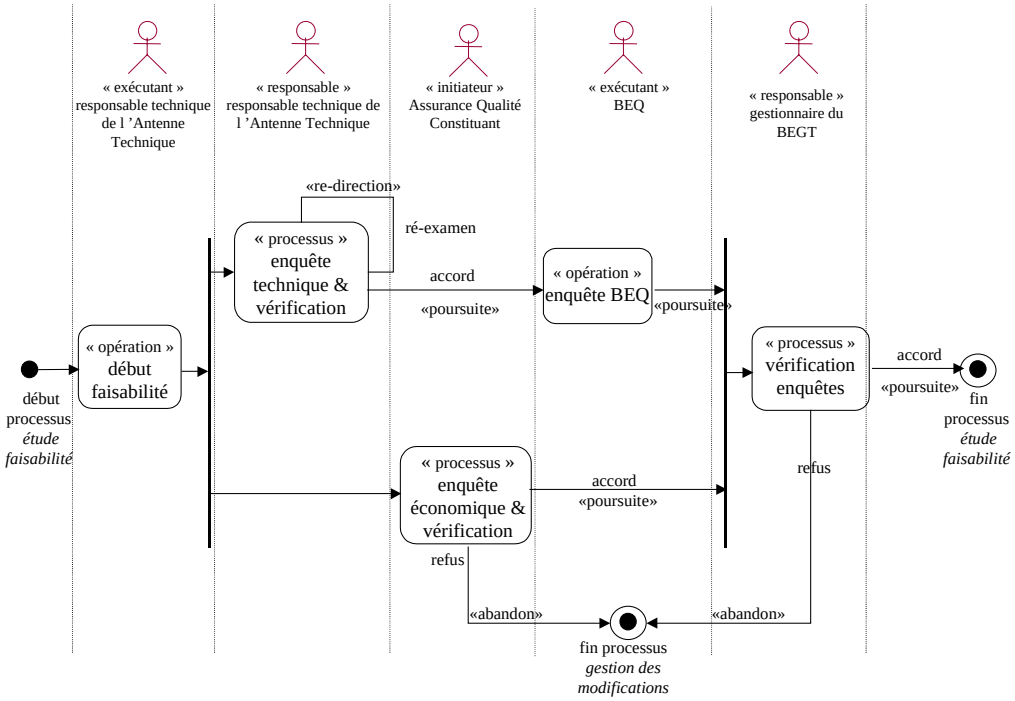


Figure b : Décomposition du sous-processus "Etude faisabilité" de Schneider

Remarque : pour une meilleure illustration, le lien de « re-direction » rebouclant sur certaines activités dans la figure a (telles que le processus *étude faisabilité*) est représenté pour exprimer aussi bien :

- un rebouclage sur l'ensemble du processus : un lien externe qui part de la dernière activité du processus et pointe sur sa première activité
- qu'un rebouclage interne à la décomposition du processus : un lien interne de « re-direction » qui part d'une activité quelconque de niveau inférieur c'est à dire composant le processus et qui y pointe.

Ainsi, en décomposant le processus *Etude Faisabilité*, le lien de « re-direction » figurant sur cette activité dans la figure a ne pointe pas obligatoirement dans la figure b sur la première activité de ce processus et il ne part pas non plus de sa dernière activité. Nous permettons qu'il parte d'une activité interne à ce processus et y pointe (processus *enquête technique & vérification*).

3. La décomposition des sous-processus peut être poursuivie en gardant le même critère de décomposition, en l'occurrence l'objectif, ou en changeant de critère de décomposition, par exemple le changement d'*acteurs* ou de rôles d'acteurs. Ainsi, pour le processus « enquête technique & vérification » nous obtenons :

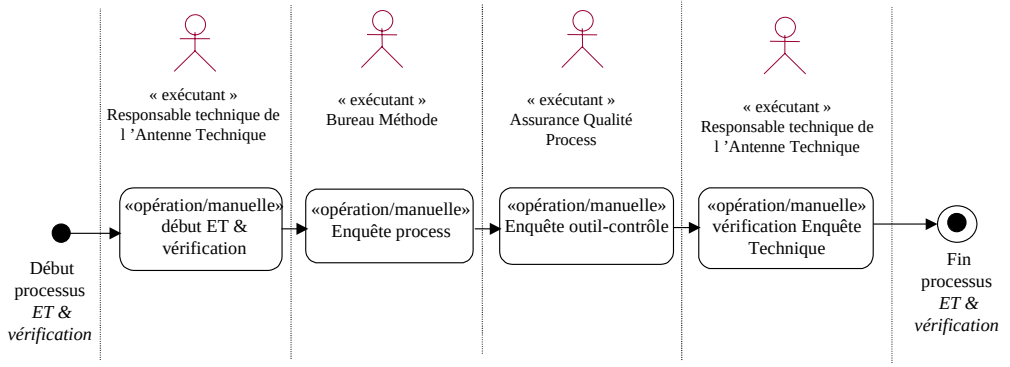


Figure c : Décomposition de l'activité "enquête technique & vérification" de Schneider

Force

Ce patron permet de décomposer un processus organisationnel selon une démarche générique, basée sur un ensemble de critères de décomposition indépendants de la spécificité des entreprises. Il permet en même temps de considérer dans la démarche proposée d'autres critères de décomposition, spécifiques à l'entreprise.

Contexte

Ce patron n'exige aucun autre patron ou modèle pour être appliqué.

Solution-Démarche

1. **Choisir un critère de décomposition.** Nous en préconisons trois :

- *Le changement d'objectif :* l'objectif d'un processus peut être trop général et doit donc être réifié afin de permettre sa décomposition.
- *Le changement d'acteur ou de rôle d'acteur :* c'est un bon indicateur quant au passage d'une activité à une autre, surtout lorsque c'est le responsable qui change.
- *Le changement de type d'activité*[153] *:* si une activité ne peut être affectée d'un unique type (informatisée ou manuelle), ceci est le signe que l'activité peut être décomposée en autant d'activités que de types auxquels elle aurait pu être associée.

2. **Décomposer le processus** en fonction du critère choisi.

- Critères suggérés:

- *Changement d'objectif* : quand l'objectif du processus est assez général, il est nécessaire de l'affiner en le décomposant. Les sous-objectifs ainsi obtenus sont associés à diverses activités qui correspondent à la décomposition du processus initial. Nous soulignons que la présence d'un objectif dans une activité donnée signifie implicitement l'occurrence d'une action de *mesure* de satisfaction de l'objectif à la fin de l'activité et donc un point de décision[154]. La décomposition de processus selon les objectifs implique donc une décomposition selon les points de décision.
- *Changement d'acteur* : un processus est assuré par des acteurs selon différents rôles. Une deuxième façon de décomposer un processus sera par changement d'acteurs. Par changement d'acteur, nous entendons aussi bien le changement d'un acteur par un autre que le changement du rôle d'un même acteur.
- *Changement de type d'activité* : à un niveau avancé de la décomposition d'un processus, les activités doivent être typées selon qu'elles soient manuelles ou informatisées. Une troisième façon de décomposer un processus sera selon le type des activités. Si une activité du processus ne peut être caractérisée de façon complète (i.e. on ne peut lui affecter un unique type : informatisée ou manuelle), il convient alors de décomposer cette activité en autant d'activités que de types auxquels elle aurait pu être associée.

- Règle de décomposition : en décomposant un processus,
- les liens qui y rentraient dans le diagramme d'activités de niveau supérieur pointeront sur la première activité de ce processus,
- les liens qui en sortaient dans le diagramme d'activités de niveau supérieur partiront de la dernière activité de ce processus,
- les liens de « re-direction » qui rebouclaient sur le processus dans le diagramme d'activités de niveau supérieur pointeront sur chacune des activités de ce processus ayant des points de

[153] Nous préconisons ce critère à un niveau avancé de la décomposition du processus.

[154] Un point de décision correspond au choix entre plusieurs transitions disjointes lors de la fin d'une activité donnée.

	décision et donnant lieu à des re-directions. 3. **Renseigner les activités** ainsi obtenues. Pour chacune d'elles : - nommer l'activité, - indiquer la nature de l'activité : opération ou processus, - lui affecter, si possible, un initiateur, un responsable et un exécutant (ça peut être un acteur ou un groupe d'acteurs ou une ressource matérielle), - lui affecter, si possible, un type : informatisée ou manuelle, - lui affecter, lorsqu'il s'agit d'une opération, les objets entrants et les objets sortants quand ils existent. Ces objets sont des objets métiers du Système d'Information Organisationnel, parmi ceux qui sont identifiés dans le modèle d'analyse produit obtenu par application du patron d'analyse produit "Points de Variabilité", - déterminer le type de succession : *ET* / *OU*, - typer les transitions entre activités : *poursuite*, *re-direction*, *abandon* (quand il s'agit de transitions de type OU), - exprimer les conditions de succession sur les transitions (accord, refus, réexamen, …). 4. **Etablir le diagramme d'activités** correspondant au découpage obtenu. Pour cela *appliquer le patron "Représenter un Processus"*. 5. **Repérer les activités non terminales** : une activité terminale correspond à une opération, donc ne pouvant être décomposée. Les activités non terminales possèdent au moins une des caractéristiques suivantes : - son objectif est trop général et peut être décomposé, - il n'a pas été possible de lui affecter *un* exécutant. Ceci est le signe que cette activité doit être décomposée en autant d'activités que d'exécutants qui auraient pu lui être associés. - il n'a pas été possible de lui affecter un unique type (informatisée ou manuelle). Ceci est le signe que cette activité doit être décomposée en autant d'activités que de types qui auraient pu lui être associés.
Solution-Modèle	Le modèle obtenu est un diagramme d'activités représentant la décomposition du processus en activités et illustrant les diverses ressources qui l'assurent ainsi que les entrants/sortants de chaque activité.

Nous soulignons que ce patron permet de décomposer un processus à n niveaux en appliquant n fois les 5 étapes de la solution-démarche. Le concepteur a la liberté de choisir le nombre n de niveaux de décompositions.
Ce patron permet donc de représenter les différentes étapes du processus (selon le niveau de détail désiré par le concepteur SIP) et d'allouer à chacune des étapes, les ressources nécessaires pour l'exécuter ainsi que les objets du domaine mis en jeu. Les diagrammes d'activités ainsi obtenus servent à spécifier le workflow associé au processus dans le SGDT considéré qui, rappelons le, propose des composants adaptés à la représentation des processus en terme d'une succession d'activités avec les divers points de décision associés et les diverses ressources allouées.
Ce patron suggère d'utiliser le patron "Représenter un Processus" pour établir le diagramme d'activités correspondant à la décomposition obtenue. Ce patron permet au concepteur SIP, non initié à la modélisation UML, de construire un diagramme d'activité représentant le découpage obtenu du processus étudié. Il est donc facultatif. Ce patron est également décrit dans l'Annexe D (§2.2).

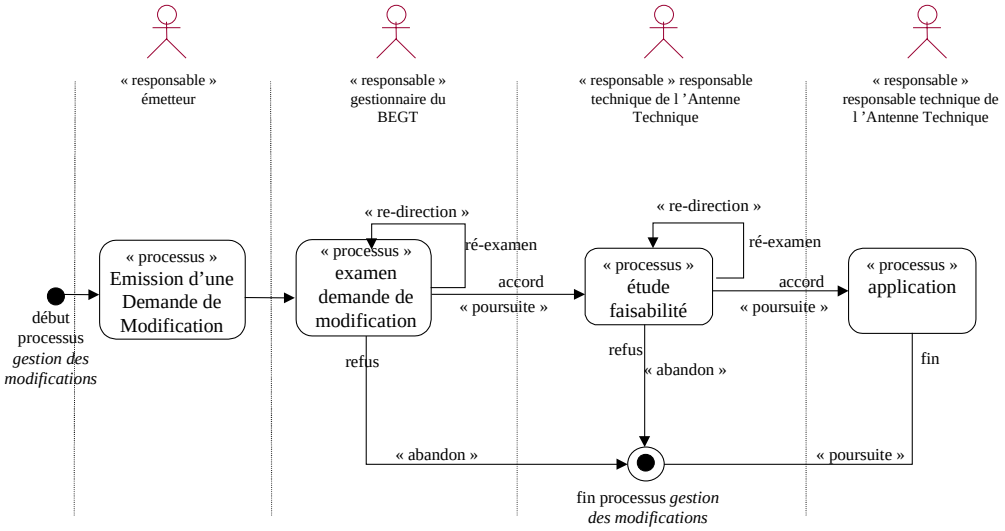
Nous finissons ainsi la description des patrons d'analyse du SIP. Nous présentons dans la suite les patrons de conception du SIP.

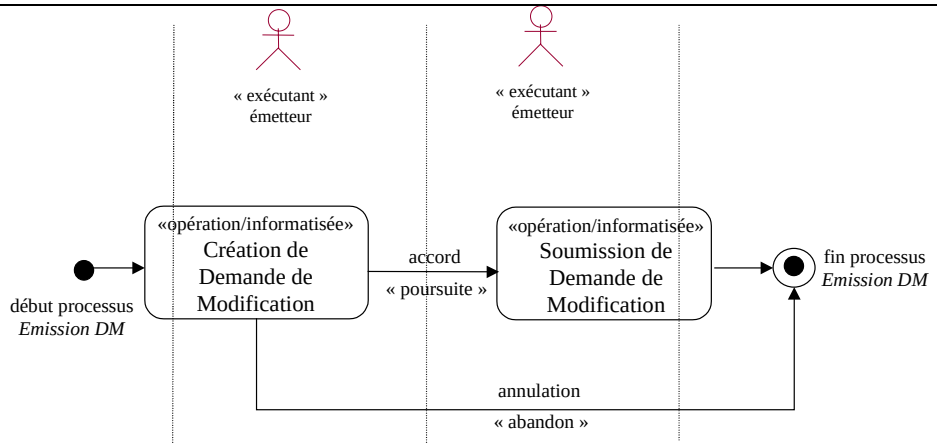
3.4. Patrons de Conception

Les patrons d'analyse produit et processus permettent de représenter l'architecture du SIP en décrivant les objets du Système d'Information Organisationnel (SIO) en terme de diagramme de classes et les processus SIP en terme de diagrammes d'activités (représentant le workflow associé). L'objectif des patrons de conception est alors de décrire les objets du Système d'Information Informatisé (SII), afin d'aboutir à un modèle de conception. Ces patrons permettent en conséquence de représenter et de supporter la dynamique des concepts produit engendrée par l'exécution des processus SIP.

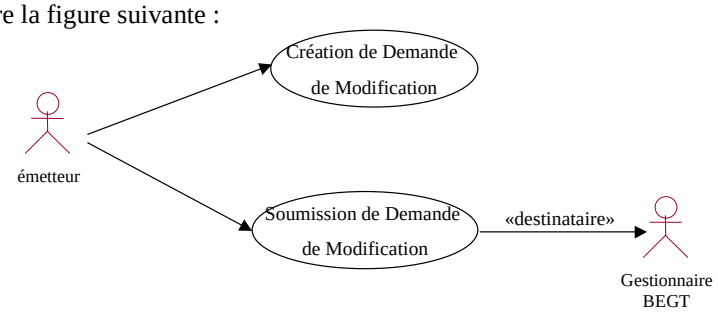
3.4.1. Point d'entrée : "Modèle de Conception SIP"

Dans les patrons d'analyse processus, une démarche est proposée pour décomposer les processus SIP. A un niveau avancé de décomposition, nous obtenons les opérations informatisées du processus (et qui seront assurées par un SGDT). Ces opérations constituent les traitements informatiques, càd les processus (informatiques) du SII associé au SIO. Ce sont les cas d'utilisation (les fonctionnalités) du SII.
La démarche proposée pour décrire les objets du SII consiste à partir de la modélisation des traitements associés aux processus métiers du SIP (les opérations informatisées de ces derniers) et à déterminer les collaborations d'objets mises en jeu pour réaliser ces traitements. Le patron "Modèle de Conception SIP" a pour objectif de guider la construction du modèle de conception du SIP.

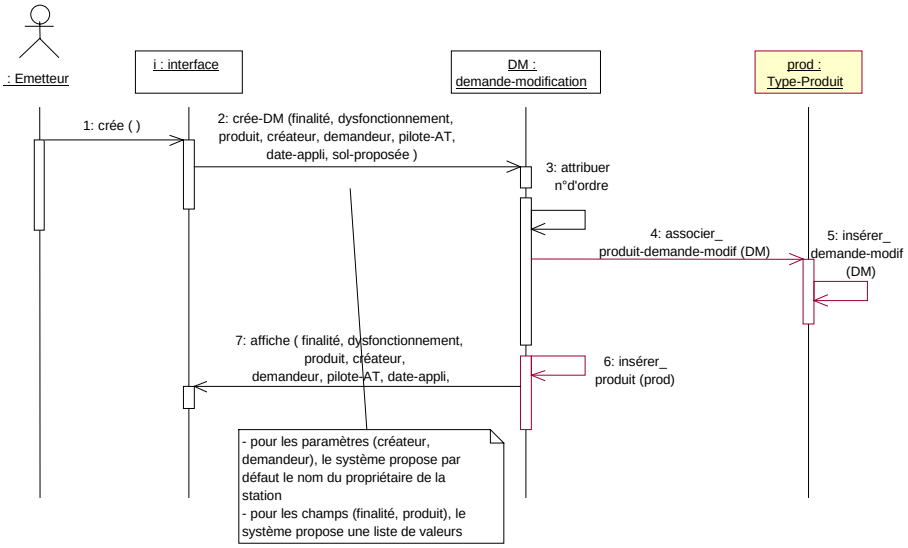
Nom	Modèle de Conception SIP.
Classification	Patron de conception
Problème	Identifier les objets du Système d'Information Informatisé (SII) associé au Système d'Information Organisationnel (SIO) à partir de la modélisation du produit et des processus métiers du SIP .
Motivation	Nous considérons le processus de gestion des modifications produit. A partir de la décomposition suivante du processus :  et en ne repérant que les activités informatisées de ce processus, telles que pour le premier sous-processus "Emission d'une Demande de Modification", les opérations suivantes :



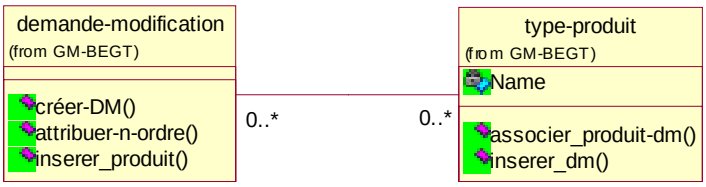
on obtient les cas d'utilisation du SII associé au processus (fonctionnalités attendues du SII), comme l'illustre la figure suivante :



En modélisant dans des diagrammes de séquence, les interactions sous forme de messages entre l'acteur et le système (informatique) nécessaires pour réaliser les cas d'utilisations attendus et par une transformation successive des diagrammes de séquences obtenus, nous identifions les sociétés d'objets du SII collaborants pour réaliser ces cas d'utilisation. Il est alors possible de construire le diagramme de classes associé, modélisant l'ensemble des objets du SII identifiés et qui enrichit le modèle d'analyse produit. Pour le cas d'utilisation "Création de Demande de Modification", le diagramme de séquence associé se présente comme suit :



et le diagramme de classes qui en découle est le suivant :

	
Force	A partir des diagrammes d'activité modélisant la décomposition des processus métiers du SIO, on aboutit à un diagramme de classe enrichissant le modèle d'analyse produit en modélisant la dynamique des objets métiers associés au produit engendrée par l'exécution des divers processus métiers (à l'aide d'opérations de classes) et en ajoutant de nouveaux objets de conception nécessaires pour l'exécution des processus informatiques du SII associé.
Contexte	Le concepteur SIP doit avoir appliquer une fois le patron d'analyse produit "Points de Variabilité" et une ou plusieurs fois le patron d'analyse processus "Décomposer un Processus" ou au moins disposer d'un modèle produit et d'un ou plusieurs modèles processus[155].
Solution-Démarche	1. Pour chaque processus métier : ▪ Etablir le modèle d'expression de besoins du SII associé au processus métier considéré. Ce modèle exprime les cas d'utilisation du SII et les documente à l'aide de diagrammes de séquence de haut niveau. Pour cela, *appliquer le patron "Expression des Besoins du SII"*. ▪ Etablir le modèle de conception du SII associé au processus métier. Ce modèle met en jeu, par une succession de modèles, des sociétés d'objets collaborants pour réaliser les cas d'utilisation du SII décrites dans le modèle d'expression de besoins du SII. Pour cela *appliquer le patron "Objets de Conception du SII"*. 2. Le diagramme de classe ainsi obtenu illustre les opérations des classes associés aux divers concepts produit mis en jeu dans le processus SIP considéré. Le modèle de conception complet du SIP est obtenu en intégrant les différentes diagrammes de classes obtenues à l'issue de l'étape 1.
Solution-Modèle	Le modèle obtenu à l'issue de la démarche proposée est un diagramme de classe représentant le modèle de conception complet du SIP. Il modélise la structure et le comportement des objets du SII et complète ainsi le modèle d'analyse produit en explicitant la dynamique des objets métiers du produit engendrée par l'exécution des processus métiers étudiés et en ajoutant de nouveaux objets (de conception).
Utilise	{Expression des Besoins du SII, Objets de Conception du SII}
Requiert(*)[156]	{Décomposer un Processus, Points de Variabilité}

Ce patron utilise donc deux patrons : "Expression des Besoins du SII" et "Objets de Conception du SII". Nous présentons brièvement dans la suite chacun de ces deux patrons. La description complète figure dans l'Annexe D (§3.2 et § 3.3).

[155] On n'oblige pas le concepteur à appliquer les patrons du catalogue SIP. Il doit dans ce cas disposer d'un modèle produit et d'un ou de plusieurs modèles processus.

[156] L'astérisque est posée pour dire que le concepteur SIP peut appliquer le patron considéré sans avoir appliquer les patrons mentionnés dans "requiert" mais que nous préconisons l'application des ces patrons car nous supposons qu'il est difficile de disposer des modèles nécessaires cités dans le contexte sans avoir appliquer les patrons mentionnés.

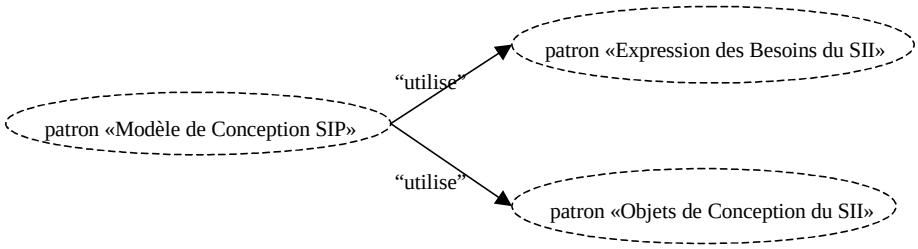


Figure 6.6 - Liens d'utilisation dans le patron "Dynamique Produit"

3.4.2. Patron "Expression des Besoins du SII

Nom	Expression des Besoins du SII.
Problème	Définir et documenter à l'aide de diagrammes de séquence les cas d'utilisation du Système d'Information Informatisé (SII) associé à un processus métier du SII.
Force	Permet de déterminer pour un processus du système d'information organisationnel (processus métier) le processus du système d'information informatisé qui lui est associé (processus informatique). Il permet d'exprimer les besoins attendues du SII associé au processus du SIO étudié dans un diagramme de cas d'utilisation du SII associé et ensuite de documenter chacun des cas d'utilisation à l'aide d'un diagramme de séquence de haut niveau.
Contexte	Le concepteur SIP doit avoir appliquer n fois le patron d'analyse processus "Décomposer un Processus" jusqu'à arriver aux opérations informatisées ou au moins disposer d'un modèle processus décomposé jusqu'aux opérations informatisées.
Solution-démarche	1. Dans la décomposition obtenue du processus métier considéré, repérer les opérations du type "informatisée". Ces opérations constituent les opérations des processus du système informatique associé au SIP, càd le SII (Système d'Information Informatisé). Elles correspondent aux cas d'utilisation de ce système. 2. Représenter ces opérations informatisées dans un diagramme de cas d'utilisation. Pour cela, procéder en suivant les règles suivantes : ▪ Chaque opération du diagramme d'activité est un cas d'utilisation dans le diagramme de cas d'utilisation. ▪ Les ressources intervenant dans la réalisation de l'opération ou de son processus (processus dans lequel l'opération est un composant direct) sont les acteurs du cas d'utilisation associé à l'opération s'ils sont des acteurs du système informatique (càd ses utilisateurs directs). Un cas d'utilisation est une fonction attendue d'un acteur principal mais peut impliquer des acteurs qu'on qualifie de secondaires (ils n'agissent pas dans le cas d'utilisation - ils sont passifs). Désigner parmi les acteurs du cas d'utilisation ainsi identifiés, l'acteur principal (celui qui agit sur le système pour satisfaire le cas d'utilisation - il est souvent celui qui a le rôle de "responsable" dans le diagramme d'activité). Les autres acteurs sont des acteurs secondaires. ▪ Construire le diagramme de cas d'utilisation en mettant en relation les cas d'utilisations entre eux et avec les acteurs. Trois types de relations peuvent exister : - Relation de communication : entre un cas d'utilisation et l'acteur qui lui est associé. Pour distinguer les différents acteurs, associer aux relations cas d'utilisation - acteur secondaire un stéréotype "destinataire". - Relation de type "extend" : entre deux cas d'utilisation A et B lorsque le cas A étend le comportement du cas B (il est un cas particulier de ce dernier). - Relation de type "use" : entre deux cas d'utilisation A et B lorsque le cas B comprend

	(dans ses actions) le comportement décrit par le cas A. D'autres types de relations spécifiques entre cas d'utilisation peuvent être définies, en stéréotypant les liens. Remarque : lorsque le processus métier étudié est de large échelle et fait donc intervenir plusieurs acteurs et cas d'utilisation, il est préférable d'organiser les divers cas d'utilisation associés (ainsi que les acteurs concernés) dans différents paquetages UML. Ceci facilite la lecture des différents modèles. Ce regroupement de cas d'utilisation peut se faire selon plusieurs critères. A titre d'exemple, selon le site industriel concerné par les cas d'utilisation. 3. Chaque cas d'utilisation dans le diagramme de cas d'utilisation obtenu apparaît comme un scénario qui peut être documenté sous forme graphique, au moyen de diagramme de séquence. Pour chaque cas d'utilisation, construire le diagramme de séquence de haut niveau associé. Pour cela, procéder comme suit : ▪ Observer à l'intérieur du cas d'utilisation la séquence, selon un point de vue temporel, des interactions entre les acteurs et le système. Ces interactions se décrivent en termes d'informations échangées pour réaliser la fonctionnalité attendue. ▪ Construire le diagramme de séquence correspondant avec, comme objets les acteurs et le système et comme messages, les interactions successifs entre ces objets. Un message est représenté au moyen d'une flèche horizontale orientée de l'émetteur du message vers le destinataire. Sur chaque flèche, indiquer le nom de l'interaction associée. Remarque : lorsqu'il existe plusieurs variantes de scénario au sein du cas d'utilisation, il est préférable de construire autant de diagramme de séquence que de variantes de scénario.
Solution-Modèle	Le modèle obtenu est un ensemble de diagrammes de séquence de haut niveau, documentant les cas d'utilisation du SII associé au processus métier étudié.
Requiert(*)	{Décomposer un Processus}

3.4.3. Patron "Objets de Conception du SII"

Nom	Objets de Conception du SII.
Problème	A partir des besoins attendues d'un système informatiques exprimés en terme de cas d'utilisation et documentés à l'aide de diagrammes de séquence de haut niveau, arriver à déterminer les objets mis en jeu dans le système informatique pour répondre à ces besoins, en terme de structure statique (classes + attributs + associations) mais également en terme de dynamique de ces objets (opérations sur les classes).
Force	Ce patron propose une démarche pour déterminer un diagramme de classes à partir d'un diagramme de séquence de haut niveau associé à un cas d'utilisation du SII, grâce à une transformation continue des modèles UML, garantissant ainsi une cohérence entre les divers modèles. Ce patron permet d'enrichir le modèle d'analyse du SIO en modélisant la dynamique des objets métiers de ce modèle, engendrée par l'exécution des cas d'utilisation associés aux processus métiers du SIO et en ajoutant de nouveaux objets nécessaires pour l'exécution du cas d'utilisation.
Contexte	Le concepteur doit avoir appliquer le patron de conception "Expression des Besoins du SII" ou disposer d'un diagramme de séquence de haut niveau documentant les cas d'utilisation du SII considéré. Le concepteur doit par ailleurs appliquer le patron d'analyse produit "Points de Variabilité" ou disposer d'un modèle d'analyse recensant les objets métiers du SIO.
Solution-Démarche	1. Dans le diagramme de séquence de haut niveau considéré, l'ensemble des interactions entre acteur et système constituent des événements extérieurs au système. Ces événements

	externes se traduisent par des interactions entre les objets du système (informatique) pour répondre à ces sollicitations, sous formes d'échanges de demandes de service entre les différents objets du système. Il s'agit donc ici de mettre en évidence comment des sociétés d'objets collaborants viennent réaliser les interactions décrites dans les diagrammes de séquences de haut niveau. Ces objets sont initialement ceux qui découlent de la phase d'analyse de besoins (objets du domaine) et ils sont complétés par des objets de conception. Ceci s'apparente à un effet de zoom à l'intérieur de l'objet unique qui représente le système dans le diagrammes de séquence de haut niveau. Construire alors le diagramme de séquence de bas niveau associé au diagramme de séquence de haut niveau considéré : ▪ A chacune des interactions du diagramme de séquence de haut niveau, associer une collaboration d'objets. Une collaboration d'objets réalise l'interaction en échangeant des demandes de service entre objets. ▪ A chacun des objets identifiés, associer un type général, càd une classe. ▪ Les demandes de services entre objets se traduisent par des opérations sur les classes associées à ces objets. En effet, une demande de service entre deux objets est une communication qui déclenche une activité dans l'objet destinataire de la demande, càd une opération sur la classe associée à l'objet en question. Traduire alors les différentes demandes de services en opérations de classes. ▪ Associer aux opérations de classes identifiées leurs paramètres, qui ne sont autre que les données communiqués lors des demandes de service. Ces paramètres constituent les attributs des classes mises en jeu. 2. A la construction du diagramme de séquence de bas niveau, les classes sont identifiées et ils sont décrits par les opérations et les attributs. On peut alors déduire le diagramme de classe partiel issu de cette analyse. Les liens entre les classes se déterminent selon le type d'opérations entre elles. Il est ensuite possible d'affiner ce diagramme de classe en introduisant des super-classes pour les classes qui s'avèrent généralisables. Remarque : on peut également déduire le diagramme d'état associé à chaque classe. En examinant la séquence des opérations sur la classe, dans les différents diagrammes de séquences où elle y figure, il est possible de déduire les différents états de la classe. Le diagramme d'état est la représentation de ces différent états avec les événements déclencheurs des diverses transitions et les actions qui s'exécutent dans la classe (qu'on détermine parmi les opérations de la classe).
Solution-Modèle	Le modèle obtenu à l'issue de la démarche proposée est un diagramme de classe représentant une partie du modèle de conception du SIP. Il modélise la structure et le comportement des objets du SII et complète ainsi le modèle d'analyse du SIO en explicitant la dynamique des objets métiers de ce modèle, engendrée par l'exécution des cas d'utilisation associés aux processus métiers du SIO et en ajoutant de nouveaux objets nécessaires pour l'exécution du cas d'utilisation.
Requiert(*)	{Expression des Besoins du SII, Points de Variabilité}

Nous soulignons que nous avons isolé la démarche de transformation des diagrammes de séquence en diagrammes de classes dans un patron spécifique (le patron ci-dessus présenté) afin de favoriser la réutilisation de cette démarche de transformation de modèles UML dans divers contextes autres que le contexte présent, relatif à l'ingénierie des SIP.

3.5. Principes d'une démarche d'utilisation du catalogue de patrons

Si l'on veut préconiser une démarche d'utilisation du catalogue de patrons, dans le but de spécifier des SIP, nous proposons de commencer d'abord par les patrons d'analyse produit. Cette première étape permet de fixer les concepts produit gérés dans le SIP et de les structurer dans des diagrammes de classes. Cette étape sera utile pour la spécification des processus à l'aide des patrons d'analyse processus en fournissant les entrants/sortants du processus (qui sont en fait les concepts produit gérés, identifiés à l'aide des patrons d'analyse produit). Ces patrons d'analyse permettent donc d'élaborer le modèle d'analyse du SIP et préparent l'élaboration du modèle de conception, à l'aide des patrons de conception. La description des processus ainsi obtenue par application des patrons d'analyse processus permettra de spécifier les traitements informatiques associés au SIP. La description des objets métiers obtenue par application des patrons d'analyse produit sera complétée (aspect dynamique spécifique au métier) et enrichie (objets de conception) par les patrons de conception.
Ainsi, une démarche de spécification de SIP est basée sur l'utilisation successive des patrons d'analyse produit, des patrons d'analyse processus et enfin des patrons de conception. Le patron "Spécifier un SIP" présentée au §3.1 a pour objectif de guider l'utilisation du catalogue de patrons et constitue ainsi le point d'entrée de ce catalogue (cf. Figure 6.7). Il établit l'ordre dans lequel les patrons d'entrée de chacune des trois catégories de patrons (analyse produit, analyse processus, conception) doivent être appliqués et ce notamment grâce aux liens du type "requiert".

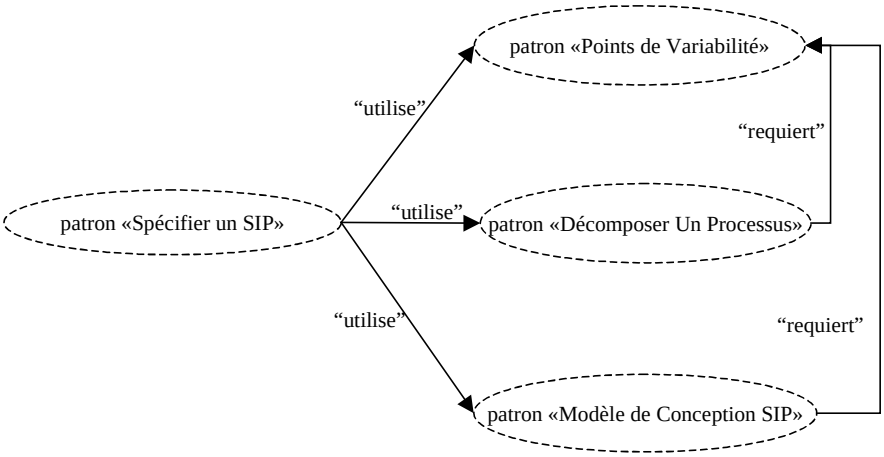


Figure 6.7 - Point d'entrée du catalogue des patrons SIP

Ensuite, l'ordre d'utilisation des patrons à l'intérieur d'une même catégorie (analyse produit, analyse processus ou conception) est défini également par les liens entre les patrons, mis en évidences dans les sections précédentes et que nous récapitulons dans la Figure 6.8, la Figure 6.9 et la Figure 6.10.

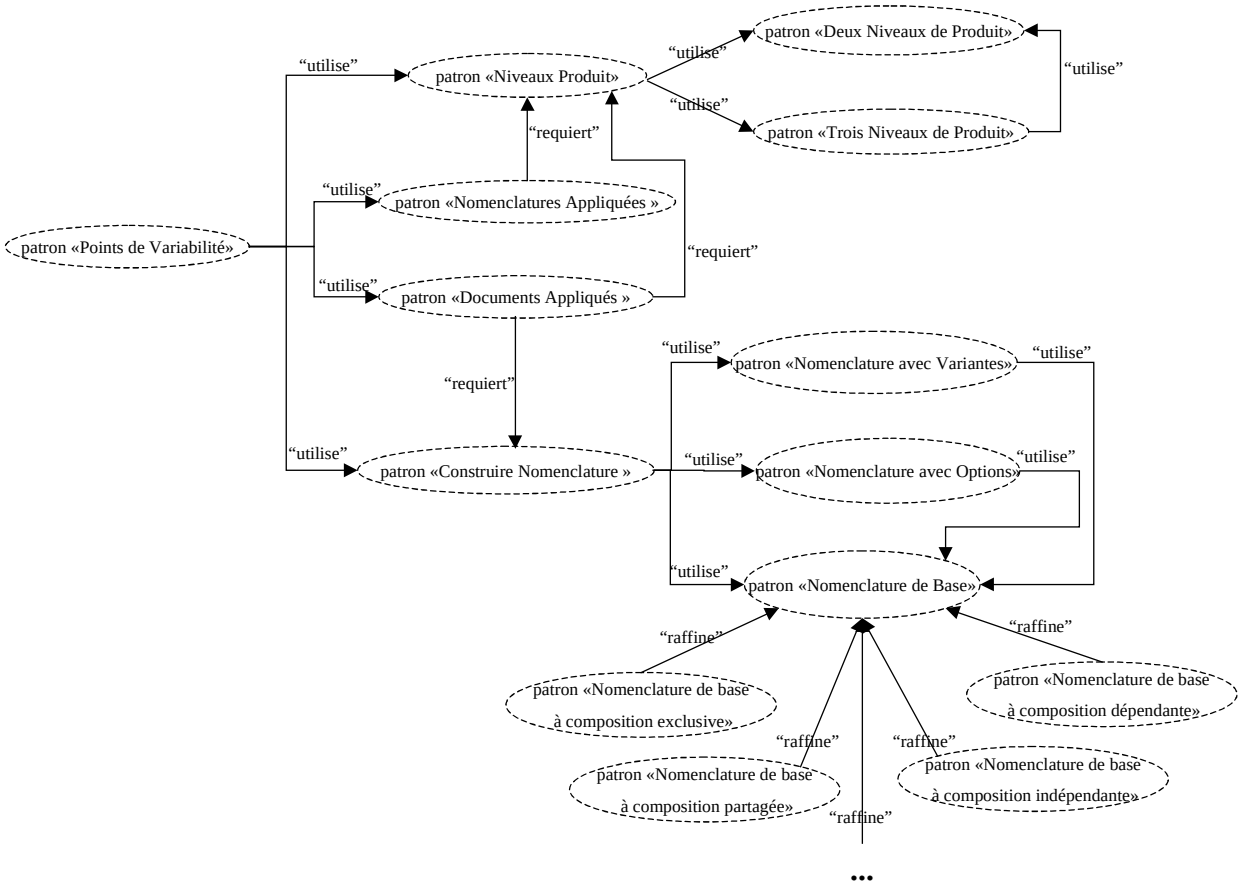


Figure 6.8 - Patrons d'Analyse Produit

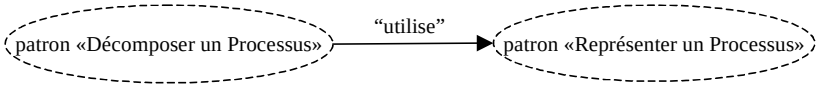


Figure 6.9 - Patrons d'Analyse Processus

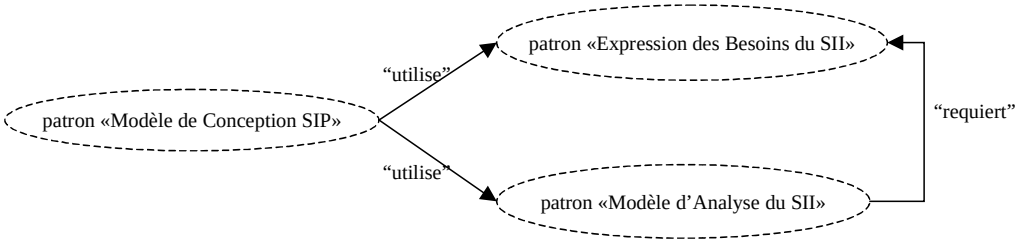


Figure 6.10 - Patrons de Conception

4. Conclusion

Ce chapitre a illustré l'organisation des diverses connaissances de domaine et de développement, identifiées lors de l'analyse de domaine, dans des patrons dédiés à l'ingénierie des SIP. Trois catégories de patrons sont définis :

- les patrons d'analyse produit, pour construire le modèle d'analyse du SIP (objets métiers relatifs au produit et gérés dans le SIP);
- les patrons d'analyse processus, pour analyser et représenter les processus métiers du SIP;
- les patrons de conception, pour construire le modèle de conception du SIP (objets informatiques associés aux divers objets métiers).

Il est sous-entendu que la liste des patrons proposés n'est pas exhaustive. Comme la majorité des langages de patrons proposés dans la littérature, il est difficile d'être exhaustif dans le traitement des problèmes rencontrés. Souvent, les problèmes les plus récurrents sont capitalisés. Le domaine d'ingénierie des SIP étant riche et complexe ; des patrons supplémentaires peuvent être définis pour spécialiser les problèmes traités dans notre catalogue (raffiner) ou pour traiter d'autres problèmes qui s'avéraient récurrents.

Par ailleurs, nous avons pu résoudre les problèmes soulignés à la fin du chapitre II, lors du choix des techniques à la base de l'approche que nous proposons (formalisme UML et patrons). Nous avons en effet souligné la nécessité de définir :

- une démarche de transformation entre les divers diagrammes d'UML, à différents niveaux d'abstraction et qui assure un continuum,
- les moyens qui assurent une gestion efficace des patrons, en matière d'organisation de ces patrons.

Pour la démarche de transformation continue de modèles UML, nous avons développé des patrons permettant d'assurer ce continuum : les patrons "Expression des Besoins du SII" et "Objets de Conception du SII".
Pour l'organisation des patrons, les diverses relations entre les patrons mises en jeu dans le formalisme de représentation de patrons proposé devraient permettre une meilleure gestion des patrons. Il est à souligner que le formalisme proposé offre également la possibilité de sélection de patrons à partir de rubriques qualifiées d'"interface". En particulier les rubriques classification, problème et contexte. Soulignons enfin l'originalité de la solution du patron qui sépare clairement le savoir et le savoir-faire associés à la résolution du problème.

Finalement, il est important de souligner que le catalogue développé constitue une proposition de patrons. Essentiellement, parce qu'il faut vérifier la réutilisabilité des patrons proposés, en les appliquant maintes fois dans des projets différents d'ingénierie de SIP, d'où l'intérêt d'expérimenter l'approche proposée à base de patrons sur un projet concret. C'est l'objectif du prochain chapitre (chapitre VII) d'illustrer cette expérimentation sur un projet industriel, mené chez notre partenaire industriel : le projet VEGA2-électronique pour la gestion du dossier technique de produits électroniques.
Par ailleurs, une approche d'ingénierie de SI, aboutissant à l'informatisation dans une étape avancée d'un projet de développement, ne saurait se passer d'un outil informatique pour supporter l'ensemble de la démarche et démontrer sa faisabilité logicielle. Le prochain chapitre décrit également cet outil support.

Chapitre VII :

Expérimentation du Catalogue de Patrons

1. Introduction

L'objectif du présent chapitre est de présenter l'expérimentation des concepts développés dans notre travail de thèse. Cette expérimentation a eu deux aspects :

- valider la démarche proposée de spécification de SIP par réutilisation de patrons en appliquant le catalogue de patrons lors de la spécification d'une application industrielle. Ceci permet de tester la réutilisabilité des patrons développés et l'adéquation de l'approche proposée pour la spécification de SIP.
- munir la démarche proposée d'un outillage informatique en définissant les types d'outils pouvant être utilisés pour supporter la démarche de spécification et préparer au développement du système informatisé associé au SIP (à l'aide d'un outil logiciel SGDT).

Le présent chapitre est divisé en deux parties, selon les deux aspects énoncés ci-dessus.

La première partie illustre l'application du catalogue de patrons dans des projets de spécification d'applications SIP chez notre partenaire industriel. En partant de l'expression des besoins collectés par le Chef de Projet Utilisateur, un modèle de spécification de l'application est proposé. Nous sommes intervenus dans trois projets d'évolution du SIP de l'activité Appareillage Basse Tension (ABT) du DAS BTP[157] de l'entreprise Schneider Electric. Ces projets concernent des sous-parties du SIP global de l'activité :

- le projet *VEGA1-mécanique* pour la gestion du processus de modification de produits mécaniques. Ce processus a ensuite fait l'objet d'une réorganisation et le projet VEGA1-mécanique a été alors suivi d'un second projet *EvolProd* pour la gestion du nouveau processus de modification.
- le projet *VEGA2-électronique* pour la gestion des produits électroniques.

Le projet VEGA1-mécanique nous a permis d'expérimenter les patrons d'analyse processus et les patrons de conception. Le projet EvolProd nous a permis d'expérimenter les patrons d'analyse processus seulement. Le projet VEGA2-électronique nous a permis d'expérimenter l'ensemble du catalogue proposé (patrons d'analyse produit, patrons d'analyse processus et patrons de conception). Il a par ailleurs donné suite à implantation au niveau de Schneider Electric. L'illustration de l'application de patrons que nous présentons dans ce chapitre porte sur le projet VEGA2-électronique.

La deuxième partie propose un outil informatique support à la démarche d'ingénierie de SIP proposée. Elle a pour objectif d'illustrer la faisabilité logicielle de la démarche proposée mais également de proposer un support à cette démarche. Nous proposons, à partir de quelques outils existants dans le commerce et de prototypes de recherche, une architecture de l'outil pouvant supporter la démarche et les phases ultérieures d'informatisation (développement du système informatique associé au SIP sur un SGDT). Cette architecture est basée sur l'utilisation de l'outil SGDT du commerce WINDCHILL et du prototype de manipulation de patrons AGAP, développé au laboratoire LSR [Duong, 00].

[157] Domaine d'Activité Stratégique : Basse Tension Puissance.

2. Application du catalogue au projet VEGA2-électronique

L'objectif de notre expérimentation est d'analyser et de spécifier les besoins exprimés par les utilisateurs (aboutir à des modèles de spécification), par réutilisation des patrons proposés, en partant de l'expression des besoins par les utilisateurs.
Cette expérimentation illustre ainsi la deuxième étape dans l'ingénierie de SIP, c'est-à-dire le processus d'ingénierie de SIP *par réutilisation*. Nous rappelons qu'au début du chapitre III, nous avons souligné que l'introduction de patrons dans le processus d'ingénierie de SIP fait évoluer le processus classique d'ingénierie vers deux processus (cf. Figure 7.1): le processus pour la réutilisation (capitalisation des connaissances dans des patrons) et le processus par la réutilisation (ingénierie de SIP par réutilisation de patrons).

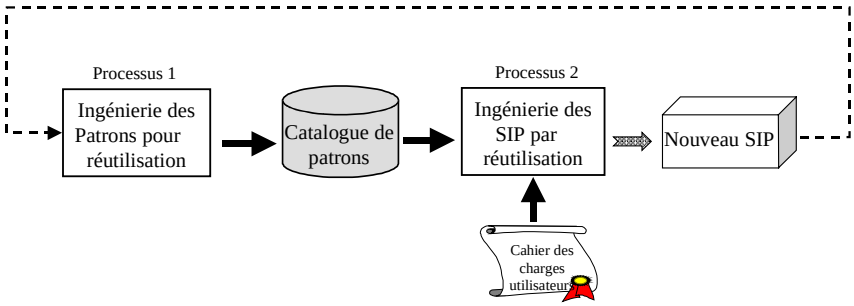


Figure 7.1 - Deux processus complémentaires pour l'ingénierie des SIP

L'ingénierie des SIP par réutilisation consiste essentiellement en la sélection et la réutilisation de patrons. Bien que ce sujet soit encore du domaine de la recherche, quelques mécanismes pour supporter ce processus sont proposés dans la littérature. D. Rieu propose dans [Rieu, 99d], deux opérations sur les patrons permettant de réutiliser les patrons proposés ; il s'agit de l'*imitation* et de l'*intégration d'imitations*.

1. **Imitation de patrons** : il s'agit de l'imitation d'une solution à un problème analogue. L'imitation porte sur les solutions-modèles des patrons et peut être facilitée par les solutions-démarches. Un patron n'est pas directement instancié au sens de l'instanciation objet. Le concepteur doit donc le dupliquer puis l'adapter à un contexte donné. L'adaptation consiste essentiellement à renommer des classes et des propriétés. Elle peut se poursuivre par l'ajout ou la suppression de propriétés. En imitant une solution-modèle, les classes (respectivement les associations et les propriétés) obtenues par duplication des éléments de la solution-modèle considérée peuvent être stéréotypées avec les noms des classes (respectivement des associations et des propriétés) dont elles sont la duplication.
   Par ailleurs, la solution-modèle d'un patron peut être imitée (donc dupliquée et adaptée) autant de fois que des objets métiers dans le SIP étudié sont organisés de la même façon que les classes de la solution-modèle. La Figure 7.2 illustre trois imitations d'un modèle associant à chaque article les documents qui le décrivent. A l'issue de n imitations, une classe ainsi que l'association qui lui est rattachée dans la solution-modèle sont dupliquées n fois.
   Ces trois imitations du même patron mettent en évidence que l'opération d'imitation n'est ni une instanciation, ni une spécialisation de modèle mais essentiellement une différenciation dirigée par le contexte.

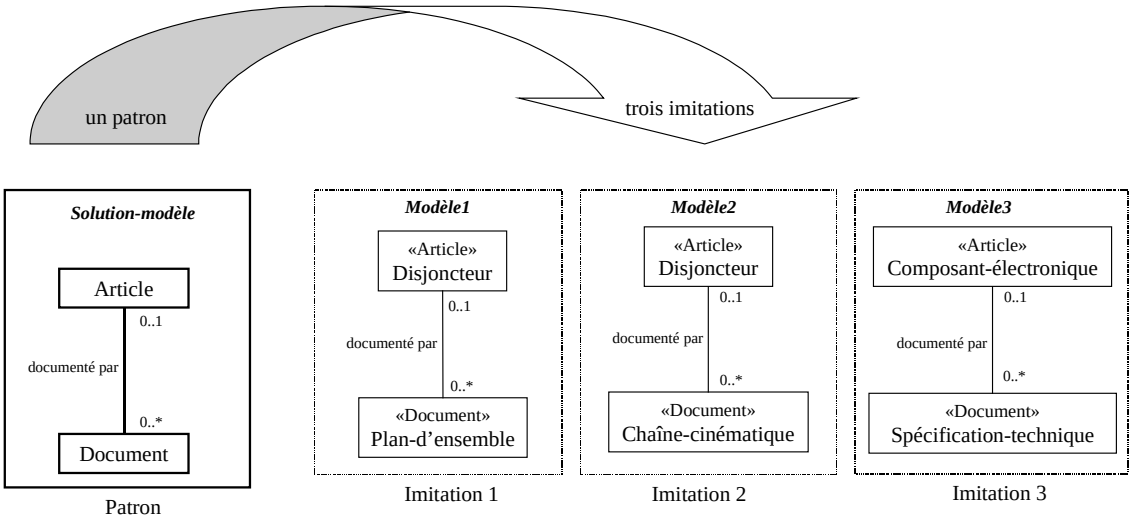


Figure 7.2 - Plusieurs imitations d'un patron

2. **Intégration d'imitations** : le résultat de l'imitation de plusieurs patrons ou de plusieurs imitations du même patron correspond à plusieurs sous-modèles qu'il s'agit d'intégrer afin de disposer d'un modèle cohérent du système à concevoir. L'intégration de classes consiste essentiellement à unir et à regrouper les classes. L'union consiste à remplacer deux classes par une seule, qui regroupe alors l'union des propriétés des deux classes initiales. Le regroupement de classes consiste à créer une super-classe à l'ensemble des classes obtenues par duplication d'une même classe de la solution-modèle, portant le nom de cette classe imitée. Ce regroupement de classes est utile pour l'implantation ultérieure sur un SGDT (pour repérer les classes du SGDT à spécialiser). Nous soulignons ici l'intérêt de la rubrique solution-démarche des patrons qui aide à l'intégration des sous-modèles en indiquant par exemple les classes à fusionner ou à regrouper. La Figure 7.3 illustre l'intégration des trois imitations présentées dans la Figure 7.2.

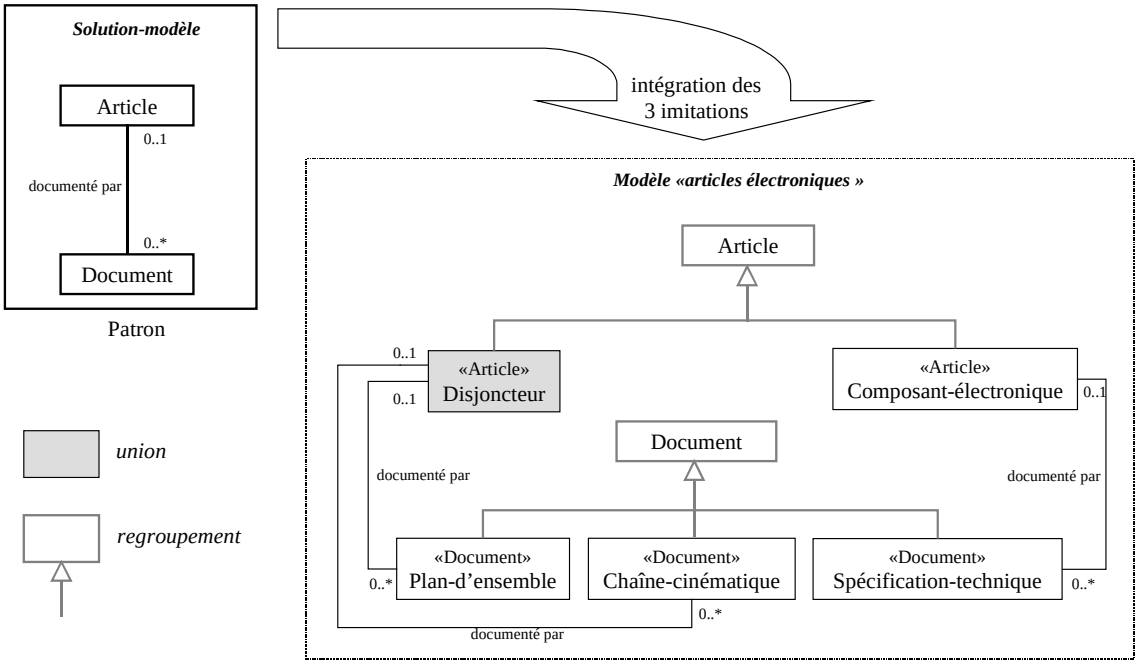


Figure 7.3 - Intégration d'imitations de patron

Des patrons de conception généraux proposent de gérer les contraintes associées à ces opérations, tel que le patron "Rôle" de P. Coad [Coad, 92] pour l'union des rôles des classes.

L'application des divers patrons dans la spécification des divers projets dans lesquels nous sommes intervenus est basée sur l'utilisation des deux opérations présentées ci-dessus.

2.1. Présentation du projet VEGA2-électronique

L'objectif du projet VEGA2-électronique est de fournir un système de gestion informatisé des données techniques propres au métier de l'électronique. Le projet concerne la mise en place d'un SIP pour le Bureau d'Etude électronique de l'activité ABT du DAS BTP. Il a pour objectif le support des activités de développement de produits électroniques au niveau du BE-électronique.

2.1.1. Activités du BE-électronique

Un produit électronique commercial (appelé Unité de Vente UV) comprend généralement une partie électronique constituée d'un ensemble de cartes électroniques[158] (circuits imprimés avec des composants) et une partie mécanique constituée de boîtiers, de système de refroidissement, de visserie, etc. (cf. synoptique du produit test Lully en Annexe E). Ces deux parties constituent ce qui est appelé Sous-Ensemble (ou encore Unité de Gestion UG). Un étiquetage externe est rajouté à ce sous-ensemble pour indiquer le réglage et la programmation du produit, personnalisés selon le client.

La conception des cartes électroniques est réalisée de manière générique pour permettre la réalisation de plusieurs variantes de cartes. Le SIP cible doit gérer la conception au niveau de trois services du BE-électronique qui interviennent dans le définition des cartes électroniques du produit :

- le *service Conception*, constitué d'électroniciens chargés de la conception des cartes. Ils définissent les schémas logiques et les fonctions des circuits imprimés à concevoir. En résultat, il produisent un schéma logique et une nomenclature de composants par variante de circuit imprimé. Ce service gère les nomenclatures sous Excel.
- le *service CFAO,* qui partant des schémas logiques définis par le service conception, conçoit les diverses variantes du Circuit Imprimé associé (le circuit qui sera fabriqué). Il définit alors les montages du "Circuit Imprimé Monté" ou CIM (la carte électronique) qui inclue notamment la définition du *routage* sur la carte et du *placement* des composants[159]. A ce niveau, il définit un montage générique du circuit imprimé considéré, incluant tous les composants possibles : c'est le "CIM source", documenté essentiellement par un schéma générique et un plan CIM source (plan de placement où figurent tous les composants qui peuvent être montés ou pas). A partir de cela, le même service (CFAO) conçoit le "Circuit Imprimé Percé" ou CIP, associé au CIM source et sur lequel les différentes combinaisons possibles de composants seront montées pour obtenir les différents CIM issus d'un même CIM source. Le CIP est documenté par un plan CIP. Le

[158] Un produit électronique peut comporter jusqu'à dix cartes.

[159] Une carte électronique peut comporter de 100 à 200 composants.

service CFAO organise ses activités de développement selon des jobs. Un job concerne la définition d'un CIP et du CIM source associé.

- le *service BE-mécanique* se charge alors de décliner les diverses variantes du CIM conçu et de compléter la définition des produits utilisants les différentes variantes de CIM. Chaque variante d'un CIM source donné (càd un CIM donné) correspond à un CIP avec un montage particulier de composants[160]. Pour ce faire, il se base sur le schéma générique et le plan du CIM source, fournis par le service CFAO mais également sur les schémas et les nomenclatures des diverses variantes de circuit imprimé, fournis par le service Conception. Il établit ainsi un dossier par variante définissant la nomenclature, le plan et le schéma du CIM considéré. Il complète également la définition des produits incluant les variantes de CIM ainsi définies. La nomenclature obtenue est appelée Sous-Ensemble. Elle est constitué en plus des cartes, de boîtiers avec diverses pièces mécaniques (telles que visserie), de conditionnement et étiquettes, de documentation, etc. Lors de la définition de cette nomenclature, la connectique est également définie en fonction des contraintes imposées par les boites.

Par rapport à la typologie que nous avons élaborée dans le référentiel (chapitre IV), le CIM source correspond à un composant d'un produit générique et le CIM à un type-produit.

2.1.2. Les principales fonctionnalités attendues du SIP

Un travail préliminaire mené en interne par F. Bounaas [Bounaas, 97] a permis de déterminer les grandes fonctions devant être assurées par l'application VEGA2-électronique. A long terme, le SIP doit supporter quatre activités du BE-électronique :

- La gestion des schémas d'origine : consiste à gérer les schémas logiques et les nomenclatures développés par le service conception.
- La gestion des jobs : consiste à gérer les activités du service CAFO, càd la définition des divers CIP et CIM source associés,
- La gestion des dossiers techniques du produit : consister à gérer les différentes variantes de circuit imprimé et les produits électroniques associés, définies au niveau du BE-mécanique,
- La gestion des contraintes mécaniques : consiste à gérer les contraintes mécaniques définies par le BE-mécanique et leur impact sur la conception des cartes.

A court terme, le choix a été fait sur le développement d'une application pour supporter la 3ème activité, càd **la gestion des dossiers techniques du produit**. Nous intervenons donc sur cette partie du projet. L'application VEGA2-électronique ne vise donc pour le moment que la gestion des phases avales de développement. Elle intervient une fois le produit défini (les différentes variantes du CIM source et les produits associés) et alors qu'une décision d'industrialisation du produit est prise. Elle gère les phases de pré-série et de série industrielle limitée. Elle concerne uniquement la gestion des types de produit (les produits définis à base de variantes de CIM source déclinées par le BE-mécanique) et éventuellement des produits physiques (les pré-séries).

En plus de la gestion du dossier technique des produits (nomenclatures, documents associés, etc.), VEGA2-électronique doit assurer :

[160] Pour un CIP donné, il peut exister de 5 à 30 variantes du CIM source.

- La gestion de l'interface avec l'application Constance gérant les composants standards du commerce ; application commune à l'ensemble des unités de Schneider et maintenue par le service Standardisation et Coordination Technique (SCT). Ceci inclut la mise à jour automatique dans VEGA2-électronique des modifications apportées dans Constance sur les composants utilisés dans VEGA2-électronique et la communication à Constance de l'ensemble des utilisateurs d'un composant donné dans VEGA-2 électronique.
- La gestion de l'interface avec Excel qui gère les nomenclatures génériques établies par les concepteurs du BE-électronique et récupérées et complétées par le service CFAO et BE-mécanique pour définir les diverses variantes de produit et leurs dossiers respectifs.
- La gestion des écarts entre les éléments évolutifs du dossier produit. La diffusion des nomenclatures aux sous-traitants est en effet accompagnée d'une fiche d'écart par rapport à l'ancienne version de nomenclature et par rapport à des données administratives liées aux composants standards (noms des fournisseurs, codes composants chez fournisseurs), dont l'évolution n'est pas toujours perceptible au niveau de l'application SIP.

Nous soulignons qu'une application antérieure "VEGA2-mécanique" a été développée pour la gestion des dossiers produit au bureau d'étude mécanique. Le souhait initial était alors de spécifier VEGA2-électronique en écart par rapport aux spécifications effectuées pour VEGA2-mécanique, c'est-à-dire en ne spécifiant que l'écart entre les deux applications. Cette démarche s'est avérée toutefois inefficace pour trois raisons. D'abord parce qu'elle dépend d'un outil existant et donc oblige les intervenants du SIP (utilisateurs et concepteurs) à raisonner en terme de solutions plutôt qu'en terme de problèmes, lors de l'expression des besoins. Ensuite, parce qu'elle limite les fonctionnalités attendues de la nouvelle application à celles présentes dans la première application. Enfin, l'application VEGA2-mécanique est très liée au métier mécanique pour pouvoir partir de là. L'adaptation de cette application au métier de l'électronique est rendue difficile ; les dossiers techniques des produits électroniques présentent des caractéristiques et des modes de structuration différentes des produits mécaniques[161].
Cette réflexion nous amène à souligner l'avantage de partir d'un référentiel commun du domaine pour spécifier à chaque fois une nouvelle application. La démarche de spécification par écart à des applications existantes pose plusieurs problèmes.

L'objectif de notre intervention dans ce projet était d'*élaborer les spécifications fonctionnelles* (le modèle d'analyse) et d'*initialiser les spécifications techniques* (modèle de conception) de l'application.
Nous avons basé notre travail sur la collecte des besoins utilisateurs, lors de réunions de spécification, qui se sont tenues régulièrement avec les différents intervenants de l'application (futurs utilisateurs, concepteurs du SIP, etc.).

[161] Liens entre objets plus complexes en électronique, périmètre des objets métiers plus large, concepts produit plus nombreux, besoin de gérer des variantes d'articles, différence terminologique[161], différence de formats pour les mêmes objets, le sous-ensemble en électronique ne peut pas exister en soi indépendamment du produit, les composants standards du commerce sont gérés par un système de gestion global commun à l'ensemble des unités de Schneider, etc..

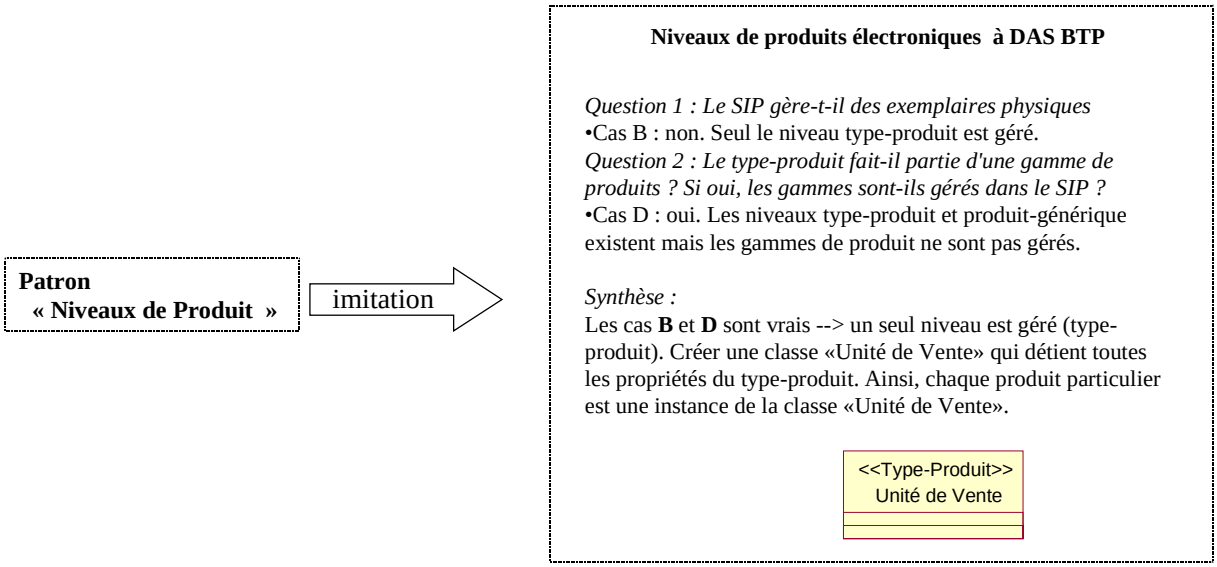
2.2. Elaboration du modèle d'analyse de l'application VEGA2-électronique

Cette section illustre l'imitation de divers *patrons d'analyse produit* pour spécifier les objets métiers gérés dans le dossier technique des produits électroniques. Elle illustre également l'imitation des *patrons d'analyse processus* pour déterminer et représenter les processus métiers mis en jeu.

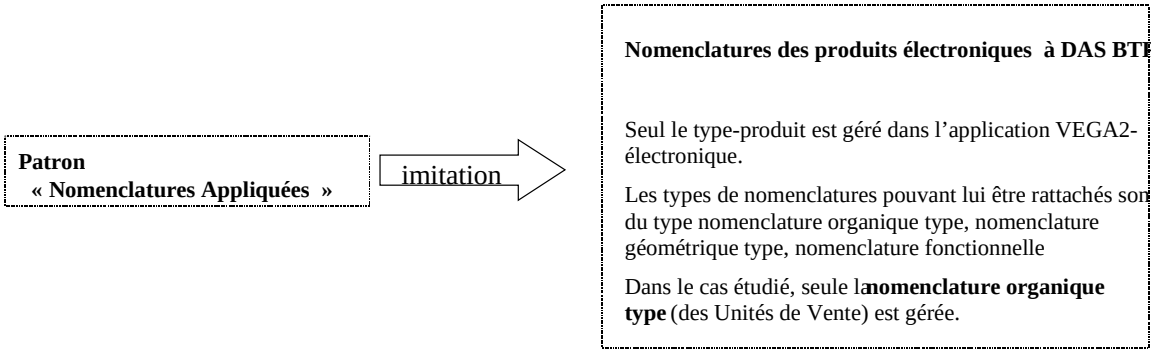
2.2.1. Imitation des patrons d'analyse produit

Comme précisé dans le chapitre VI, le point d'entrée des patrons d'analyse produit est le patron "Points de variabilité". Ce patron incite à utiliser séquentiellement les patrons "Niveaux de Produit"; "Nomenclatures Appliquées", "Documents Appliqués" et "Construire Nomenclature" avant d'associer les divers documents et nomenclatures aux niveaux de produit pour aboutir au modèle de produit global. Nous illustrons dans ce qui suit les étapes de la solution-démarche proposée par le patron "Points de Variabilité", ce qui revient à imiter les patrons "utilisés" par ce patron.

1. Application du patron "Niveaux de Produit"



2. Application du patron "Nomenclatures Appliquées"



3. Application du patron "Construire Nomenclature" et association des nomenclatures au produit

La nomenclature identifiée présente des éléments à variantes. A l'imitation du patron "Construire Nomenclature", il est préconisé d'utiliser le patron "Nomenclature avec Variantes".

Le fragment de modèle obtenu par plusieurs imitations du patron "Nomenclature avec Variantes" représente la nomenclature des produits en électronique à l'aide de composants spécifiques au métier (CIP, composant-mécanique, soft, etc.). Chacun des composants ainsi définis sont stéréotypés selon les différents types d'Elément définies dans la solution-modèle du patron "Nomenclature avec Variantes" (Elément constant feuille, Elément constant composite, Elément à variantes feuille, Elément à variantes composite). Le modèle de nomenclature ainsi obtenu est présenté en Figure 7.4.

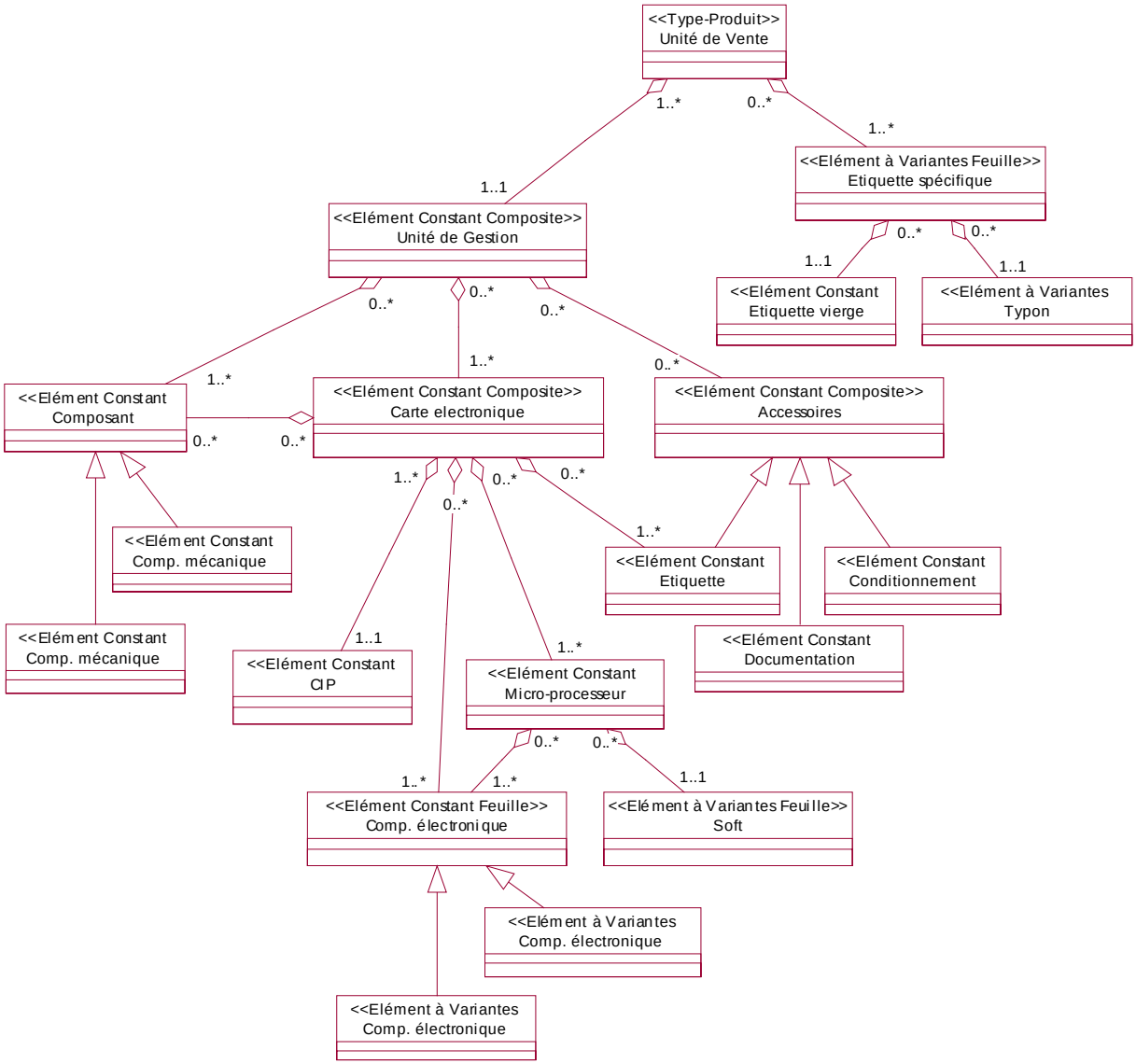


Figure 7.4 - Nomenclature des produits électroniques

Par souci de lisibilité, nous n'avons pas illustré sur ce modèle le regroupement, par généralisation, des classes selon les différentes classes de la solution-modèle du patron "Nomenclature avec Variantes" dont elles sont issues par imitation.

Le modèle obtenu illustre ainsi la nomenclature d'une Unité de Vente. Une Unité de Vente est composée d'une Unité de Gestion et d'Etiquettes Spécifiques.

- Les Etiquettes Spécifiques servent à différencier les diverses Unités de Vente issues d'une Unité de Gestion donnée. Ils spécifient des données différentes sur les Unités de Vente selon la personnalisation apportée à ces derniers.
- L'Unité de Gestion correspond à l'ensemble monté. Elle est ainsi composée de Composants mécaniques, de Cartes électroniques (des CIM : Circuit Imprimé Monté) et d'Accessoires. Chaque carte est à son tour composée d'un CIP (Circuit Imprimé Percé), de Composants électroniques, de Micro-processeurs et d'Etiquettes de cartes. Par ailleurs, un Micro-processeur est constitué de Composants Electroniques et d'un Soft. Les Composants électroniques tout comme les Composants mécaniques peuvent être internes ou externes. Il est à noter qu'on désigne par :
  - composants internes : ceux qui sont gérés par le DAS BTP (en création et évolution),
  - composants externes : soit les composants standards du commerce (gérés par le service central SCT), soit ceux qui sont gérés par un autre DAS.

  Quant aux Accessoires, ils regroupent tous les autres constituants d'une Unité de Gestion qui ne relèvent pas des parties mécanique et électronique du produit. Ainsi, les Accessoires peuvent être des Etiquettes sur le produit, de la Documentation et du Conditionnement.

4. Application du patron "Documents Appliqués"

Une fois les niveaux de produit fixés et les constituants du produit identifiés (dans une nomenclature), il est possible de déterminer les divers types de documents appliqués à chacun de ces éléments. Dans le patron "Documents Appliqués", il est préconisé de caractériser les documents selon qu'ils sont dépendants ou non-dépendants de l'objet qu'ils documentent. Dans le BE-électronique, une autre typologie de documents est employée : *représentation principale* et *représentation secondaire* (une représentation correspond à un document, au sens du référentiel SIP établi).

- Une représentation principale est une représentation créée systématiquement avec la création de l'objet qu'elle documente et qui sert à le décrire. Une représentation principale est propre à l'objet qu'il documente et elle porte le même identifiant que l'objet. Par ailleurs, il existe une seule représentation principale par objet. A titre d'exemple, le plan d'un composant mécanique est une représentation principale de celui-ci.
- Une représentation secondaire est une représentation qui documente un objet mais qui peut être partagée par plusieurs autres objets. A titre d'exemple, un cahier des charges est une représentation secondaire qui peut être associée au CIP, à un composant électronique interne, etc.

Il ressort de ces définitions qu'une représentation principale correspond à un document dépendant et qu'une représentation secondaire correspond à un document non-dépendant.

Le Tableau 7.1 résume l'ensemble des documents associés aux divers objets métiers, identifié par application du patron "Documents Appliqués".

Objet documenté	Document	Caractéristiques du Document
Sous-Ensemble	Ensemble Monté	Dépendant
	CdC	Non-dépendant
Carte électronique	Schéma	Non-dépendant
	Spécification de test	Non-dépendant
	Plan CIM	Dépendant
	CdC	Non-dépendant
Micro-processeur	CdC	Non-dépendant
Composant électronique interne	Plan mécanique	Dépendant
	CdC	Non-dépendant
CIP	Plan CIP	Dépendant
	CdC	Non-dépendant
Composant mécanique interne	Plan mécanique	Dépendant
	CdC	Non-dépendant
Documentation	Notice	Non-dépendant
	Caractéristiques	Dépendant
Conditionnement	Plan-conditionnement	Dépendant
Soft	Master	Dépendant

Tableau 7.1 - Documents gérés dans VEGA2-électronique

5. **Association des nomenclatures au produit et des documents aux objets documentés** La dernière étape de la solution démarche du patron "Points de Variabilité" consiste à associer les nomenclatures construites aux niveaux de produits identifiés et les divers documents appliqués aux objets qu'ils documentent. Nous complétons ainsi la construction du modèle d'analyse produit.

A l'issue de l'imitation des patrons d'analyse, nous obtenons donc un modèle d'analyse représentant l'ensemble des concepts métiers rattachés aux produits électroniques. Ce modèle est présenté en Figure 7.5.

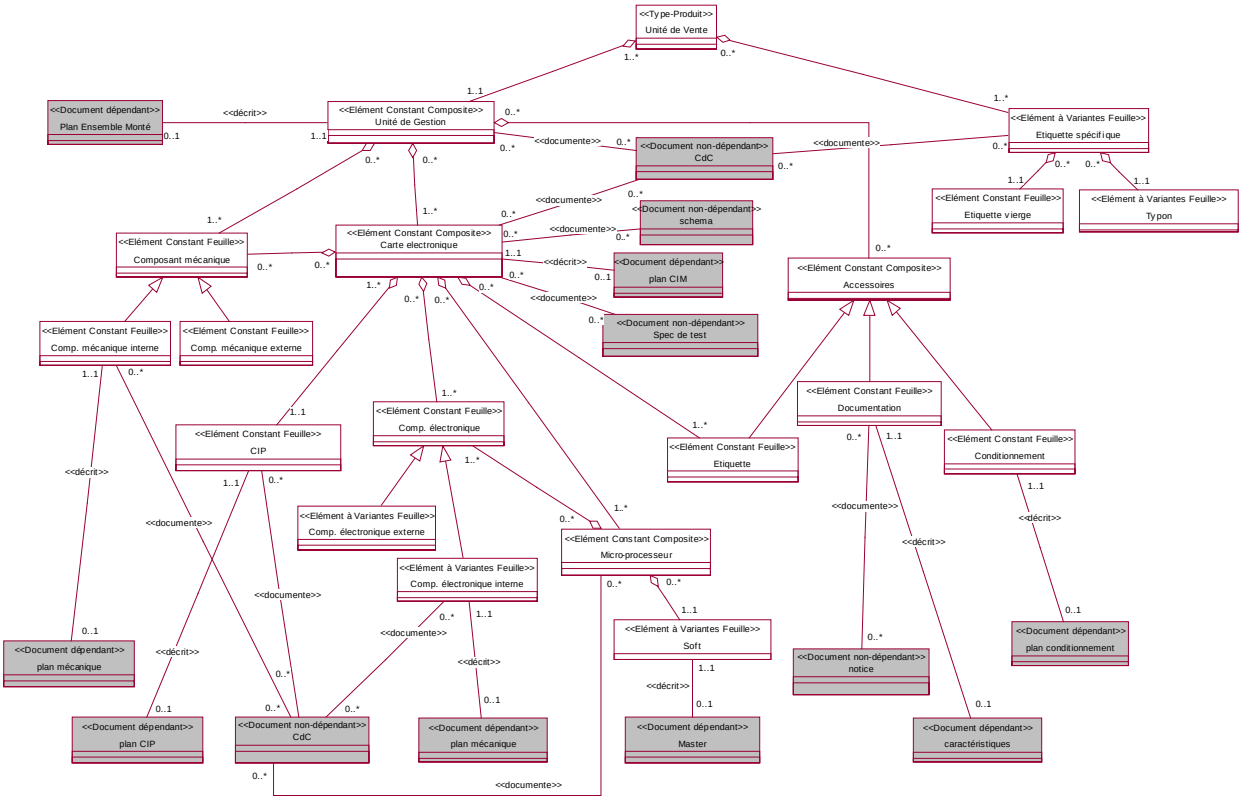


Figure 7.5 - Modèle d'analyse produit de VEGA2-électronique

Ce modèle permet d'associer à chaque type de produit (Unité de Vente) sa nomenclature organique (celle présentée sur la Figure 7.4) mais également l'ensemble des documents qui lui sont rattachés (représentés dans la Figure 7.5 avec un fond grisé). Ces documents décrivent les différents composants du type-produit. Un composant peut être soit décrit par des documents non-dépendants (correspondant aux représentations secondaires chez Schneider) soit documenté par des documents dépendants (correspondant aux représentations principales chez Schneider). A titre d'exemple, l'Unité de Gestion est décrite par un plan Ensemble monté et elle est documentée par un CdC (Cahier des Charges). Ce dernier documente également les cartes électroniques et les étiquettes spécifiques.

2.2.2. Imitation des patrons d'analyse processus

Il s'agit ici de mettre en évidence et d'analyser les activités mise en jeu pour accomplir la fonctionnalité "gestion des dossiers techniques du produit".
La gestion des dossiers techniques est le processus de plus haut niveau à assurer dans l'application étudiée. Il s'agit ici de décomposer ce processus, en imitant le patron "Décomposer un Processus" afin de représenter les différentes étapes de ce processus, la répartition des tâches entre les acteurs concernés et les activités à informatiser. Nous illustrons la décomposition de ce processus jusqu'aux activités élémentaires (opérations). Nous restreignons toutefois cette illustration, en ne nous focalisant à chaque nouvelle décomposition que sur une des activités obtenues dans la décomposition précédente.

1. Premier niveau de décomposition du processus : L'objectif du processus global considéré est de "gérer les dossiers techniques des produits". Par décomposition de cet objectif en des objectifs plus fins, nous obtenons les activités suivantes (cf. Figure 7.6) :

- Création des dossiers : à partir du schéma générique et du plan du CIM source fournis par le service CFAO mais également des schémas et nomenclatures par variante de circuit fournis par le service Conception, les gestionnaires du BE-mécanique déclinent les diverses variantes et établissent un dossier par variante incluant la nomenclature, le plan et le schéma du CIM considéré. Ils complètent également la définition des produits incluant les différentes variantes de CIM ainsi définies.
- Diffusion : les dossiers une fois créés et validés par les gestionnaires du BE, peuvent être diffusés pour industrialisation.
- Evolution des dossiers : des modifications peuvent être apportées aux variantes de CIM, aux schémas ou aux composants, durant tout le cycle de vie des produits. Une procédure d'évolution des dossiers est alors instruite. Cette procédure s'applique aux dossiers déjà validés.

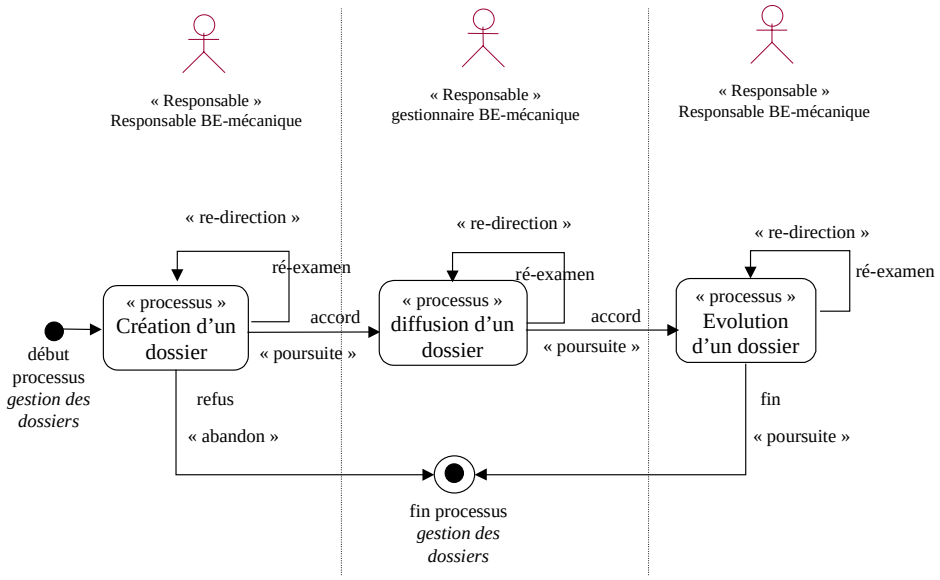


Figure 7.6 - Sous-processus de la gestion des dossiers techniques

2. Deuxième niveau de décomposition : on se focalise sur l'activité "Création d'un dossier". Nous continuons à décomposer cette activité selon le même critère de changement d'objectif. Nous obtenons les activités suivantes (cf. Figure 7.7) :

- Création de nomenclatures : la première étape dans la création du dossier d'une variante (d'un CIM) consiste à définir et à créer la nomenclature associée. Pour assurer cette activité, les gestionnaires du BE-mécanique se basent sur les nomenclatures par variante de circuit fournies par le service conception (définies dans un tableau Excel) mais également sur le schéma générique et le plan du CIM source fournis par le service CFAO. A la fin de cette activité, une étape de validation des nomenclatures créées est nécessaire pour pouvoir les diffuser.
- Création de représentations : une fois les nomenclatures créées et validées, les différentes représentations (documents) relatives au produit sont créées et associées aux objets qu'ils documentent. De même que pour les nomenclatures, une étape de validation des représentations créées est nécessaire pour pouvoir les diffuser.

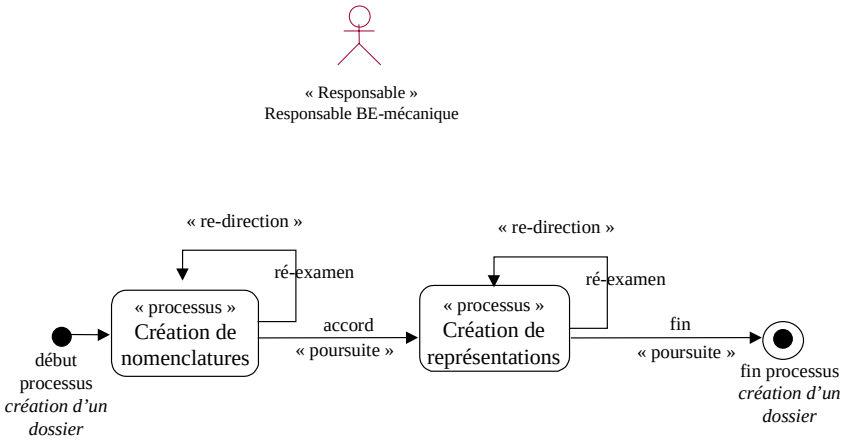


Figure 7.7 - décomposition de l'activité "Création d'un dossier"

3. Troisième niveau de décomposition : on se focalise sur l'activité "Création de nomenclatures". En se basant cette fois sur le critère de décomposition "changement d'acteur", nous obtenons la décomposition suivante :

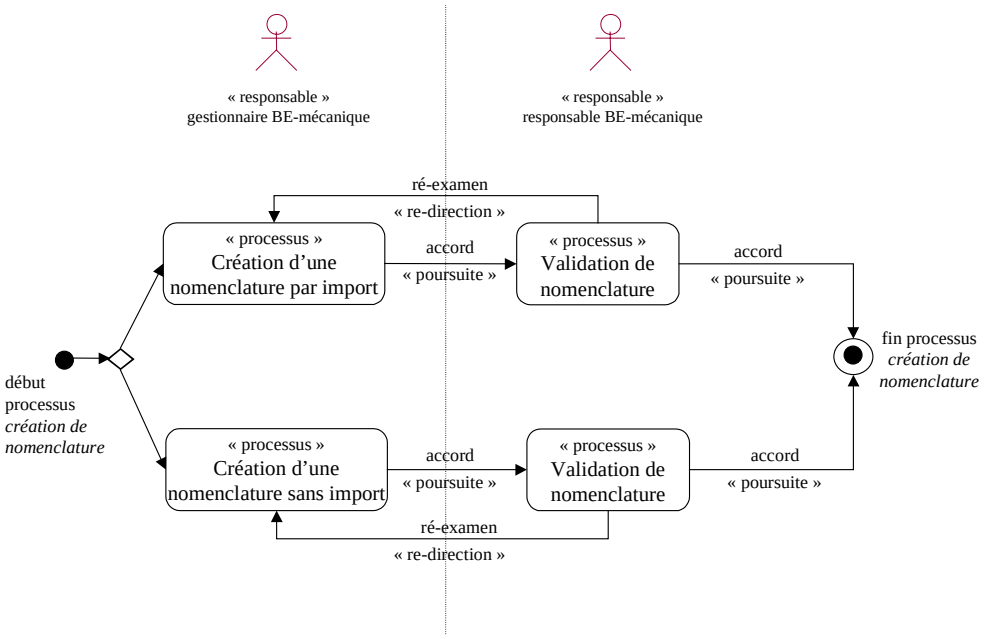


Figure 7.8 - Décomposition de l'activité "Création de nomenclatures"

Nous soulignons ici que l'import de nomenclature permet la création des liens de structures à partir d'un fichier d'import obtenu à partir d'une nomenclature définie dans un tableau Excel (la nomenclature générée par le service conception).

4. Quatrième niveau de décomposition : nous nous focalisons sur l'activité "Création d'une nomenclature par import" du sous-processus précédent (Figure 7.8). En décomposant selon le "changement d'acteur ou de rôle d'acteur" et selon le "changement de type d'activité" (manuelle ou informatisée), nous obtenons la décomposition suivante :

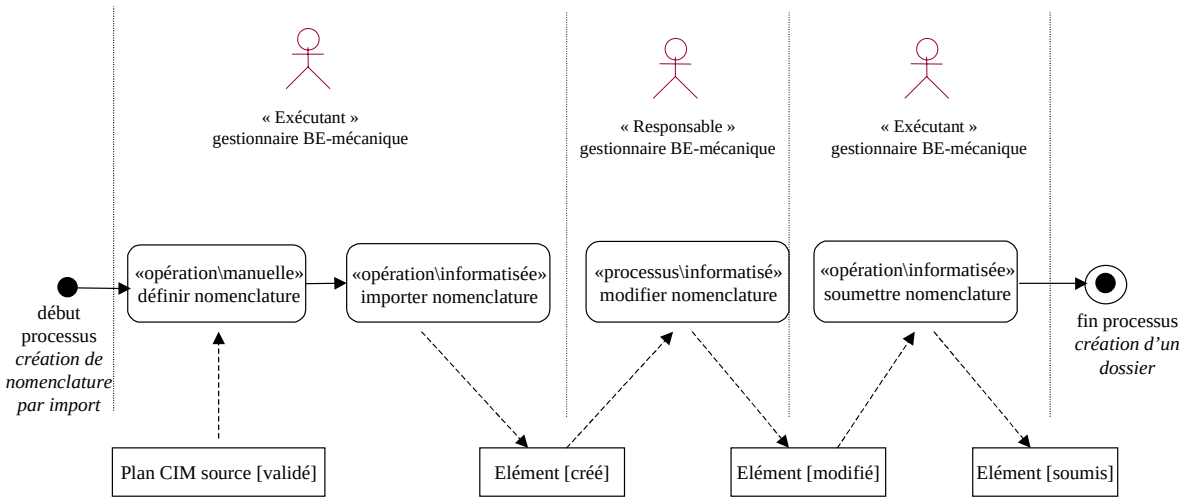


Figure 7.9 - Décomposition de l'activité "création de nomenclature par import"

Dans le diagramme obtenu, nous désignons par "Elément" tous les constituants de la nomenclature (sous-ensemble, carte, CIP, composants mécaniques et électroniques, etc.). Ces éléments prennent toutefois des états différents dans le processus "modifier nomenclature" : certains composants sont supprimés, d'autres sont rajoutés, d'autres sont modifiés (en terme d'attributs). Nous désignons ces différents états par "modifié" et c'est dans cet état qu'ils sont soumis pour validation de la nomenclature créée.

5. Cinquième niveau de décomposition : nous finissons par la décomposition de l'activité "Modifier nomenclature" obtenue dans la décomposition précédente. Nous nous basons cette fois sur le critère de décomposition "changement d'objectif".

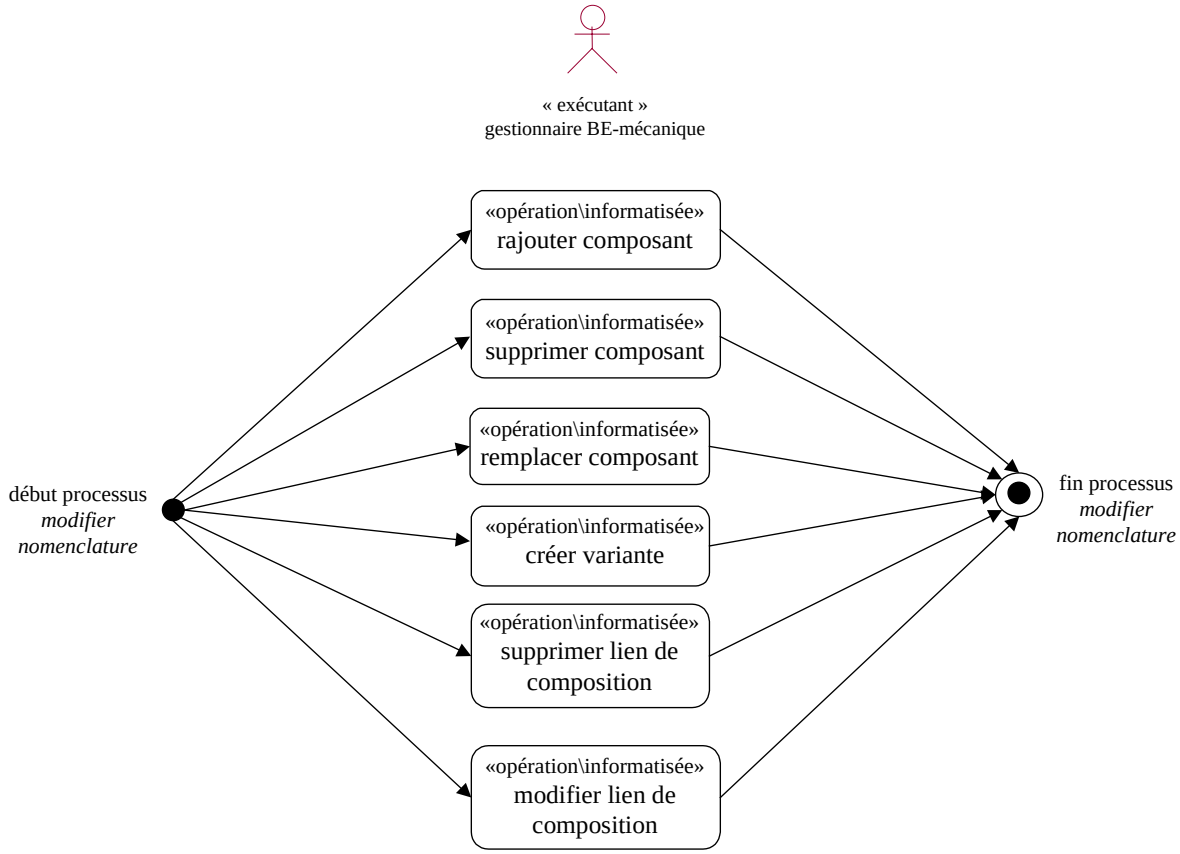


Figure 7.10 - Décomposition de l'activité "modifier nomenclature"

- Rajouter composant : cette opération s'applique à un ensemble de nomenclatures (de divers CIM) et permet le rajout d'un lien de structure avec un composant. Le composant peut être interne ou externe (Constance).
- Supprimer composant : cette opération s'applique à un ensemble de nomenclatures et permet la suppression du lien de structure avec le composant.
- Remplacer composant : cette opération s 'applique à un ensemble de nomenclatures et permet le remplacement d'un composant par un autre.
- Créer variante : cette opération permet d'ajouter une variante de composant dans la nomenclature.
- Modifier lien de composition : après un déploiement des liens d'une nomenclature, cette opération permet la modification des attributs du lien de composition : q*uantité, unité et repère-topologique*.

- Supprimer lien de composition : après un déploiement des liens d'une nomenclature, cette opération permet la destruction d'un lien.

Nous finissons ainsi l'illustration de l'imitation du patron "Décomposer un Processus" pour le cas du processus de gestion des dossiers techniques du produit. Le modèle complet, obtenu par décomposition de chacune des activités obtenues à chaque niveau de décomposition, représente le modèle d'analyse du processus.
Dans ce qui suit, nous illustrons l'application des patrons de conception sur le projet VEGA2-électronique.

2.3. Elaboration du modèle de conception de l'application VEGA2-électronique

Dans cette section, il s'agit d'imiter le patron "Construire Modèle de Conception". Nous illustrons dans ce qui suit chacune des étapes de la solution-démarche offerte par ce patron.

2.3.1. Imitation du patron "Modèle de Conception"

La solution-démarche de ce patron consiste à :

1. Pour chaque processus métier :
   - établir, le modèle d'expression de besoins du SII associé. Ce modèle exprime les cas d'utilisation du SII et les documente à l'aide de diagrammes de séquence de haut niveau. Cela nécessite d'imiter le patron "Expression des besoins du SII".
   - établir le modèle d'analyse du SII associé au processus métier pour mettre en jeu, par une succession de modèles, les sociétés d'objets collaborants pour réaliser les cas d'utilisation du SII décrites dans le modèle d'expression de besoins du SII. Cela nécessite d'imiter le patron "Modèle d'analyse du SII".
2. établir le modèle de conception complet en intégrant les différents diagrammes de classes obtenues à l'issue de l'étape 1.

Dans ce qui suit, nous illustrons donc l'imitation des patrons "Expression des besoins du SII" et "Modèle d'analyse du SII" pour le sous-processus "Création d'une nomenclature par import", étudié dans la section 2.2.2. L'étape 2 de la solution-démarche du patron "Modèle de Conception" ne peut être illustrée puisque seul ce sous-processus est étudié.

2.3.2. Imitation du patron "Expression des besoins du SII"

1. Dans la décomposition du processus métier considéré, repérer les opérations du type "informatisée" et représenter ces opérations informatisées dans un diagramme de cas d'utilisation. En considérant les opérations informatisées obtenues dans la Figure 7.9 et la Figure 7.10, nous obtenons le diagramme de cas d'utilisation suivant :

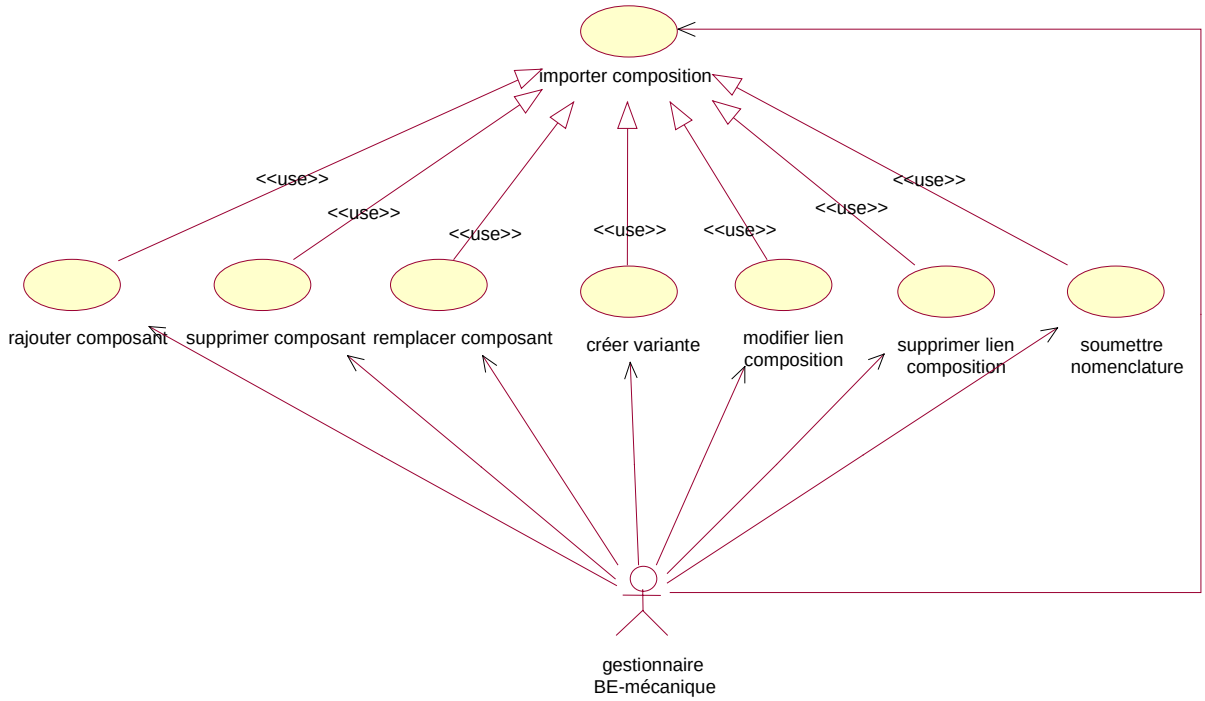


Figure 7.11 - Diagramme de cas d'utilisation partiel de l'application VEGA2-électronique

2. Chaque cas d'utilisation dans le diagramme de cas d'utilisation obtenu apparaît comme un scénario qui peut être documenté au moyen de diagrammes de séquence : nous illustrons cette imitation sur un seul cas d'utilisation du diagramme ci-dessus : "rajouter composant". Nous rappelons que cette opération s'applique à un ensemble de nomenclatures de divers CIM et permet le rajout d'un lien de structure avec un composant. Plusieurs scénarios existent pour ce cas, selon que le composant est mécanique ou électronique et qu'il soit externe ou interne. Nous considérons ici le cas de composants électroniques externes, gérés dans Constance. Dans ce cas, le rajout de composant dans une nomenclature revient à créer dans la nomenclature, définie dans Vega2-électronique, un composant électronique externe de type Constance et d'un lien de structure avec le CIM considéré. Le composant ajouté est alors initialisé à partir des informations extraites de la base Constance. Le diagramme de séquence de haut niveau documentant ce cas d'utilisation est présentée dans la Figure 7.12 (cf. page suivante).

2.3.3. Imitation du patron "Modèle d'analyse du SII"

1. Construire alors le diagramme de séquence de bas niveau associé au diagramme de séquence de haut niveau considéré (présenté dans la Figure 7.12) : le diagramme de séquence de bas niveau obtenu est présenté dans la Figure 7.13 (cf. page suivante).

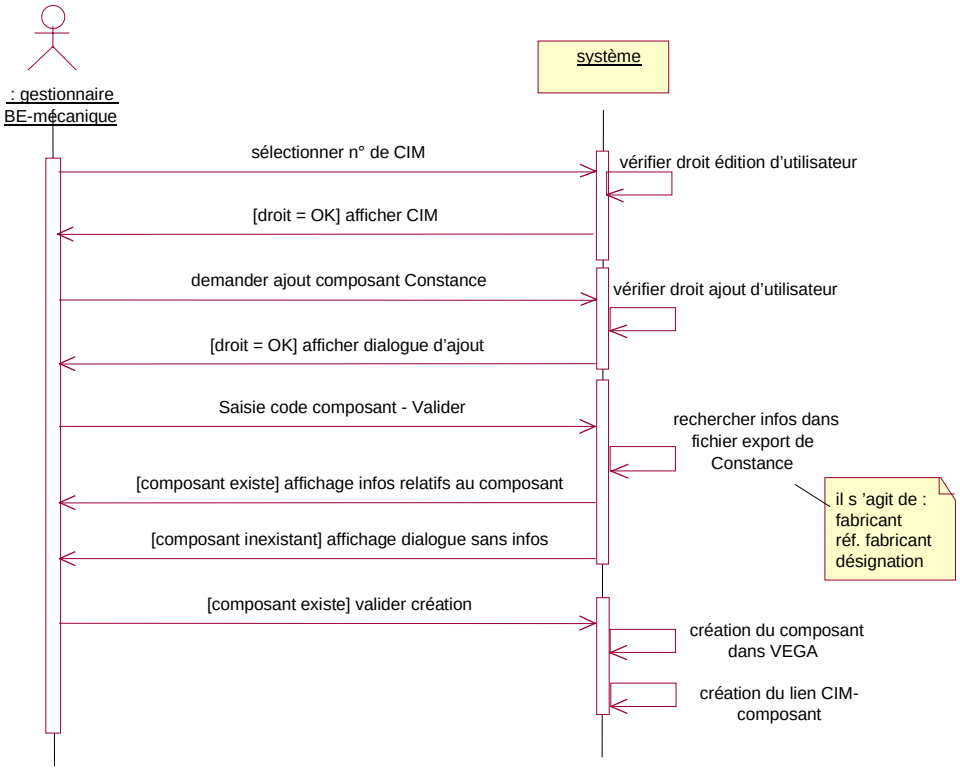


Figure 7.12- Diagramme de séquence de haut niveau du cas d'utilisation "rajouter composant"

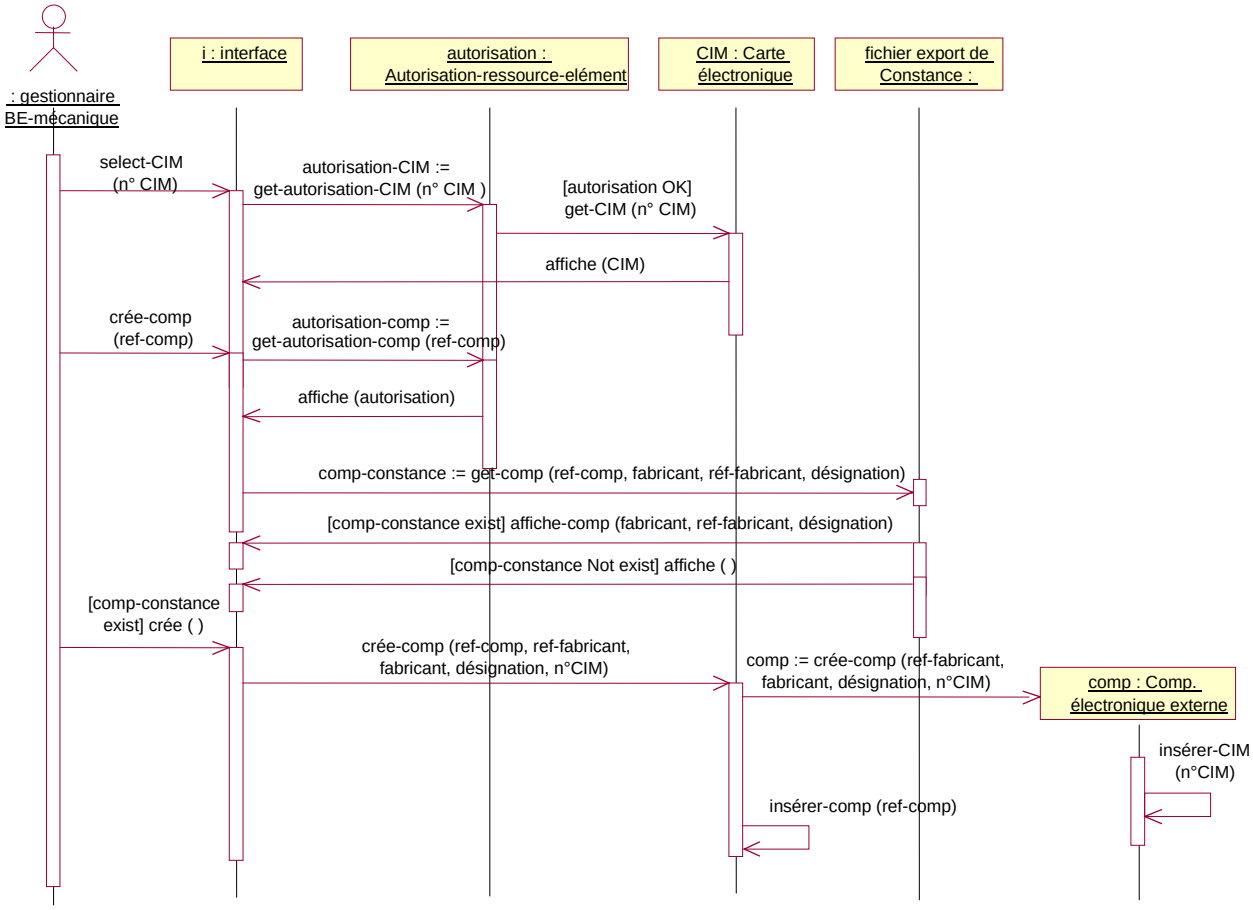


Figure 7.13 - Diagramme de séquence de bas niveau du cas d'utilisation "rajouter Composant"

2. Déduire le diagramme de classe partiel issu de cette analyse ; nous obtenons le modèle suivant :

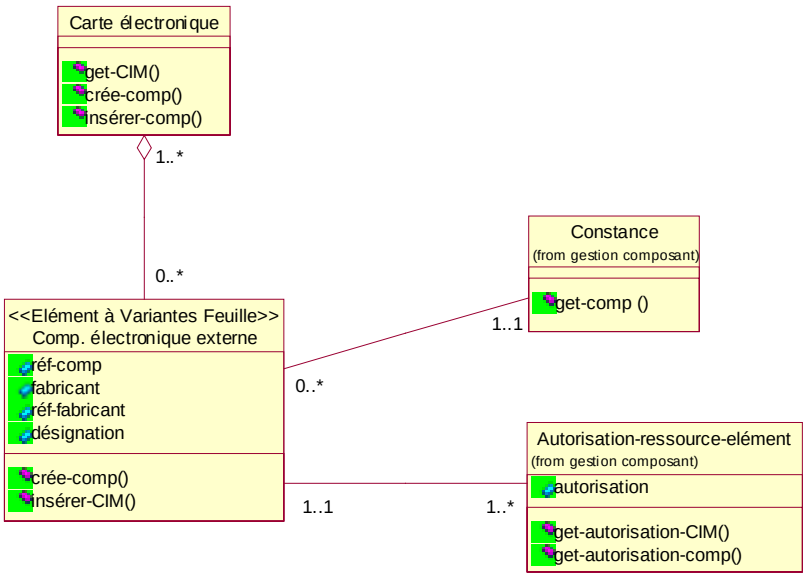


Figure 7.14 - Modèle de Conception partiel de VEAG2-électronique

Par rapport au modèle d'analyse produit élaboré au §2.2.1 (Figure 7.5), on retrouve les objets métiers "Composant électronique externe" et "Carte électronique" complétés par quelques attributs et opérations de classes. D'autres objets apparaissent lors de cette analyse; il s'agit de la classe "Constance" représentant le lien vers le système Constance et la classe "autorisation-ressource-élement" relative à l'autorisation. Cette classe représente le lien entre les classes "Activité" et "Elément" pour désigner les autorisations d'une ressource sur un élément lors d'une activité donnée d'un processus SIP.

2.4. Implantation des modèles de spécification de VEGA2-électronique

Par rapport au processus développement des SIP à Schneider (que nous avons décrit dans le chapitre I), notre intervention dans le projet VEGA2-électronique couvre la phase d'analyse (expression des besoins) et initialise la phase de conception (analyse fonctionnelle). Les modèles de spécification ainsi élaborés par imitation des patrons ont donné lieu par la suite à une troisième phase d'implantation, assurée par les équipes de développement de Schneider. L'implantation a concerné l'interface avec Constance et l'import-export de nomenclatures de et vers Microsoft Excel. Nous donnons ici un aperçu de cette étape d'implantation.

Comme nous l'avons précédemment décrit dans le chapitre I, l'implantation débute par une étape d'analyse technique au cours de laquelle les ingénieurs d'application élaborent des spécifications techniques décrivant de façon précise l'implantation dans le SGDT (Metaphase en l'occurrence) dans un Dossier d'Analyse Technique par Ecart (DATE). Le DATE indique les classes et les méthodes de Metaphase à récupérer et les ajustements à apporter. Un dossier préalable d'Analyse Fonctionnelle (DAF) peut être établi pour détailler les traitements associés et l'interface utilisateur. Les spécifications techniques ainsi décrites dans le DATE sont ensuite implantées par les développeurs sur Metaphase (codage des classes).

Pour des raisons de confidentialité, nous présentons le DAF et le DATE relatifs au cas étudié dans le §2.3 (rajout de composants Constance) en annexe (cf. Annexe E). Ces documents sont donnés à titre d'illustration et ils ne sont pas particulièrement nécessaires pour présenter l'expérimentation que nous avons conduit sur l'application VEGA2-électronique.

3. Outil Support de la démarche

L'outil support que nous proposons consiste à utiliser un atelier de manipulation de patrons qui permet d'abord de créer et d'organiser les patrons puis de les imiter (et intégrer les imitations). Ainsi, à l'entrée de l'atelier, le concepteur du SIP dispose d'une librairie de patrons. Par des mécanismes de recherche et de sélection de patrons (en se basant sur les rubriques "interface" des patrons), le concepteur choisit les patrons qu'il veut imiter. A l'imitation des solutions des patrons sélectionnés, le concepteur obtient un modèle de conception spécifique à son cas d'étude, sous forme de diagrammes de classes UML qu'il peut par la suite convertir dans un langage de programmation objet (C++ ou Java par exemple) et ensuite exporter vers un SGDT utilisant le langage choisi.
Dans ce qui suit, nous étudions donc les outils permettant de réaliser l'outil support visé.

3.1. Atelier de manipulation de patrons : l'outil AGAP

3.1.1. état de l'art sur les outils

Depuis l'émergence des patrons dans la communauté objet, plusieurs outils ont été proposés pour les mettre en œuvre. Dans [Duong, 00], trois catégories d'outils sont distinguées en fonction du type de support proposé. La première catégorie d'outils propose des extensions de langages de programmation en intégrant les éléments de description de patrons dans les primitives des langages. La deuxième catégorie d'outils fournit des systèmes de génération indépendants à la fois de l'environnement de modélisation et du langage de programmation cible. Il s'agit ici d'outils de type boite noire auxquels l'utilisateur fournit des paramètres en entrée. La troisième catégorie d'outils étend les environnements de développement. Ces derniers outils, destinés à améliorer les fonctionnalités d'un environnement de génie logiciel, paraissent les plus complets et les plus prometteurs.
Dans le cadre des activités de l'équipe Sigma du laboratoire LSR, une analyse de divers outils de cette troisième catégorie est proposée [Duong, 00].
Il ressort de cette étude que la jeunesse du concept de patron dans les systèmes d'information implique que peu d'outils les mettent aujourd'hui en œuvre. Les ateliers commerciaux étudiés présentent des lacunes : le formalisme de représentation des patrons est souvent imposé dans ces ateliers, la recherche de patrons est uniquement axée sur la rubrique solution et enfin la traçabilité n'est pas souvent assurée lors du processus d'application ; une fois que l'opération d'imitation a été effectuée, le patron est dilué dans le code.
Ainsi, un besoin pour développer un atelier permettant de s'affranchir de ces difficultés est né. C'est l'objectif du travail proposé par B. Barthès et Q.P. Duong [Duong, 00] qui porte sur la spécification et l'implantation d'un outil de manipulation de patrons, appelé AGAP (Atelier de Gestion et d'Application des PattErns). Cet atelier devra favoriser la mise en œuvre de patrons de conception (Gamma, Coad, etc.) et d'autres patrons, l'ajout de patrons, la personnalisation

de patrons, la définition de divers formalismes, la recherche de patrons sur mot clé et également la recherche poussée, l'imitation et la traçabilité des patrons.
Nous avons choisi d'utiliser l'atelier AGAP comme atelier de manipulation de patrons dans l'outil support proposé. Nous présentons dans ce qui suit cet atelier.

3.1.2. Présentation de AGAP

L'atelier AGAP proposé dans [Duong, 00] a deux principales fonctionnalités : la gestion des patrons et le développement à base de patrons. Il est destiné principalement à deux types d'acteurs, qui sont l'ingénieur de patron (Gestionnaire de patron) et l'ingénieur d'application (développeur d'applications). Nous détaillons dans ce qui suit chacune de ces deux focntionnalités.

1. **Gestion de patrons :** l'atelier permet à l'ingénieur de patrons de définir des formalismes de patrons ainsi que des systèmes de patrons décrits suivant ces formalismes et également d'organiser ces systèmes. Les principales fonctionnalités assurées sont les suivantes:
   - ✓ Visualiser un patron
   - ✓ Visualiser un système de patrons
   - ✓ Créer un formalisme
   - ✓ Modifier un formalisme
   - ✓ Valider un formalisme
   - ✓ Créer un système de patrons
   - ✓ Créer un patron
   - ✓ Modifier un patron
   - ✓ Valider un patron
   - ✓ Organiser des patrons

2. **Développement à base de patrons :** l'atelier permet à l'ingénieur d'application d'appliquer des patrons pour modéliser et concevoir son système d'information. Cette application est appuyée sur un ou plusieurs systèmes de patrons disponibles dans AGAP. Les principales fonctionnalités sont les suivantes :
   - ✓ Visualiser un système de patrons
   - ✓ Rechercher un patron dans un système de patrons
   - ✓ Visualiser un patron
   - ✓ Créer une imitation de patron
   - ✓ Supprimer une imitation

Par rapport à l'outillage que nous proposons, les fonctionnalités dans AGAP qui nous semblent intéressantes sont : *la création de systèmes de patrons* (en l'occurrence le catalogue de patrons SIP que nous avons proposé), *la création de patrons* (les divers patrons développés dans le catalogue proposé et éventuellement de nouveaux patrons à développer) et *l'imitation de patrons*. Les deux premières fonctionnalités sont utilisées une seule fois pour déclarer les patrons dans l'atelier. La dernière fonctionnalité est plus récurrente; elle est utilisée à chaque nouveau projet de spécification d'un SIP. Nous décrivons dans le Tableau 7.2 le scénario associé à chacune de ces fonctionnalités.

Fonctionnalité	Scénario
Créer un système de	1. Rechercher les formalismes prédéfinis.

patrons	2. Choisir un formalisme parmi les formalismes prédéfinis. 3. Créer un nouveau formalisme dans le cas où aucun formalisme prédéfini n'est adapté au futur système de patrons. 4. Rechercher des types de solution-modèle[162] reconnus par le formalisme choisi. 5. Choisir les types de produits reconnus par le futur système de patrons. 6. Rechercher les domaines[163] prédéfinis. 7. Choisir un domaine parmi les domaines prédéfinis. 8. Créer un nouveau domaine dans le cas où aucun domaine prédéfini. 9. Enrichir le domaine choisi en ajoutant les mots du domaine. 10. Rechercher les technologies prédéfinies. 11. Choisir une technologie parmi les technologies prédéfinies. 12. Créer une nouvelle technologie dans le cas où la technologie du futur système ne figure pas dans la liste des technologies prédéfinies. 13. Enrichir la technologie choisie en ajoutant les critères de qualité de la technologie. 14. Saisir le nom, l'auteur. 15. Enregistrer la description du système de patrons.
Créer un patron (pré-condition : il existe au moins un système de patrons défini)	1. Rechercher les systèmes de patrons existants dans l'atelier. 2. Choisir un système de patrons. 3. Saisir le nom du patron à créer. 4. Saisir le contenu des rubriques obligatoires, une par une. 5. Saisir le contenu des rubriques optionnelles, une par une. 6. Enregistrer le contenu du patron créé.
Créer une imitation (pré-condition: disposer d'un système d'information[164] courant).	1. Afficher la liste des systèmes de patron (SP) associés au SI courant. Dans le cas où il n'existerait qu'un SP pour le SI cette étape est sautée, on passe alors directement à l'étape 3. 2. Choisir un SP parmi ceux attachés au SP. 3. Afficher l'ensemble des patrons imitables[165] du SP choisi. 4. Choisir un patron. 5. Afficher l'ensemble des rubriques de type Réalisation avec leurs différents champs[166]. 6. Le champ est de type UML, ouvrir l'AGL associé (RationalRose en occurrence) a. Créer une classe stéréotypée « patron » dont le nom est celui du patron à imiter suivi du numéro d'occurrence de l'imitation. b. Exécuter les actions i, ii et iii pour chaque classe présente dans le fichier de type UML (XMI) selon la cardinalité de la classe. participante (si cardinalité n : on offre la possibilité d'effectuer autant d'itération que l'utilisateur le désire, si la cardinalité est 1 : on oblige l'utilisateur à ne faire qu'une itération, si la cardinalité est de type 0 on

[162] Chaque rubrique du patron est composée d'un ou plusieurs champs. Un champ prend un type quelconque. AGAP supporte des champs de type imitable tel que un diagramme UML, non imitable tel qu'un texte, etc.

[163] Le domaine d'application du système de patterns. Dans notre cas, il s'agit des SIP.

[164] Un SI est défini comme l'exécution d'un ou plusieurs patrons. Chaque exécution donne lieu à un certain nombre de produits. Elle se base sur le contenu du patron et les produits sont forcément des imitations des rubriques réalisation des paterons. Un SI est composé au minimum d'un nom, d'un fichier XMI et d'un type d'AGL.

[165] Un pattern est imitable s'il contient au moins une rubrique dont un champ est imitable.

[166] Pour l'instant, seul les champs valués par des diagrammes UML sont imitables.

	rend l'itération facultative) : i. deux possibilités afin d'implanter la classe participant au patron : 1. Choisir une classe du SI courant. 2. Créer une nouvelle classe ii. Créer un lien entre la classe et le patron (classe de stéréotype <<patron>>) qui est stéréotypé par le nom initial de la classe dans le patron. (Suivi du numéro d'occurrence s'il existe plusieurs imitations) iii. Ajouter les attributs et les méthodes de la classe initiale du patron. c. Ajouter les associations entre les classes.

Tableau 7.2 - Principales fonctionnalités d'AGAP

Il est à souligner que dans le Méta-modèle d'AGAP, le patron est modélisé en tant que classe. A chaque patron, est associé un certain nombre de classes représentant les diverses rubriques du patron. Chaque rubrique est composée d'un ou de plusieurs champs (de type quelconque). Ainsi, les rubriques du patron sont imitables ou non selon le type des champs. A titre d'exemple, le diagramme UML dans la rubrique Solution-modèle peut être instancié alors que le diagramme UML dans la motivation ne peut être instancié.
Par ailleurs, les manipulations sur les patrons (créer, supprimer, etc.) sont des opérations sur la classe du patron.

De la description du scénario associé à l'imitation de patrons (cf. Tableau 7.2), un passage d'AGAP à un AGL supportant le langage UML est nécessaire pour construire les diagrammes UML imitant la solution-modèle du patron (qui est exprimée à l'aide de champ de type UML). Dans le cas présent, il s'agit de l'atelier Rational/Rose. La solution-modèle d'un patron à imiter est alors transférée à Rational/Rose sous forme de fichier XMI. C'est dans Rational\Rose que les diagrammes UML représentants l'imitation de la solution du patron sont construits selon le scénario d'imitation décrit dans le Tableau 7.2 (tâche 6.). Les diagrammes ainsi obtenues dans Rose peuvent ensuite être exportés sous forme de fichiers SQL, XML, Java ou autre pour les intégrer dans un SGDT utilisant l'un de ces formats de fichier.
Deux autres outils sont donc nécessaires dans l'outil support de la démarche que nous définissons : l'atelier Rational Rose et le SGDT. Nous les décrivons dans la suite de cette partie.

3.2. Atelier de gestion de modèles UML : Rose

L'atelier de développement objet Rational/Rose, support du langage UML, est utilisé dans l'outillage proposé pour implémenter les diagrammes UML obtenus par imitation des patrons dans AGAP. L'atelier Rose propose des éditeurs graphiques de modèle basés sur le formalisme UML. Rose présente un gestionnaire de modèles qui se décompose en plusieurs vues :

- la vue use-case : supportant les diagrammes de cas d'utilisation, de séquence et de collaboration

- la vue logique : supportant les diagrammes de séquence, de collaboration, de classes et d'états-transitions
- la vue composants : supportant les diagrammes de composants
- la vue déploiement : supportant les processeurs

Dans le cadre de notre outillage, nous utilisons uniquement la vue logique pour construire les diagrammes de classes obtenus par imitation des patrons.

Les principales fonctionnalités de l'outil que nous utilisons sont d'une part la création à l'intérieur d'une vue logique d'éléments de modélisation tels que les classes, les liens, les attributs et les opérations et d'autre part la génération de code à partir des modèles UML élaborés. Rational/rose permet en particulier de générer du code C++ et des tables SQL.

3.3. Un SGDT : Windchill

Nous choisissons le progiciel Windchill pour illustrer l'outillage SGDT de la démarche. Windchill est supporté par une architecture d'exécution entièrement basée sur les technologies Web et répartie en trois tiers : client, serveur et base de données (cf. Figure 7.15).

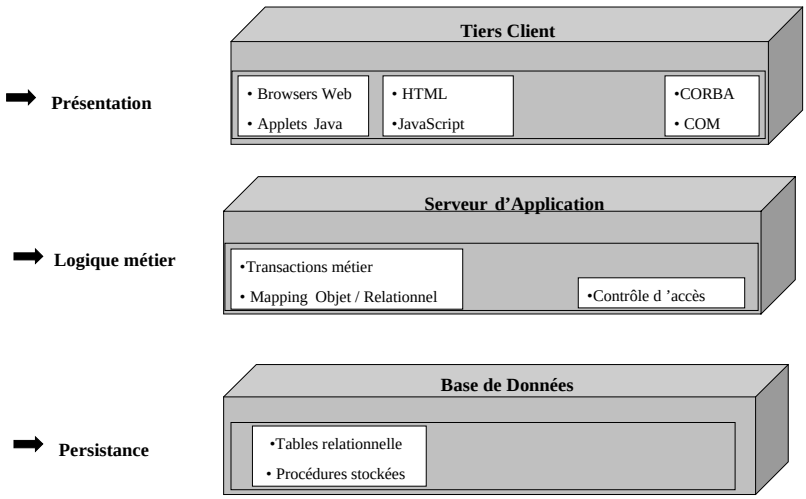


Figure 7.15 - Architecture d'exécution de Windchill

En terme de base de données, Windchill utilise le SGBD relationnel Oracle (V8), mais permet de supporter des modèles de données orienté objet grâce à un mapping objet-vers-relationnel. En effet, Windchill intègre l'atelier Rational/Rose98 pour le développement du modèle métier (business model) en UML. Après avoir fait un mapping des classes du modèle métier UML sur les classes persistantes de Windchill, le SGDT permet de générer des tables Oracle associées aux divers éléments du modèle UML. Cette génération est assurée par un système de génération intégré à Windchill (cf. Figure 7.16) qui, partant des modèles UML sous Rose :

- génère d'abord des fichiers mData : ce sont des fichiers du modèle de données, indépendants du fournisseur. A Chaque package de UML, un fichier mData est généré. Ces fichiers constituent le points d'entrée pour la génération des objets fabriqués,
- les fichiers mData sont ensuite utilisés pour la génération du code Java. En plus du code Java, sont également générés des fichiers Info contenant les méta informations des classes nécessaires à l'environnement d'exécution. Ces fichiers Info sont par la suite utilisés pour générer des fichiers SQL contenant des directives pour la génération des tables Oracle pour les classes persistantes.

Nous soulignons que Windchill offre près de 500 classes dédiées au domaine des SIP. Le mapping des classes du modèle métier UML sur les classes persistantes de Windchill consiste à ajouter dans le modèle métier des relations structurelles vers les classes fondamentales de Windchill. Une façon de faire est de créer des spécialisations entre les classes du modèle de métier et les classes Windchill pour étendre ces dernières[167]. A titre d'exemple, les classes du modèle métier UML étendent généralement une des classes du package wt.enterprise de Windchill; les associations du modèle métier étendent des classes "Link" de Windchill et les attributs structurés dans le modèle métier étendent la classe ObjectMappable de Windchill.

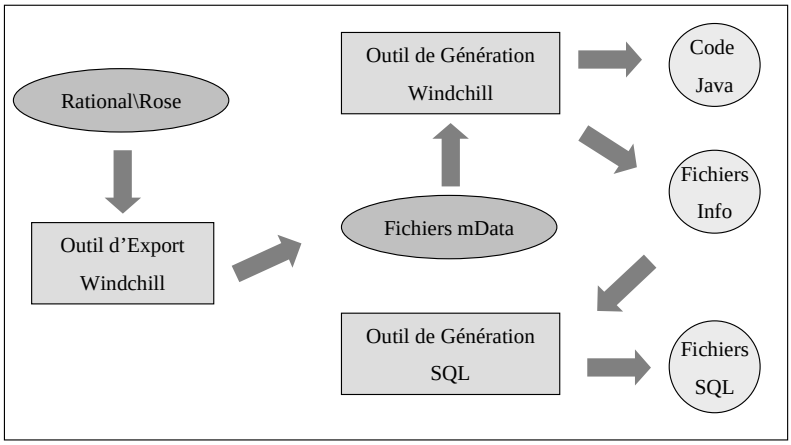


Figure 7.16 - Vue générale du système de génération de Windchill

3.4. Architecture de l'outil support proposé

En résumé, l'outil support de la démarche de spécification par réutilisation a l'architecture suivante :

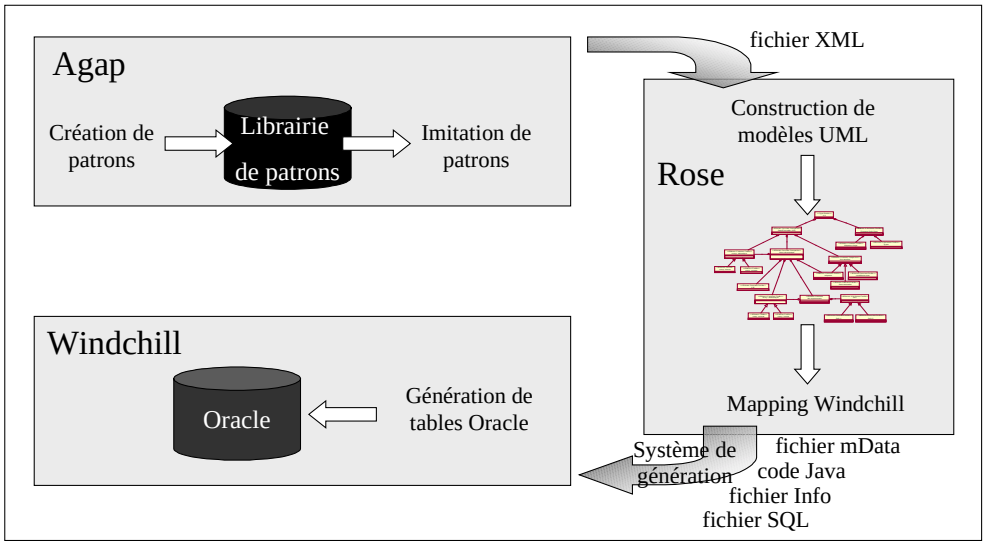


Figure 7.17 - Architecture de l'outil support proposé

[167] C'est ce qui correspond à la notion de regroupement de classes lors de l'opération d'intégration des imitations de patron, que nous avons présenté au début de la Section 2 du présent chapitre.

Dans cette architecture, une phase préalable à la réutilisation des patrons, consiste à créer les patrons du domaine étudié dans la librairie de patrons gérée dans AGAP. Ensuite, le concepteur du SIP, disposant d'un système de patrons dans AGAP va pouvoir sélectionner les patrons permettant de résoudre le problème qu'il traite. Puis, à la demande d'imitation des patrons sélectionnés, l'atelier AGAP génère un fichier XMI supportant le modèle UML de la solution des patrons à imiter. Ce fichier permet au concepteur du SIP de construire sous Rational/Rose les modèles UML définis par imitation. A ce stade, les modèles de spécification UML élaborés sont indépendants du SGDT dans lequel ils seront implantés. Ce sont des modèles d'analyse, qui ne sont pas définis en fonction de classes types proposés dans le SGDT cible.
Une phase préalable à l'intégration des modèles UML élaborés dans le SGDT cible consiste à effectuer un mapping entre le modèle UML et les classes types du SGDT considéré.
Ensuite, le modèle UML mappé peut être exporté. Dans le cas du SGDT Windchill, un système de génération permet de générer des tables Oracle à partir du modèle UML mappé sous Rose, après génération de code Java et de fichiers SQL.
Par défaut, l'atelier rational Rose permet de générer du code C++ et des fichiers SQL. A l'import de ces fichiers dans un SGDT, une manipulation des fichiers SQL est nécessaire pour créer les tables associées dans le SGBD utilisé par le SGDT. A ce titre, Windchill présente l'avantage d'assister cette phase de mapping des modèles UML en tables Oracle. C'est ce qui a motivé notre choix pour le SGDT Windchill dans l'outil support proposé.
Il est évident que le choix effectué des outils n'est fait que pour illustrer une architecture possible de l'outil support de la démarche de réutilisation de patrons. D'autres outils assurant des fonctionnalités analogues peuvent être considérés.

4. Conclusion

Ce chapitre a illustré d'une part l'expérimentation de la démarche de spécification de SIP par réutilisation de patrons que nous avons proposé et d'autre part l'outil logiciel pouvant supporter la démarche. L'expérimentation de la démarche a été menée en appliquant le catalogue de patrons sur des projets réels de développement d'applications SIP chez notre partenaire industriel. Trois applications ont pu être effectuées. Nous en avons décrit dans ce chapitre celle que nous avons jugé la plus complète. Cette application ne suffit certes pas à valider l'approche proposée. L'objectif était plutôt de confronter cette approche à des projets réels de développement et de vérifier ainsi les trois points suivants :

- la mise en œuvre de la démarche de spécification par réutilisation de patrons est facilement appréhendable par les concepteurs ;
- les patrons proposés répondent à des problèmes récurrents rencontrés dans les projets de développement de SIP sur le terrain industriel ;
- les modèles de spécification obtenus par réutilisation des patrons proposés sont conformes aux objectifs de développement.

Nous soulignons qu'au fur et à mesure de l'avancement de notre travail, nous avons essayé d'appliquer les patrons proposés aux cas industriels observés. Cela a permis de mettre au point les patrons développés. Tel est le cas pour les patrons d'analyse processus, d'abord appliqués au projet VEGA1-mécanique (ancien processus de gestion des modifications produit). La version mise au point de ces patrons a été ensuite appliquée au projet EvolProd (sur le

nouveau processus d'évolution de produits). Il est en de même pour les patrons de conception, d'abord appliqués au projet VEGA1-mécanique (sur le processus de gestion des modifications). La version mise au point de ces patrons a été ensuite appliquée au projet VEGA2-électronique (sur le processus de gestion des dossiers techniques des produits).
En ce qui concerne les patrons d'analyse produit, notre démarche a été légèrement différente. Les patrons produit ont été définis à l'issue de l'analyse de domaine basée en particulier sur l'étude des modèles existants dans l'application VEGA2-mécanique destinée à gérer les dossiers techniques des produits mécaniques. Les patrons ainsi définis par abstraction, ont été appliqués aux produits électroniques (projet VEGA2-électronique).

Les principaux apports de l'application des patrons dans le cadre du projet VEGA2-électronique peuvent être résumés par :
- la clarté de communication entre concepteurs et utilisateurs du SIP autour des besoins, grâce à l'utilisation des modèles UML, facilement accessibles par les utilisateurs et suffisamment formels pour structurer l'ensemble des besoins ;
- l'aide apportée au concepteur SIP pour aborder successivement les divers problèmes de spécification, en s'appuyant sur un enchaînement préconisé des patrons ;
- l'aide apportée aux intervenants pour exprimer avec précision leurs besoins (les objets du SIP) en les obligeant à déterminer exactement les responsabilités dans les processus métiers qu'ils souhaitent gérés, à fixer les objets manipulés à chaque étape, à classer les objets du domaine qu'il manipulent, etc. ;
- l'accélération des délais de spécification.

Au delà des effets directs et souhaités, la mise en œuvre de la démarche proposée a eu quelques effets de bord appréciables tels que :
- la mise en œuvre d'un arbre de décision pour le bureau d'étude électronique afin d'évaluer l'impact d'une évolution apportée à l'une des représentations (principales ou secondaires) sur les composants des produits électroniques correspondants et vice-versa (en terme de versions, révisions, corrections). Cet arbre de décision a été ensuite généralisé pour déterminer l'impact de n'importe quel type d'objet métier sur les autres objets métiers du domaine. Cet arbre a été décrit sous forme de patron dans [Gzara, 00b].
- le support au projet de réorganisation du processus de modification de produits mené à Schneider, en aidant le groupe de réflexion désigné à cet effet dans la structuration du processus et la définition des diverses responsabilités impliquées.

Le catalogue de patrons proposé reste toutefois à enrichir par d'autres patrons spécialisant davantage les problèmes traités. En terme d'implantation, et en l'absence de patrons d'implantation, l'apport de la réutilisation de patrons dans l'accélération des processus de développement ne peut être maîtrisé. L'observation de la phase d'implantation qui a suivi notre intervention dans la spécification de l'application VEGA2-électronique nous a permis d'estimer la difficulté de mise en œuvre de spécifications techniques par écart au SGDT utilisé. Le développement de patrons d'implantation, bien que coûteux et spécifiques à chacun des SGDT, devrait faciliter cette phase.
Enfin, l'outil logiciel proposé pour supporter la démarche proposée reste à expérimenter. Actuellement, l'atelier AGAP ne permet pas d'imiter les patrons créés dans l'atelier. Toutefois cette fonctionnalité est en cours de mise en place. Il serait opportun à la fin de ces développements de tester l'outil support proposé.

Conclusion Générale

Bilan

Cette thèse a présenté un cadre méthodologique pour l'ingénierie des systèmes d'information produit, adapté aux objectifs de recherche initiaux.
Les hypothèses formulées au début du travail et sur lesquelles s'est basée la définition de la problématique traitée étaient :

- le manque de *modèles formels* d'expression des besoins suffisamment compréhensibles pour assurer une communication efficace et non ambiguë entre les acteurs concernés lors de la spécification des SIP,
- l'absence de *continuum* de transformations cohérentes des différents modèles élaborées aux différentes phases d'ingénierie,
- la *lenteur* de développement des SIP.

Partant de ces hypothèses, les travaux présentés dans le cadre de ce mémoire ont eu pour objectif de mettre en place un cadre méthodologique pour l'ingénierie des SIP en abordant trois aspects :

- la définition de *modèles* de spécification de différents niveaux d'abstraction couvrant la complexité des informations gérées dans le SIP et favorisant l'échange d'information entre acteurs,
- la mise en place d'une *démarche* générale de conduite de projet SIP pour construire les différents modèles couvrant l'ensemble des phases du processus de développement des SIP (analyse, conception, réalisation) et assurant un continuum de transformations dans les spécifications. Cette démarche doit être flexible et adaptable au contexte de chaque organisation,
- l'*accélération* du processus de développement des SIP. L'objectif est de favoriser la capitalisation et la réutilisation de savoir et de savoir-faire expérimentés et mis en œuvre dans des projets antérieurs.

L'investigation dans la littérature récente des techniques de modélisation et des approches de réutilisation susceptibles de réaliser le cahier de charges ainsi posé a permis de fixer les éléments clés sur lesquels se base le cadre méthodologique proposé. Il s'agit alors de développer une démarche d'ingénierie de SIP basée sur la réutilisation de patrons et l'utilisation du langage UML.

L'introduction des patrons dans une démarche d'ingénierie de SI en général et de SIP en particulier nécessite de mettre en œuvre deux processus complémentaires :

- un processus *pour la réutilisation* est dédié à *l'ingénierie des patrons*. Il consiste à identifier, spécifier et organiser les patrons à utiliser durant la spécification des SIP.
- un processus *par la réutilisation* est dédié à *l'ingénierie des SIP*. Partant des spécifications utilisateurs, ce processus facilite la recherche, la sélection, l'adaptation et l'intégration de patrons pour spécifier et implanter le SIP. Ce processus implique une re-formulation du processus classique d'ingénierie de SIP pour intégrer de manière systématique la réutilisation de patrons.

Une démarche d'ingénierie de patrons est alors définie pour identifier et spécifier les patrons du domaine SIP. Un ensemble de mécanismes sont par ailleurs définis pour organiser et faciliter la réutilisation des patrons proposés.

Le catalogue de patrons dédié au domaine des SIP que nous avons proposé est constitué de trois types de patrons :

- des patrons d'analyse produit, pour représenter les objets métiers relatifs aux produits et gérés dans le SIP. Ils permettent ainsi de construire le modèle d'analyse du SIP ;
- des patrons d'analyse processus, pour analyser et représenter les processus métiers du SIP ;
- des patrons de conception, pour représenter les objets informatiques associés aux divers objets métiers identifiés à l'aide des deux types de patrons précédents. Ils permettent de construire le modèle de conception du SIP.

Ces patrons permettent de réutiliser des *fragments de modèles* représentant divers aspects dans le SIP mais également des *fragments de démarches*, définissant ainsi la démarche de spécification par réutilisation. La démarche globale d'ingénierie du SIP sera donc un "assemblage" de fragments de démarches, de même que les modèles de spécification du SIP sont obtenus par assemblage de fragments de modèles, et ce par réutilisation de divers patrons. Par ailleurs, le formalisme proposé pour représenter les patrons favorise la sélection et l'organisation des patrons. Il prend en compte également le caractère contextuel des activités d'ingénierie.

Ce catalogue de patrons a été validé en l'expérimentant dans des projets industriels.

En résumé, les principaux apports de notre proposition sont :

- La proposition d'un catalogue de patrons pour l'ingénierie des SIP qui capitalise les savoir et les savoir-faire mis en œuvre dans l'ingénierie des SIP, couvre la complexité des connaissances gérées dans les SIP et propose diverses structurations pour organiser ces connaissances et faciliter leur gestion.
- La contribution à la formalisation de la gestion des données techniques en proposant un référentiel pour le domaine. Ce référentiel délimite l'ensemble des concepts gérés dans les SIP, propose un cadre terminologique et une sémantique à ces concepts et identifie les problèmes de structuration et de gestion associés aux divers concepts. Il propose par ailleurs de résoudre certains de ces problèmes.
- La définition d'une démarche générale pour construire les patrons. Si les patrons commencent à devenir un sujet commun du domaine de l'ingénierie des SI, leur ingénierie pose actuellement de nombreux problèmes et la mise en œuvre de cette forme d'ingénierie constitue une problématique de recherche.
- La contribution à une meilleure formalisation du concept de patron, notamment en mettant en évidence deux aspects de capitalisation de connaissances dans les patrons : la capitalisation de savoir (modèles) et la capitalisation des savoir-faire (démarches).
- La contribution à la mise en place d'une nouvelle vision pour organiser les projets de spécification des systèmes d'information. Les approches classiques d'ingénierie de SI sont basées sur l'application de démarches et de modèles de spécification pre-établis, ne tenant pas compte de la variabilité entre les organisations. L'approche proposée ici est

contextuelle. Elle repose sur un ensemble patrons dédiés au domaine étudié, indiquant comment adapter les connaissances qu'ils capitalisent, selon le contexte dans lequel elles sont utilisées.

Perspectives

Ce travail ouvre la voie à notre sens vers diverses perspectives de recherche qui se situent sur deux plans : un plan d'approfondissement de la recherche réalisée et un plan d'élargissement du domaine de la recherche.
Pour ce qui est de l'approfondissement du travail proposé, il serait intéressant dans un premier temps de compléter le catalogue de patrons proposé par d'autres patrons raffinant les patrons actuels. L'application du catalogue actuel sur des projets industriels a en particulier mis en évidence la difficulté à spécifier d'une façon détaillée la dynamique associée aux divers objets métiers identifiés. Par ailleurs, la définition de patrons logiciels SIP associés aux patrons métiers que nous avons proposés constitue un atout pour le cadre méthodologique proposé. Soulignons toutefois que la définition de ce type de patrons est extrêmement liée aux SGDT ; des progiciels divers et en constante évolution. Dans un second temps, et en ce qui concerne l'expérimentation de la démarche proposée, il serait intéressant d'appliquer les patrons proposés dans d'autres projets industriels de spécification de SIP appartenants à des métiers et des secteurs d'activité variés. Dans ce cadre, une expérimentation de l'outil support proposé permettra de le valider et de tester ses capacités et ses limites à supporter la démarche proposée.

Les perspectives de notre travail s'articulent autour de deux axes:

1. **Les démarches d'ingénierie des SI à base de patrons** : il serait intéressant de "formaliser" les patrons (actuellement, la majorité des rubriques sont spécifiées en langue naturelle) afin de mieux les gérer et les organiser. Par ailleurs, l'ingénierie des patrons pour la réutilisation est un sujet de recherche essentiel. Nous avons proposé une première démarche pour identifier et spécifier les patrons, qu'il serait intéressant d'appliquer à l'ingénierie d'autres catalogues de patrons. La formalisation des processus d'ingénierie de SI par réutilisation de patron est également un sujet de recherche intéressant. L'expérimentation du catalogue de patrons SIP sur une application industrielle a mis en évidence le besoin pour formaliser les démarches de développement par réutilisation et les doter de mécanismes favorisant l'intégration systématique des patrons en facilitant la recherche, la sélection et surtout l'adaptation de patrons. Par ailleurs, l'approche à base de réutilisation que nous avons proposé met l'accent sur la réutilisation aux phases amonts du processus d'ingénierie des SIP. Il serait opportun de définir et de mettre en place des formes de réutilisation aux phases avales d'ingénierie, notamment au déploiement des applications SIP pour pouvoir réutiliser des architectures de déploiement, de formation, etc.. Enfin, il serait intéressant d'étudier l'extension de l'utilisation des patrons pour la résolution de problèmes récurrents dans divers domaines d'ingénierie. Ils permettent en effet d'organiser hiérarchiquement, fonctionnellement et contextuellement les problèmes et les manières de les résoudre.

2. **La rationalisation de la gestion des données techniques du produit** : l'analyse de domaine que nous avons mené a été de nature exploratoire et a permis de mettre en évidence l'existence de plusieurs problèmes liés à la structuration des données techniques et par suite à la rationalisation de leur gestion. Nous citons en particulier les problèmes de gestion des configurations de produits et de la gestion des évolutions des données produit. Certaines de ces perspectives sont déjà mises en œuvre. Elles ont donné lieu à deux sujets de thèse au sein de l'équipe systèmes d'information techniques du laboratoire Gilco. Le premier sujet porte sur la gestion de la diversité produit en rationalisant la gestion des données de configuration des produits génériques et des types de produits associés. Le deuxième sujet porte principalement sur la gestion des évolutions de produit. En matière de processus SIP, la gestion du workflow d'ingénierie fait émerger plusieurs questionnements de recherche pour mettre en œuvre notamment des workflow flexibles adaptés à des processus de forte variabilité tels que ceux de développement de produits. Enfin, la recherche que nous avons menée s'est focalisée sur les SIP en offrant un référentiel pour ce domaine et formalisant l'ensemble des connaissances qu'ils couvrent. Ce travail pose les jalons d'un projet plus ambitieux lié à l'organisation de l'ensemble des systèmes d'information de l'entreprise, dédiés à la productique. Notre souhait est d'étudier les liens, tant au niveau conceptuel que technique, entre ce système essentiel dans l'entreprise avec divers autres systèmes d'information spécifiques à certaines fonctions de l'entreprise. La considération des aspects normatifs associés à ce problème est essentielle dans cette étude. Dans le cadre du projet régional OSCAR (Organisation de Simulations pour la Capitalisation et la Réutilisation), nous contribuons, dans le cadre d'un stage post-doctoral à l'Ecole Polytechnique de Montréal (Canada), à l'étude et la formalisation des liens entre le SIP et deux systèmes d'information spécifiques dédiés d'une part aux simulations de produit (maquette numérique) et d'autre part aux analyses fonctionnelles de produit. Cette étude est essentiellement de nature conceptuelle. Elle s'inscrit dans l'objectif d'une intégration des applicatifs spécifiques dans le cadre du SIP global de l'entreprise.

A

[Aalst, 99] W.M.P van der Aalst, *Flexible Workflow Management Systems : An Approach Based on Generic Process Models,* Proceedings of the 10th International Conference on Database and Expert Systems Applications (DEXA'99), volume 1677 of Lecture Notes in Computer Science, pages 186-195. Springer-Verlag, Berlin, 1999.

[Afnor, 82] Afnor, *NF X60-012 - Termes et définitions des éléments constitutifs et de leurs approvisionnements pour les biens durables,* 1982.

[Afnor, 85] Afnor, *NF X11-500 - Tamis et tamisage – Terminologie,* 1985.

[Afnor, 90] Afnor, *NF X50-150 - Analyse de la valeur, caractéristiques fondamentales,* 1990.

[AIT, 93] *Advanced Information Technology for European Manufacturing Industry*. http://www.ait.org.uk/index.htm

[AIT, 97a] Projet E.J. WAITE, *AIT - Advanced Information Technology for Design and Manufacture- WP2 definition,* 1997.

[AIT, 97b] Projet E.J. WAITE, *AIT - Advanced Information Technology for Design and Manufacture- WP 4.1 Terminolgy,* 1997.

[AIT, 97c] Projet E.J. WAITE, *AIT - Advanced Information Technology for Design and Manufacture- CEN/TC 310 definition,* 1997.

[AIT, 97d] *EP22148: Integration Platform for AIT*. http://www.ait.org.uk/projects/aitip2/leaflet.htm, 1997.

[Alexander, 77] Alexander C., Ishikawa S. et al., *A Pattern Language,* Oxford University Press, new York, NY, 1977.

[Ambler, 98] Ambler S.W., *Process Patterns : building Large Scale Systems using Object Technology,* SIGS Books, Cambridge University Press, december 1998.

[ANSI, 75] ANSI, *Interim Report ANSI/X3/SPARC Study Group on Data Base Management Systems,* ACM SIGMOD, Vol 7, N°2, 1975.

[Arasti, 99] Arasti M. R., *Aide à l'élaboration de stratégies technologiques cohérentes avec la stratégie globale de l'entreprise,* thèse de doctorat de l'Institut National Polytechnique de Grenoble, 1999.

[Avison, 97] Avison D., Shah H., *The information Systems Development Life Cycle : A first Course in Information Systems,* McGraw-Hill, Berkshire, 1997.

B

[Beck, 87] Beck K., Cunnimgham W., *Using Pattern Languages for Object-Oriented Programs,* Proceedings of ECOOP'94, 1994.

[Belsnes, 99] Belsnes D., Bergan M., Devos M. Gronmo R., Hjelle B., Solberg A., *POSC/CAESAR distributed application development framework,* Product Data Technology Europe'99 8the Symposium, 13-16 April 1999, Stavanger, Norway.

[Benchimol, 93] G. Benchimol, *L'entreprise étendue,* Ed. Hermes, Paris, 1993.

[Bernard, 99] Bernard A., *Modèles de produit et de processus*, Groupement Primeca - Université d'automne "Modélisation des processus de conception", 20-22 octobre 1999, Nancy, France.

[Blanchard, 98] Blanchard B.S., *System Engineering Management*, Ed. John Wiley&Sons, New York, 1998, 480 pages.

[BNAE, 90] BNAE - Bureau de Normalisation de l'Aéronautique et de l'Espace, RG Aéro 000 40 - *Recommandation Générale pour la Spécification de Management de Programme*, 47 pages, 1991.

[BNAE, 92] BNAE, *NF L00 007 B - Industrie aéronautique et spatiale - Vocabulaire – Termes Généraux*, 1992.

[BNAE, 95] BNAE, *XP L 00 022 - Série aérospatiale - Hiérarchisation des caractéristiques de Définition*, 1995.

[Bodington, 00] Bodington R., Lämmer L., *Integrating the Enterprise using the PDM schema and the PDM Enablers*, Product Data Technology Europe 2000 9th Symposium, 2-5 May 2000, Noordwijk, The Netherlands.

[Bonfatti, 99] Bonfatti F., Monari P.D., *An SME-targeted Process Model for Non-conventional Processes*, Product Data Technology Europe'99 8th symposium, 13-16 April 1999, Stavanger, Norway.

[Boulet, 95] Boulet C., Ballieu J., *L'Analyse de la Valeur*, Ed. Afnor, 1995.

[Bounaas, 97] Bounaas F., *vers une méthode de conduite de projet*, rapport de stage année spéciale, ENSGI - INPG, Grenoble, juin 1997.

[Bourdichon, 94] P. Bourdichon, *L'ingénieire simultanée et la gestion d'informations*, Ed. Hermès, Paris, 1994.

[Brude, 98] Brude M.B., *La gestion de Données Techniques*, PCO technologies, document interne, 1998.

C

[Carré, 89] B. Carré, *Méthodologie orientée objet pour la représentation de connaissances : concept de point de vue, de représentation multiple et évolution d'objet*, Thèse de l'Université des Sciences et Techniques de Lille Landres Artois, 1989.

[Cauvet, 00a] Cauvet C., Rieu D., Front-Conte A., Ramadour P., *Réutilisation dans l'ingénierie des systèmes d'information*, dans l'ouvrage collectif Conception des Systèmes d'Information, Rosenthal-Sabroux C.et Cauvet C., Hermès, 2000.

[Cauvet, 00b] C. Cauvet, L. Gzara, P. Ramadour, D. Rieu. " Ingénierie des systèmes d'information produit : une approche méthodologique centrée réutilisation de patrons". A paraître dans la revue *L'Objet*, Vol 6 - n°4. Ed. Hermès.

[Chabre, 97] Chabre-Peccoud M., Freire-Junior J.C., Front A., Giraudin J-P et Guetari R., *Ingénierie Objet - Concepts, techniques et méthodes, Chapitre 1 - Objets, Langages et méthodes*, InterEditions, 1997.

[Chambolle, 99] Chambolle F., *Un modèle produit piloté par les processus d'élaboration : application au secteur automobile dans l'environnement STEP*, Thèse de Doctorat de l'Ecole Centrale Paris, 1999.

[Chen, 98] Chen Y-M, Tsao T-H, *A structured methodology for implementing engineering data management*, Robotics and Computer-Integrated Manufacturing, Vol 14 (1998), pp 275-296.

[CIMdata, 95] CIMdata Corp., *CIMdata Glossary*, (http://www.cimdata.com/), 1995.

[CIMdata, 97] CIMdata Inc., *Product Data Management : the definition. An introduction to concepts, Benefits, and Terminology*, (http://www.cimdata.com/), 1997.

[CIMdata, 00] CIMdata, *CIMdata Reports cPDm Market Grows 26% to Reach $1.76 Billion in 1999*, http://www.cimdata.com/PR000307B.htm, 2000.

[CIR, 99] *Benefits of EDM.PDM in Manufacturing Industry*, Engineering Data Management Newsletter, December 1999.

[Coad, 92] Coad P., *Object-Oriented Patterns*, *Communications of the ACM*, Vol 35, N°9, September 1992.

[Coad, 96] Coad P., *Object Models – Strategies, Patterns and Applications*, Yourdon Press Computing Series, 1996.

[Condor, 99] CONDOR Project Final Report, University of Salfrod, March 1999.

[Coplien, 92] Coplien J.O., *Advanced C++ Programming Styles and Idioms*, Addison-Wesley, 1992.

[Cutting-Decelle, 00] Cutting-Decelle A.F., Michel J.J,. *A Standardised Data Model for Manufacturing Management: the ISO 15531 MANDATE standard*, 7th ISPE International Conference On Concurrent Engineering, Lyon, France, july 17-20, 2000.

D

[Dahchour, 98] Dahchour M., *Formalizing Materialization Using a Meta-Class Approach*, in Proceedings of CAISE'98.

[Desclaux, 90] Desclaux C., Guthauser A., Pécoud T., Pomian J, Storr P. and Thierry A., *Design & maintenance process Rationale under MACS*, Conf. Le Génie Logiciel et ses applications, Toulouse, France, décembre 1990.

[Djeraba, 93] Djeraba C., *Quelques liens sémantiques dans un Système à Base de Connaissances*, Thèse de Doctorat de l'Université Claude Bernard de Lyon – Lyon 1. 143 pages. 1993.

[D'Souza, 98] d'Souza F., Wills A.C., *Objects, Components and Frameworks with UML : The CATALYSIS Approach*, Eds Addison-Wesley, 1998.

[Duong, 00] Duong P.Q., *Spécification d'une démarche de développement à base de patterns*, Mémoire de fin d'études de l'Institut de la Francophonie pour l'Informatique, Vietnam - Laboratoire LSR/INPG, France. Septembre 2000.

E

[EDS, 98] EDS, *EDS terminology* – Definitions, (http://www.eds.com/), 1998.

[Ehrler, 99] Ehrler A., Kunze H., Maier M., *Development of configurable semantic mappers*, Product Data Technology Europe'99 8th symposium, 13-16 April 1999, Stavanger, Norway.

[El Mhamedi, 95] El Mhamedi A., Lerch C., Sonntag M., *Modélisation des activités et des processus des systèmes de production : une approche multidisciplianire*, Congrés International de génie Industriel, Montréal, Canada, 18-20 octobre 1995.

[El Mhamedi, 97] El Mhamedi A., Lerch C., Marier S., Sonntag M., Vernadat F., *Intégration des activités non structurées das la modélisation des systèmes de production ACNOS*, Rapport final du projet ACNOS – action incitative du DSPT 8 en productique du MENESR. 1997.

F

[Féru, 98] Féru F., Viel C., Echanger avec le protocole d'application 203 de STEP, Echange et partage de données CAO et GDT, Association GOSET, 1998. ISBN 2-9513382-0-1.

[Fowler, 97] Fowler M., Analysis Patterns - reusable Object Models, Addison-Wesley, Reading MA, 1997.

[Fowler, 97] Fowler M., *Analysis Patterns - reusable Object Models*, Addison-Wesley, 1997.

[Frakes, 94] Frakes W.B., Pole T.P., *An Empirical Study of Representation Methods for Resuable Software Components*, IEEE Transactions on Software Engineering, Vol 20, N°8, August 1994.

[Front, 97] Front A., *Développement de systèmes d'information à l'aide de patrons - Applications aux bases de données actives*, Thèse de Doctorat, spécialité Informatique, UJF - Grenoble 1, France, décembre 1997.

[Foulard, 94] Foulard C., *La modélisation en entrteprise - CIMOSA et ingénierie simultanée*, Ed. Hermès, Paris, 1994.

G

[Gama, 97] Groupe GAMA, *Modélisation par entités*, Primeca - 5ème Colloque sur le Conception Mécanique Intégrée, 2-4avril 1997, La Plagne, France..

[Gamma, 95] Gamma E., Helm R., Johnson R., Vlissides J., Design Patterns, Elements of reusable Object-Oriented Software, Addison-Wesley Publishing Company, 1995.

[Ghodous, 95] Ghodous P., Vandorpe D., *Standardization of product data models using STEP*, INRIA/IEEE Conference on Emerging Technologies and Factory Automation ETFA'95, Paris, France.

[Giraudin, 00] Giraudin J-P, Chabre M., Rieu D., Saint-Marcel C., *Modèles de spécification pour l'ingénierie des SI*, dans l'ouvrage collectif Conception des SI, C.Rosenthal-Sabroux et C. Cauvet, Hermès, 2000.

[Gomaa, 93] Gomaa H., *A reuse-oriented approah for structuring and configuring distributed applications*, Software Engineering Journal, March 1993.

[Grosz, 97] Grosz G., Rolland C., Schwer S. et al., *"Modelling and Engineering the Requirements Engineering Process : An overview of the NATURE Approach"*, Requirement Engineering (1997) 2:115-131, Springer-Verlag, 1997.

[Guffond, 98] Guffond J-L, Leconte G., *L'activité modification de produit à ABT/Schneider Electric: procédure, acteurs, régulation*, rapport interne CRISTO, juin 1998.

[Guffond, 99] Guffond J-L, Leconte G., *Dynamisation du processus d'évolution des produits à ABT/Schneider Electric : création et réglage d'un nouveau dispositif de travail*, rapport interne CRISTO, janvier 1999.

[Gzara, 99] L. Gzara, D. Rieu, M. Tollenaere. "Un Référentiel Générique De Données Techniques à Des Fins De Réutilisation En Ingénierie Des Systèmes D'information Produit". Actes du *3e Congrès International de Génie Industriel*, 26-28 Mai 1999 – Montréal, Canada.

[Gzara, 00-a] L. Gzara, D. Rieu, M. Tollenaere. "Product Information Systems Engineering : an approach by reuse of patterns". Actes de *Product Data Technology conference PDT Europe 2K*, 2-5 mai 2000, Noordwijk, The Netherlands.

[Gzara, 00b] Gzara L., Rieu D., Tollenaere M., *Towards a Dynamic Reference Framework for Product Information Systems: Focus on Engineering Change Process*, proc. of the *3rd International Conference on Integrated Design and Manufacturing in Mechanical Engineering*, 16-19 mai 2000, Montréal, Canada.

[Gzara, 00c] L. Gzara, D. Rieu, M. Tollenaere. "An Approach for Building Product Models by Reuse of Patterns". Actes du *7th ISPE International Conference On Concurrent Engineering CE2000*. 17-20 juillet 2000, Lyon, France.

[Gzara, 00d] Gzara L., Rieu D., Tollenaere M.. "Patterns Engineering For Reuse at Product Information Systems Development". *Requirements Engineering Journal*, Vol 5 - N°3, 2000, Ed. Springer-Verlag.

[Gzara, 00e] Gzara L., Rieu D., Tollenaere M.. "Product Information Systems Engineering : An Approach For Building Product Models By Reuse Of Patterns". A paraître dans *Robotics and Computer-Integrated-Manufacturing Journal*. Ed. Pergamon.

H

[Habrias, 88] Habrias H., *Le modèle relationnel binaire - méthode I.A. (NIAM)*, Ed. Eyrolles, Paris, 1988.

[Hamaidi, 97] Hamaidi L., Vernadat F.B., *Une approche de modélisation des processus opérationnels d'entreprise à des fins de réorganisation*, 2ème Congres International Franco-Québécois de Genie Industriel, Albi, France, 3-5 septembre 1997.

[Hameri, 98] Hameri A-P., Nihtilä J., *Product data management - exploratory study on state-of-the-art in one-of-a-kind indutsry*, Computers in Industry, Vol 35 (1998), pp195-206, Ed. Elsevier.

[Hammer, 93] Hammer M., Champy J., *Reengineering the corporation : a manifesto for business revolution*, Harper Collins Publishers, Inc. Ed., New-York . Traducion française : « le reenigineering », Paris, Dunod, 1993.

[Harani, 97] Harani Y., *Une approche Multi-Modèles pour la capitalisation des connaissances dans le domaine de la conception*, Thèse de doctorat de l'Institut National Polytechnique de Grenoble, 1997.

[Harrington, 91] Harrington H.J., *Business Process Improvement - The Breakthrough Startaegy for Total Quality, Productivity, and Competitiveness*, Ed. McGraw-Hill Inc., 1991

[Hassine, 00] Hassine I., Formalisation, Instanciation et composition des Patterns, Rapport de DEA Système d'Information, Université Pierre Mendès France, Septembre 2000.

[Hay, 99] Hay D.C., *A Comparison of Data Modeling Techniques*, Essential Startegies Inc, 1999. Disponible sur le site internet : http://www.essentialstrategies.com/

[Hensinger, 00] Hensinger C., Reichert M., Bauer Th., Strzeletz Th., Dadam P., *ADEPT workflow - Advanced Workflow Technology for the Efficient Support of Adaptive, Enterprise-wide Processes*, VII. Conference on Extending Database Technology, Demotrack, Konstanz, Germany, March 2000.

[Hess, 90] Hess J., Cohen S., Holibaugh B., Kyang K., Peterson A.S., Novak W., Carroll P., *A Domain Analysis Bibliography*, Software Engineering Institute - Special report, CMU/SEI-90-SR-3, 1990.

[Hollingsworth, 95] Hollingsworth D., *Workflow Management Coalition - The Workflow Refrence Model*, WfMC, Brussel, 1995.

[Huang, 00] Huang G.Q., Yee W.Y., Mak K.L., Reengineering The Engineering Change Management Process, 7th ISPE International Conference On Concurrent Engineering, Lyon, France, July 17-20, 2000.

I

[IGES, 80] IGES, *Initial Graphics Exchange Specification : ANSI Y 14.26M*, ANSI - American National Standard Institute, USA, 1980.

[IPDMUG, 97] International Product Data Management Users Group, *IPDMUG Glossary*, (http://www.pdmic.com/IPDMUG/members/glossary1.shtml) - mise à jour du 19/08/97.

[ISO, 94a] ISO (International Organisation for Standardization), *ISO 10303-1 : 1994 Industrial Automation Systems and integration – Product data representation and exchange – Part1 : Overview and fundamental principles*, Geneva, 1994.

[ISO, 94b] ISO, *ISO 10303-21 - Industrial Automation Systems and Integration - Product Data Representation and Exchange - Part 21 : Implementation Methods : Standard data Access Interface*, ISO - International Organisation for Standardization, 1998.

[ISO, 94c] ISO, *ISO 10303-203 Industrial Automation Systems and integration – Product data representation and exchange – Part 203 : Application Protocol : Configuration Control Design, ISO*, Geneva, 1994.

[ISO, 97] ISO/Afnor, *Les normes ISO 9000 pour les PME. Recommandations de l'ISO/TC 176*, Ed. AFNOR, (1997).

[ISO, 98a] ISO, *ISO 13584 - Industrial automation systems and integration -- Parts library*, ISO - International Organisation for Standardization, 1998.

[ISO, 98a] ISO, *ISO 13584 - Industrial automation systems and integration -- Parts library*, ISO - International Organisation for Standardization, 1998.

[ISO, 98b] ISO, *ISO 10303-22 DIS - Industrial Automation Systems and Integration - Product Data Representation and Exchange - Part 22 : Implementation Methods : Clear Text Encoding of the Exchange Structure*, ISO - International Organisation for Standardization, 1994.

J

[Jacobson, 92] Jacobson I., Christerson M., Jonson P., Ôvergaard G., *Object Oriented Software Engineering : A use case driven approach*, Addison-Wesley, 1992.

[Jagou, 93] Jagou P., *Concurrent Engineering : La maîtrise des coûts, des délais et de la qualité*, Ed. Hermès, Paris, 1993.

[Jensen, 96] Jensen P. B,. *Guide d'interprétation des normes ISO 9000*, Ed. AFNOR, Paris, 1996.

[John, 99] John U., Geske U., *Reconfiguration of Technical Products Using ConBacon*, 16th National Conference on Artificial Intelligence AAAI'99, Workshop on Configuration, July 18-22 1999, Orlando, Florida.

[Johnson, 92] Johnson R.E., *Documenting frameworks using patterns*, in Proceedings of OOPSLA'92, ACM Press. 1992.

K

[Kang, 90] Kang K.C., Cohen S.G., Hess J.A. et al., *Feature-Oriented Domain Analysis (FODA) Feasibility Study*, Carnegie-Mellon University - Software Engineering Institute - Technical Report AD-A235 785, 1990

[Katz, 90] Katz R.H., *Toward a Unified Framework for Version Modeling in Engineering Databases*, ACM Computing Surveys, Vol. 22, No. 4, 1990. pp. 375-408.

[Kerr, 99] Kerr M., Rollo T., Bodington R., *Experience in the use of STEP for data exchange between PDM systems*, Product data Technology Europe 1999 8th Symposium, 13-16 April 1999, Stavanger, Norway.

[Kettani, 98] Kettani N., Mignet D. Paré P., Rosenthal-Sabroux C., *De Merise à UML*, Ed. Eyrolles, Paris, 1998.

[Kramer, 92] Kramer T.R., Palmer M.E., Bernard Feeney A., *Issues and Recommandations for a STEP Application Protocol Framework*, NIST, 1992.

[Krause, 90] Krause F.L., Ulbrich A., Vosgerau F.H., *Feature-based approach for the integration of design and process planning systems*, In proceedings of the 23rd International Symposium on Automative Technology and automation, 1990.

L

[Lämmer, 00] Lämmer L., Machner B., *Standards as enabler for PDM in virtual enterprises*, Product Data Technology Europe 2000 9th Symposium, 2-5 May 2000, Noordwijk, The Netherlands.

[Lea, 99] Lea D., *Patterns Discussion FAQ*, site internet : http://g.aswego.edu/.

[Limam, 99] Limam S., *Contribution à la modélisation et à la simulation des systèmes de production de services : proposition d'une méthode basée processus, UML et réseaux*

de Petri objets, Thèse de Doctorat de l'Institut National Polytechnique de Grenoble, 1999.

[Lin, 96] Lin J., M.S. Fox, Bilgic T., *A requirement Ontology for Engineering Design*, Concurrent Engineering: Research and Applications, Vol 4(3), September 1996, pp279-291.

[Lorino, 97] Lorino P., *Méthodes et pratiques de la performance – le guide du pilotage*, Les Editions d'organisation. Paris, 1997.

M

[Maiden, 95], Maiden N., Sutcliffe A., Taylor C., Till D., A Set of Formal Problem Abstractions for Reuse During Requirements Engineering, Ingénierie des Systèmes d'Informations. Volume 2(6), Special Issues on Requirements Engineering, pp. 679-698. 1995.

[Manhes, 98] Manhes S., *Les patterns métiers : extraction dans l'existant logiciel*, rapport de DEA, Institut de Recherche en Informatique de Nantes, septembre 1998.

[Maret, 96] Maret P., Poullet L., Pinon J.M., *Des modèles conceptuels pour capitaliser la connaissance au sein d'une organisation*, revue ISI, volume 4, n°4, 1996, pp. 491-540.

[Martin, 85] Martin J., *Manifeste pour un système d'information*, Traduction Française, Les Editions d'Organisation, 1985.

[Maurino, 93] Maurino M, *La gestion des données techniques – Technologie du concurrent engineering*, Ed. Masson, Paris, 1993.

[Megarsti, 99] Megarsti R., Ayadi K., Cauvin, A., Kieffer J-P, *L'activité : vers un concept fédérateur pour l'analyse de la conduite des systèmes de production et des processus de conception de produits*, 3ème Congrés International de Génie Industriel, 26-29 mai 1999, Montréal, Canada.

[Mentzas, 93] Mentzas G.N., *Coordination of Joint Tasks in Organisational Processes*, Journal of Information Technology, Vol 8, 1993, pp 139-150.

[Meszaros, 00] Meszaros G. and Doble J., *"A pattern language for pattern writing"*. (http://hillside.net/patterns/Writing/pattern_index.html).

[Morel, 93] Morel J-M, Faget J., *The REBOOT Environment*, IEEE, 1993.

[Morris, 00] Morris, K.C., *Improving PDM Testability through Standards Harmonization*, Product Data Technology Europe 2000 9th Symposium, 2-5 May 2000, Noordwijk, The Netherlands.

[Muller, 97] Muller P.A., *Modélisation objet avec UML*, Ed. Eyrolles, 1997.

O

[Olcoz, 99] Olcoz S., Ruiz F., Gutiérrez A., *Design Reuse Means Improving Key Business Aspects*, IEEE, 1999.

[OMG, 97] Object Management Group, *The Unified Modeling Language*, Release 1.1, Reference Manual, 1997.

[OMG, 98a] *OMG Manufacturing Domain Task Force : PDM Enablers revised Submission* (janvier 1998). http://www.omg.org/homepages/mfg/mfgppepdm.htm

[OMG, 98b] OMG - *The Common Object Request Broker : Architecture and Specification*, revision 2.2, 1998. http://www.omg.org.

[OMG, 99a] Object Management Group, *Meta Object Facility (MOF) Specification,* Version 1.3RTF, September 1999, disponible sur : http://cgi.omg.org/cgi-bin/doc?ad/99-09-04

[OMG, 99b] *OMG Manufacturing Domain Task Force : STEP and OMG Product Data Management Specifications : A Guide for Decision Makers*, (octobre 1999). http://www.omg.org/homepages/mfg/mfgppepdm.htm

[OPAL, 96] *EP 20377: Integrated Information and Process Management*. http://www.ait.org.uk/projects/projects.htm#OPAL

[Ouali, 97] Ouali M-S., El Mhamedi A., Binder Z., *La modélisation des Systèmes de Production Intégrant les Concepts de Structures, d'Activités et de Flux*, 2ème Congrés International Franco-Québecois de Génie Industriel, Albi, France, 1997.

[Oussalah, 97a] Oussalah C. et al., *Ingénierie objet – concepts et techniques*, Ed. Masson, Paris, 1997.

[Oussalah, 97b] Oussalah C., Puig V., Objets et Contraintes, chapitre dans l'ouvrage collectif "Ingénierie Objet - Concepts et techniques" coordonné par C. Oussalah, InterEditions, 1997.

[Ouazzani, 97] Ouazzani A., Bernard A., Bocquet J-C., *Design Process Management System : SAGEP, International Conference on Engineering Design*, ICED 97, Volume 1, pp221-226, Tampere, August 19-21, 1997.

P

[PDMIC, 98] Product Data Management Information Center, *Glossary of Product Data Mangement Related Terms*, (http://www.pdmic.com/glossary/index.html) - mise à jour du 2/11/98.

[PDM Schema, 99] version 1.1 de la spécification pré-normative de PDM Schema, disponible sur le site : http://www.pdm-if.org/pdm_schema.

[Pels, 00] Pels H.J., Helms R.W., Mandemaker T.H., *RapidPDM : Faster implementation of PDM*, PDT Europe 2000, 2-5 May 2000, Noordwijk, Netherlands.

[Pernelle, 00] Pernelle P., Theroude F., Courtois A., *Mise en œuvre et utilisation des systèmes de gestion de données techniques dans les petites et moyennes entreprises*, Proc. IDMME'2000, Montréal, 16-19 mai 2000, Presses Internationales Polytechnique.

[Portella, 00] Portella J., *Collaborative Management of the Product Definition Lifecycle for the 21et Century*, Actes du Product Data Technology Europa 2000 9th Symposium, 2-5 May 2000, ESTEC, Noordwijk, Netherlands.

[Portland, 00] About the Portland Form. (http://c2.com/ppr/about/portland.html).

[Poulin, 93] Poulin J.S., Caruso J.M., A reuse Metrics and Return On Investment Model, IEEE, 1993.

[Pourcel, 95] Pourcel C., Gourc D., Jia T., *Apport à la modélisation du système de production*, Actes de la 2ème Conférence Internationale sur l'Automatisation Industrielle, Nancy, 1995.

R

[Randoing, 95] Randoing J-M, *Les SGDT*, Ed. Hermès, Paris, 1995.

[Ramachnandran, 99] Ramachnandran S., Gil Y., *Knowledge Acquisition for Configuration Tasks : The EXPECT Approach,* 16th National Conference on Artificial Intelligence AAAI'99, Workshop on Configuration, July 18-22 1999, Orlando, Florida.

[Rieu, 97] Rieu D., Bounaas F., Morat P., Tamzalit D., *La Méta-circularité au service de la méta-modélisation,* Congrés INFORSID'97, Toulouse, 10-13 juin 1997, pp.577-599.

[Rieu, 99a] Rieu D., Giraudin J.P., Saint-Marcel C., *Réutilisation et patrons d'ingénierie,* Chapitre dans l'ouvrage collectif « Génie Objet – analyse et conception de l'évolution d'objets ». C. Oussalah, Ed Hermes, 1999.

[Rieu, 99b] Rieu D., *MERISE 2 - concepts de base, démarche et modèles,* Support de cours, IUT2-Université Pierre Mendès France, 1999.

[Rieu, 99c] Rieu, D., *Ingénierie des Systèmes d'Information,* Mémoire d'Habilitation à Diriger les Recherches, Institut National Polytechnique de Grenoble, 1999.

[Rieu, 99d] Rieu D., Giraudin J.P., Saint-Marcel C., Front-Conte A., *Des opérations et des relations pour les patrons de conception,* Actes d'Inforsid 1999, 1-4 juin 1999, Lagarde, France.

[RIS, 98] *RiseStep EP 20459: Enterprise Wide Standard Access to STEP Distributed Databases*. Final Demonstration (mars 1998).

http://www.ait.org.uk/projects/risestep/risestep.html

[Rivers, 00] Rivers Moore D., *An XML Update,* Prodcut Data Technology 2000 9th symposium, 2-5 may, 2000, Noordwijk, The Netherlands.

[Robert, 93] *Le nouveau petit Robert 1, Dictionnaire de la langue française*, Ed. Dictionnaires le Robert, Paris, 1993.

[Rolland, 88] Rolland C., Foucault O. et Benci G., *Conception des systèmes d'information - la méthode REMORA,* Ed. Eyrolles, 1988.

[Rolland, 93] Rolland C., *Adapter les méthodes à l'Objet : Challenges et Embûches,* Journées Méthodes d'Analyse et de Conception Orientées Objet des Systèmes d'Information, AFCET, Paris, Novembre 1993.

[Rolland, 96] Rolland C., *L'ingénierie des processus de développement de systèmes : un cadre de référence,* Revue Ingénierie des Systèmes d'Information, Ed. Hermès, Vol 4, n°6, 1996.

[Rolland, 98a] Rolland C., *A comprehensive view of Process Engineering*, Proc. of the 10th International Conference CAISE 98, Pise, Italie, juin 1998.

[Rolland, 98b] Rolland C., Loucopoulos P., Grosz G., Nurcan S., *A framework for Generic Patterns dedicated to the Management of Change in the Electricity Supply Industry,* In the 9th International Conference on Database and Expert Systems Applications (DEXA'98), 20-24 August 1998, Vienne, Austria.

[Rosenman, 96] Rosenman M. A., Gero J. S., *Modelling multiple views of design objects in a collaborative CAD environment*, CAD, Vol 28, No 3, pp. 193- 205, Elsevier science Ltd, 1996.

[Rouibah, 00] Rouibah K., Caskey K., *An engineering workflow system for the management of engineering processes across company border,* 7th ISPE International Conference On Concurrent Engineering, Lyon, France, july 17-20, 2000.

[Rumbaugh, 95] Rumbaugh J., Blaha M., Premerlani W., OMT 1 - Modélisation et Conception Orientées Objet, Traduction par Fontaine A-B, Reich, J-P et Zaim V., Ed. Prentice hall; Masson, 1995.

S

[Sandoval, 93] Sandoval V., *CALS - Introduction et mise en œuvre*, Ed. Hermès, 1993.

[Saucier, 97] Saucier A., *Un modèle multi-vues du produit pour le développement et l'utilisation de systèmes d'aide à la conception en ingénierie mécanique*, Thèse de l'Ecole Normale Supérieure de Cachan, 1997.

[Schlenoff, 00] Schlenoff C., Gruninger M., Tissot F., Valois J., Lubell J., lee J., *The Process Specification Language (PSL) - Overview and version 1.0 Specification*, NISTIR 6459, NIST, 2000.

[Schwarze, 96] Schwarze S., *Configuration of Multiple-Variant Products : Application Orientation and Vagueness in Customer Requirements*, vdf Verlag, Zurich, 1996.

[Schwarze, 97] Schwarze S., Schönsleben P., *recent Developments in the Configuration of Multiple-Variant Products : Application Orientation and Vagueness in Customer Requirements*, Proc. of APMS'96, Kyoto, verlag, Chapman&Hall, 1997.

[Scott, 93] Scott Tsao S-K., *An overview of Product Information Management*, 1993 Pan Pacific Conference on Information Systems, National Sun Yat-Sen University, Taiwan, China, May 30-June1, 1993.

[Semmak, 98] Semmak F., *Réutilisation de composants de domaine dans la conception des systèmes d'information*, thèse de doctorat de l'université Paris I, spécialité Informatique, 1998.

[Soininen, 99a] Soininen T., Niemelä I., *Developing a Declarative Rule Language for Applications in Product Configuration*, Proc.of the 1st International workshop on practical aspects of declarative languages PADL99, Janvier 18-19, 1999, San Antonio, texas. Lecture Notes in Computer Science, Springer-Verlag.

[Soininen, 99b] Soininen T., Gelle E., *Dynamic Constraint Satisfaction in Configuration*, 16th National Conference on Artificial Intelligence AAAI'99, Workshop on Configuration, July 18-22 1999, Orlando, Florida.

[Sommerville, 1992] Sommerville I., *Le génie Logiciel*, traduction française par J.M. André. Eds. Addison-Wesley France, 1992.

[Spath, 99] Spath D., Lanza M., Sauter G., *Target-oriented Product Data Management*, Product Data Technology Europe'99 8th symposium, 13-16 April 1999, Stavanger, Norway.

[Spur, 94] Spur G., Mertins K., Jochem R., *Integrated Enterprise Modelling*, Beuth Verlag, Berlin, 1994.

[Stark, 00] Stark J., *Product Data Management*, http://www.johnstark.com/pd37.html, 2000.

[stepwise, 00] http://www.stepwise.org/, *Stepwise Handbook*, February 2000.

[Stumptner, 97] Stumptner M., *An overview of knowledge-based configuration*. In AI Communications, 10(2), June 1997.

[Suh, 99] Suh N.P., *Axiomatic Design : Advances and Applications,* 1999.

[Sunyé, 99] Sunyé G., *PatternGen, un outil de mise en œuvre de patterns de conception,* thèse de Doctorat de l'Université Paris VI, spécialité Informatique, 1999.

T

[Tessier, 95] Tessier C., *La pratique des Méthodes en Informatique de Gestion,* Les éditions d'Organisation, Paris, 1995.

[Tracz, 94] Tracz W., *Reuse State of the Art and State of the Practice Report Card,* proceedings of 3rd International Conference on Software Reuse, November 1-4, 1994, Rio de Janeiro, Brazil. Eds. Frakes W.B. pp 193-194.

[Tollenaere, 99] Tollenaere M., *Internet, Intranet, SGDT : couvertures fonctionnelles et complémentarités,* Journée PRIMECA, IFMA, Clermont-Ferrand, 10 juin 1999.

V

[VDA, 86] VDA-FS, *Vereinung Deutsche Autombilindustie Flächen Schnittstelle : DIN 66301,* DIN-Deutsches Instiut für Normung, Germany, 1986.

[Vernadat, 96] Vernadat F.B., *Enterprise Modeling and Integration : principles and applications,* Ed. Chapman & Hall, 1996.

[Vernadat, 99] Vernadat F, *Techniques de Modélisation en Entreprise : Application aux Processus Opérationnels,* Ed. Economica, 1999.

[Voirpin, 97] Voirpin J-M., Morel M., El Mhamedi A., Sonntag M., *Analyse du processus de création de produits en vue de la réduction du Time to Market,* Actes du 2ème Congrès International Franco-Québécois de Genie Industriel, Albi, France, 3-5 septembre 1997.

W

[W3C, 98] W3C, *Extensible Markup Language (XML) 1.0. W3C recommendation,* February 1998, http://www.w3.org/TR/1998/REC-xml-19980210

[Weske, 99] Weske M., *Adapative Workflows based on Flexible Assignment of Workflow Schemas and Workflow Instances,* Proc. Workshop Informatik'99, enterprise-wide and cross-enterprise Workflow Management : Concepts, Systems, Applications, Paderborn, Germany, October 6, 1999.

[Wikes, 00] Wikes W., Bröking J., *MERCI : Integration of EXPRESS based and XML based component information representation,* Product Data Technology Europe 2000 9th Symposium, 2-5 May 2000, Noordwijk, The Netherlands.

Z

[Zarli, 98] Zarli A., Amar V., Diard F., Marache M., *Combler le fosse entre STEP, CORBA et la réalité Virtuelle pour les applications de nouvelle génération dans l'industrie du BTP,* Actes du MICAD 98, Conférence C3-SGDT : Réussir l'intégration dans l'entreprise, paris, 17-20 mars 1998, France.

Annexe A : Modélisation de données - UML

1. Quelques Notions de Modélisation

Dans cette partie de l'annexe A, nous proposons de définir et positionner certaines notions de modélisation, afin de lever toute ambiguïté terminologique dans la présentation que nous faisons des techniques de modélisation en SI.

Modèle : les définitions du concept modèle sont nombreuses. Un modèle tel que défini par N. Kettani [Kettani, 98] est *une abstraction de la réalité qui, pour un domaine, est prise comme une représentation d'une classe de phénomènes plus ou moins habilement dégagés de leur contexte par un observateur, pour servir de support à l'investigation et/ou à la communication. Un système est décrit par plusieurs modèles; chacun d'entre eux définit un aspect spécifique de ce système (statique, dynamique, fonctionnel).*

Cette représentation, comme le précise F. Vernadat [Vernadat, 99] est exprimée dans un langage de représentation qui peut être formel (ayant une syntaxe et une sémantique bien définies), semi-formel (notation graphique normalisée) ou informel (description en langage naturel).

P-A Muller [Muller, 97] caractérise un modèle comme étant une unité de base du développement qui est très fortement cohérent avec lui-même et faiblement couplé avec les autres modèles par des liens de navigation.

Langage : *c'est un ensemble de constructions qui permettent de décrire formellement les spécifications du SI élaborées aux différents stades du processus de conception, en s'appuyant éventuellement sur les modèles de la méthode d'ingénierie de SI* [Rolland, 88]. Les éléments constituants d'un langage de modélisation sont : des concepts, une syntaxe et une sémantique [Kettani, 98]. Une notation graphique peut lui être associée. Elle en constitue l'expression graphique et visuelle.

Elément de modélisation : un élément de modélisation est *la représentation d'une abstraction issue du domaine du problème. Il constitue un concept fondamental pour la représentation de la chose modélisée (systèmes, phénomènes) et donc pour construire des modèles* [Muller, 97]. Ainsi les notes reportées sur les partitions musicales sont des éléments de modélisation pour la musique, de même que les objets constituent les éléments de modélisation dans l'approche objet.

Un modèle est construit donc à partir d'éléments de modélisation. Les éléments de modélisation correspondent en fait aux concepts des langages de modélisation.

On parle également d'artefacts de développement pour désigner les éléments de modélisation.

Formalisme : ce terme est utilisé pour désigner un langage de modélisation. Il désigne un ensemble d'éléments de modélisation, auquel est associée une représentation, souvent graphique, qui permet de manipuler aisément les modèles et de communiquer l'information entre les différents intervenants d'un projet informatique.

Diagramme : les diagrammes constituent l'expression visuelle et graphique d'un langage. Un diagramme est ainsi *une représentation graphique d'une collection d'éléments de modèle, le plus souvent présentés comme un graphe connexe d'arcs (relations) et vertices (d'autres éléments du modèle)* [Kettani, 98]. N. Kettani attire l'attention sur le fait qu'un diagramme n'est pas un modèle mais une représentation graphique de quelques éléments du modèle. Un diagramme est donc une projection d'un modèle. Plusieurs diagrammes sont généralement nécessaire pour illustrer complètement un modèle. P-A Muller [Muller, 97] précise que les modèles sont regardés et manipulés au travers des vues graphiques, véritables projections des éléments de modélisation contenus par un ou plusieurs modèles. A chaque vue correspondent un ou plusieurs diagrammes. C'est ainsi que des notations différentes peuvent être des vues du même modèle. Par exemple, les notations de Booch, OMT et OOSE utilisent des syntaxes graphiques différentes mais représentent les mêmes concepts objet.

Elles ne sont que des vues différentes des mêmes éléments de modélisation. Une **notation** correspond donc à un ensemble de diagrammes.
Par ailleurs, D. Rieu [Rieu, 99b] note que par abus de langage, le terme "modèle" est souvent utilisé pour parler aussi bien des formalismes que de leur utilisation. Ainsi lorsqu'on parle de modèle de données, il s'agit plutôt du formalisme utilisé pour modéliser des données et lorsqu'on parle du modèle de données du domaine vente, il s'agit de son utilisation pour modéliser les données concernées par le domaine "vente".
Ainsi, l'étude que nous présentons dans le paragraphe 3.1 du chapitre II concerne les *langages* ou *formalismes* de modélisation proposés dans la littérature.

2. Langage de Modélisation UML

Nous décrivons dans cette partie les concepts de base du langage UML, nécessaire pour notre étude. UML s'appuie sur des **concepts** (des choses), des **relations** et des **diagrammes**.

- Les ***concepts*** manipulés dans UML sont de quatre natures. On distingue des concepts *structurels* (les classes, les interfaces, les collaborations, etc.), *comportementaux* (les inter-actions, les états d'objets), *annotationnels* (les notes) et *de groupement* (les package, les sous-systèmes, etc.).
- Les ***relations*** permettent de lier les concepts. UML offre quatre types de relations : les associations, les généralisations, les dépendances et les réalisations.
- Les ***diagrammes*** constituent des représentations graphiques d'ensemble de concepts. Ils sont définis par des graphes connexes dont les sommets sont des concepts et les arcs des relations inter-concepts. Les diagrammes permettent de représenter les systèmes sous différentes perspectives et différents niveaux d'abstractions. UML offre neuf types de diagrammes offrant des visions statiques ou dynamiques des systèmes (*Figure 1*).

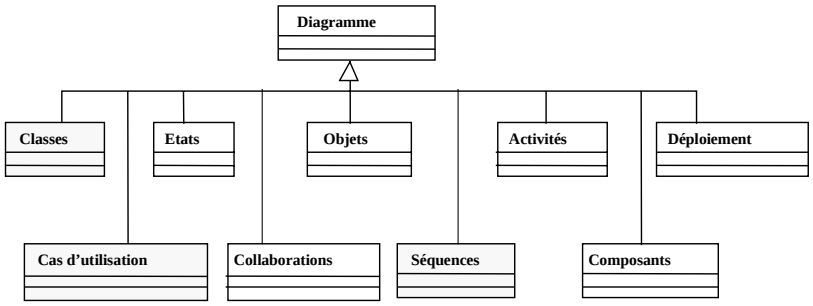


Figure 1 : Les diagrammes UML

Diagrammes statiques

- Les diagrammes des cas d'utilisation facilitent la mise en accord entre d'une part les utilisateurs du système et les experts du domaine et d'autre part les concepteurs informaticiens. Ils permettent de déterminer, de modéliser et d'organiser les fonctionnalités du système cible par catégorie d'acteurs (utilisateurs).
- Les diagrammes de classes expriment la structure statique du système. Ils décrivent les classes et les relations entre elles. Ces diagrammes sont les plus connus et les plus utilisés pour la modélisation des systèmes orientés objet.
- Les diagrammes d'objets illustrent un instantané de la réalité en montrant des objets (instances des classes) et des liens (instances des associations) existants entre ces objets.
- Les diagrammes de composants décrivent l'architecture physique du système. Ils montrent l'organisation des composants et les dépendances entre eux. Ces diagrammes donnent une vision statique de l'implantation du système.

- Les diagrammes de déploiement décrivent la disposition physique des différents matériels entrant dans la composition d'un système.

Diagrammes dynamiques

- Les diagrammes de séquences et les diagrammes de collaborations décrivent les interactions entre objets en montrant en particulier les messages échangés entre eux. Les diagrammes de séquences représentent ces demandes de services selon un point de vue temporel alors que les diagrammes de collaboration mettent davantage l'accent sur une organisation spatiale des objets.
- Les diagrammes d'états décrivent les comportements des objets sous la forme d'automates d'états finis. Tout objet suit le comportement (passe par les états) défini par le diagramme d'états associé à sa classe.
- Les diagrammes d'activités sont des variantes des diagrammes d'états. Ils mettent davantage l'accent sur les aspects opératoires (les activités). Les diagrammes d'activités sont souvent utilisés pour formaliser les comportements des cas d'utilisation.

UML propose donc un langage standard de modélisation. Il est cependant évident que le langage " universel " n'existe pas. UML offre donc des **mécanismes d'extension** permettant de définir de nouveaux concepts ou de nouvelles relations de manière à étendre et à adapter le vocabulaire de modélisation. Ces mécanismes sont au nombre de trois : les *stéréotypes*, les *étiquettes* et les *contraintes*. Les stéréotypes permettent en particulier de définir de nouvelles classes de concepts et de relations.
L'objectif de cette annexe n'est pas de proposer une étude complète du langage de modélisation, mais uniquement de permettre de lire les divers diagrammes UML proposés dans ce manuscrit. Nous détaillerons successivement les diagrammes de classes, les diagrammes d'états, les diagrammes des cas d'utilisation, les diagrammes de séquences et les diagrammes d'activités.

Diagrammes de classes et diagrammes d'objets

▪ Classe et Objet

Le concept de base fondamental utilisé dans les diagrammes de classes est bien entendu celui de classe. Une classe décrit un ensemble d'objets de même structure définie par un ensemble d'attributs et de relations et de même comportement défini par un ensemble d'opérations. Les classes identifient les abstractions du domaine du problème. C'est ainsi que dans le contexte d'un Système d'Information Produit les classes du domaine concerne les articles, les produits, les documents, etc..
Une classe UML comprend trois compartiments qui permettent respectivement de représenter le nom de la classe, ses attributs et ses opérations. Si un diagramme de classes est trop important (trop de classes et de relations) ou si seule une vision globale est nécessaire, il est possible de donner une vision moins complète des classes, par exemple en ne faisant apparaître que les noms des attributs et/ou des opérations, voire en ne donnant que le premier compartiment (nom de la classe). Ce premier compartiment peut être doté d'un stéréotype permettant de " typer " les classes. Par exemple, le stéréotype <<entité>> a été introduit pour modéliser les classes d'objets métiers où un objet métier est "un objet au cœur du système et représentatif d'une partie du métier de l'entreprise indépendamment de son implantation technique ou de son utilisation" [Kettani, 98].

▪ Association et Lien

Les associations constituent les relations fondamentales des diagrammes de classes. Elles permettent d'associer un objet d'une classe à un ou plusieurs objets d'une autre classe. Les liens sont des instances (éléments) des associations au même titre que les objets sont des instances des classes.

Les rôles dans une association décrivent comment une classe voit l'autre classe au travers de l'association. Cette notion est particulièrement importante dans deux cas. Le premier cas concerne les associations multiples entre deux classes. La Figure 2 illustre par exemple deux associations différentes entre les classes Article et Entreprise. Une entreprise peut jouer le rôle de fournisseur vis à vis de certains articles et/ou celui de client vis à vis d'autres articles qui sont alors perçus comme des produits finis par l'entreprise cliente.

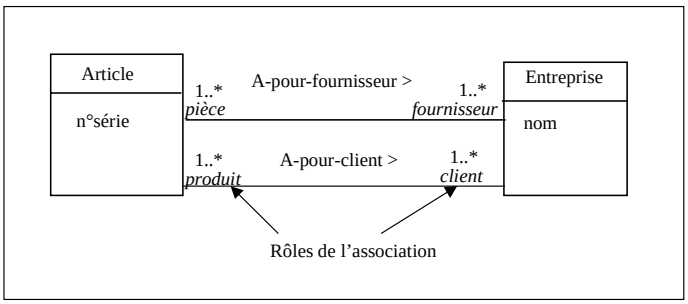


Figure 2 : Association multiple entre deux classes

Le deuxième cas pour lequel la notion de rôle est fondamentale est celui des associations récursives. Une association récursive relie une classe à elle-même. Un article est par exemple composé d'autres articles. La Figure 3 illustre les deux rôles de l'association A-pour-composant liant un article à ses articles composants. Le nommage des rôles est ici indispensable car un même objet peut jouer le rôle de composant dans un lien et celui de composé dans un autre. Notons que pour alléger les diagrammes il est inutile de nommer à la fois l'association et les rôles.

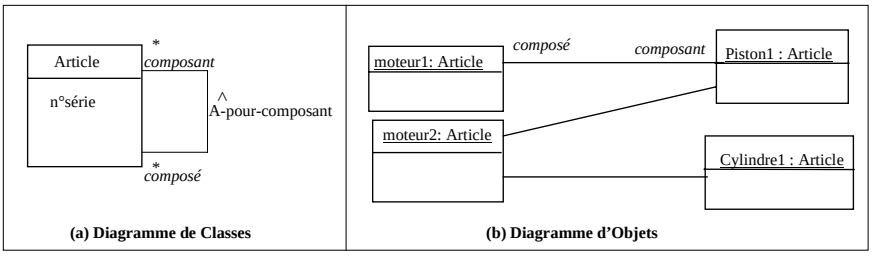


Figure 3 : Association récursive

Dans une association, chaque rôle porte une multiplicité qui indique combien (au minimum et au maximum) d'objets de la classe jouant le rôle peuvent être liés à un objet de la classe associée. La Figure 4 donne les multiplicités usuellement utilisées [Muller, 97].

1	Un et un seul
0..1	Zéro ou un
M..N	De M à N
*	De Zéro à plusieurs
0..*	De Zéro à plusieurs
1..*	De un à plusieurs

Figure 4 : Valeurs usuelles des multiplicité [Muller 97]

Une association peut tout comme une classe détenir des attributs, on parle d'association attribuée [Muller 97]. C'est ainsi qu'un article est fourni par un fournisseur à un certain prix (cf. Figure 5). Dans certains cas il devient nécessaire d'élever une association au rang de classe, c'est le cas si on veut pouvoir la doter d'un comportement (d'opérations) et/ou l'associer à d'autres classes. Dans ce cas le premier compartiment de la classe association comporte un nom.

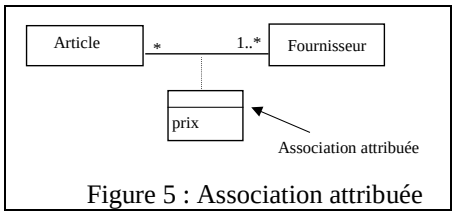


Figure 5 : Association attribuée

- **Agrégation et Composition**

UML offre deux types d'association particulièrement intéressantes pour la gestion des données techniques : l'agrégation et la composition. L'agrégation est une association " dans laquelle une classe joue un rôle prédominant par rapport à l'autre "[Muller, 97]. Il peut d'ailleurs s'agir de la même classe dans le cas des associations récursives. Une association d'agrégation est notée par un losange blanc du côté de l'agrégat. L'agrégation est utilisée lorsque l'on souhaite indiquer qu'un des objets doit être manipulé comme " un tout ", en particulier que l'on aura certainement à traiter des propagations structurelles (valeurs d'attributs), comportementales (une action sur l'objet agrégat déclenche par exemple une action similaire dans ses parties), existentielles, etc.. La multiplicité du côté de l'agrégat peut rester très ouverte.
La composition est un cas particulier de l'agrégation qui dénote une dépendance quasi complète du composant vis à vis du composé : le composant peut naître hors de son composé mais n'est pas partageable. Il s'ensuit que la multiplicité du côté de l'agrégat ne peut prendre que les valeurs 0 ou 1.

- **Héritage**

La relation d'héritage permet d'établir des classifications de concepts. Dans un premier temps on peut l'assimiler à l'inclusion ensembliste : les voitures sont des " sorte de " de produits qui eux-mêmes peuvent être perçus comme des " sorte de " d'articles, etc.. La relation d'héritage permet d'organiser les classes par niveau d'abstraction. La notion d'article est plus abstraite que celle de produit fini elle même plus abstraite que celle de voiture, etc.. La relation d'héritage est représentée par une flèche (dont la tête est un triangle blanc) qui pointe de la classe la plus spécialisée, par exemple Voiture, vers la classe plus générale, par exemple Produit. On dit que Voiture est une sous-classe de Produit et inversement que Produit est la super-classe de Voiture. La Figure 6 illustre une hiérarchie d'héritage[168].

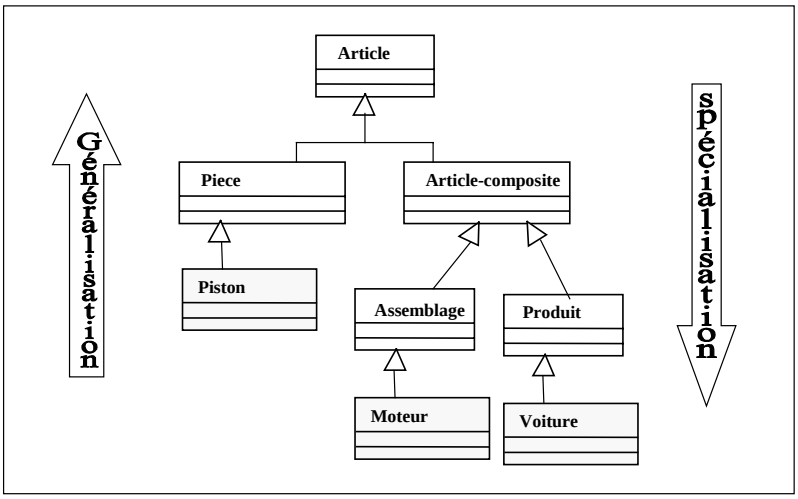


Figure 6 : Relation d'héritage

[168] Nous ne traitons ici que des cas d'héritage simple où une classe admet une seule super-classe. UML autorise l'héritage multiple : une classe peut hériter de plusieurs super-classes.

La généralisation est un mécanisme ascendant consistant à factoriser au sein d'une super-classe des propriétés communes aux objets de classes différentes. Par exemple les assemblages et les produits ont des propriétés communes liées au fait qu'ils sont des articles composés : en particulier ils admettent et gèrent un ensemble de composants. La classe Article-composite doit détenir des propriétés (attributs, associations, opérations) lui permettant cette gestion. Inversement la spécialisation est un mécanisme descendant qui consiste à étendre les propriétés d'une classe en définissant des sous-classes de celle-ci.

Diagrammes d'états-transitions

Les diagrammes d'états-transitions visualisent des automates d'états finis, du point de vue des états et des transitions. Les automates permettent de décrire globalement le comportement d'éléments individuels.
Dans le cas des objets, le comportement peut se décrire de manière formelle en termes d'états et d'événements, au moyen d'un automate relié à la classe de ces objets. Un diagramme d'états-transitions est associé à une classe et met en évidence tous les états possibles de cette classe ainsi que les événements provoquant un changement d'état. Un état est une situation durable dans laquelle peuvent se trouver les objets d'une classe et à laquelle on associe des règles de gestion et des activités particulières. Un état peut aussi être défini comme un ensemble de valeurs prises par les attributs de l'objet et traduisant le comportement global de l'objet. La Figure 7 illustre ainsi les passages entre états de la classe Article correspondant aux événements de stockage et de déstockage : un article peut par exemple être disponible, en rupture de stock, etc.
Le formalisme retenu par UML pour représenter les automates s'inspire des Statecharts[169].

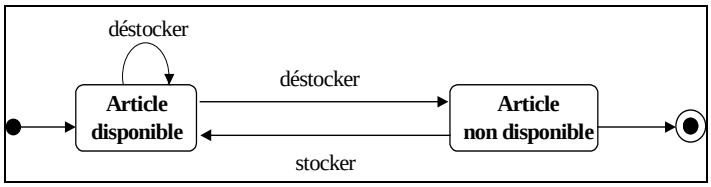


Figure 7 : Diagramme d'états-transitions de la classe Article

Diagrammes des cas d'utilisation

Les diagrammes des cas d'utilisation (use cases) permettent de structurer les besoins des utilisateurs en mettant en évidence le comportement attendu du système. Ils constituent un très bon modèle pour déterminer les limites du système et ses inter-actions avec son environnement. Un cas d'utilisation correspond à la représentation d'une fonctionnalité du système, fonctionnalité qui sera déclenchée par un utilisateur (acteur).
Un diagramme de cas d'utilisation est constitué des acteurs du système, du système et des cas d'utilisation. Un acteur est une entité externe au système et qui en attend quelque chose. Il peut s'agir d'un acteur humain mais aussi d'un autre système, de matériel externe, etc.. Un acteur représente un rôle ou une fonction et non une personne ou une chose. Par exemple on parlera du " Responsable des stocks " ou du " Responsable des nomenclatures " et non de Mr Dupont. Une personne ou une chose peut jouer plusieurs rôles et inversement un rôle peut être joué par plusieurs personnes.
La Figure 8 illustre un diagramme de cas d'utilisation faisant intervenir deux acteurs. Le " Responsable des stocks " est associé à ses cas d'utilisation : il peut par exemple stocker, déstocker les articles et consulter leur prix.

Les cas d'utilisation peuvent être organisés en paquetages[170], ils peuvent également être liés les uns aux autres par trois types de relations, illustrés dans la Figure 9.

[169] Harel D. *Statecharts : A Visual Formalism for Complex Systems*. Science of Computer Programming, vol. 8, 1987.

- La relation de généralisation permet de définir un cas d'utilisation comme la généralisation d'un ou plusieurs autres cas. C'est ainsi que la modification d'un article peut être perçue comme la généralisation de cas d'utilisation plus spécifiques tels que la suppression ou l'ajout d'un composant d'un article.
- La relation d'extension permet de définir un cas d'utilisation comme l'extension d'un autre cas. En particulier, elle permet de décrire des processus exceptionnels ou optionnels. Par exemple le stockage d'un article en rupture de stock est un cas exceptionnel du stockage d'un article.

La relation d'inclusion permet à un cas d'utilisation de faire appel aux services d'un autre cas. Cette relation permet de « factoriser » une fonctionnalité invocable dans plusieurs cas d'utilisation. Par exemple l'identification d'un article peut être appelée par la consultation, la modification, la suppression des articles.

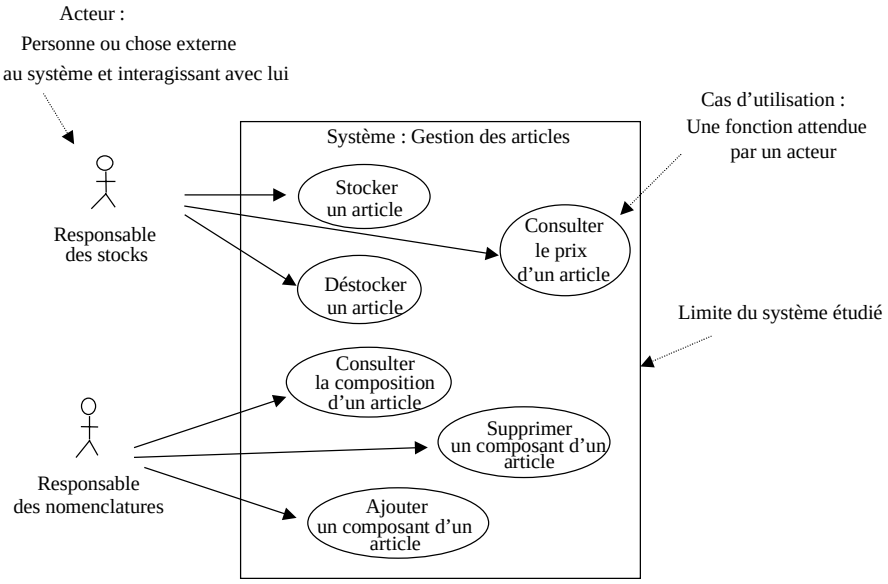


Figure 8 : Diagramme des cas d'utilisation

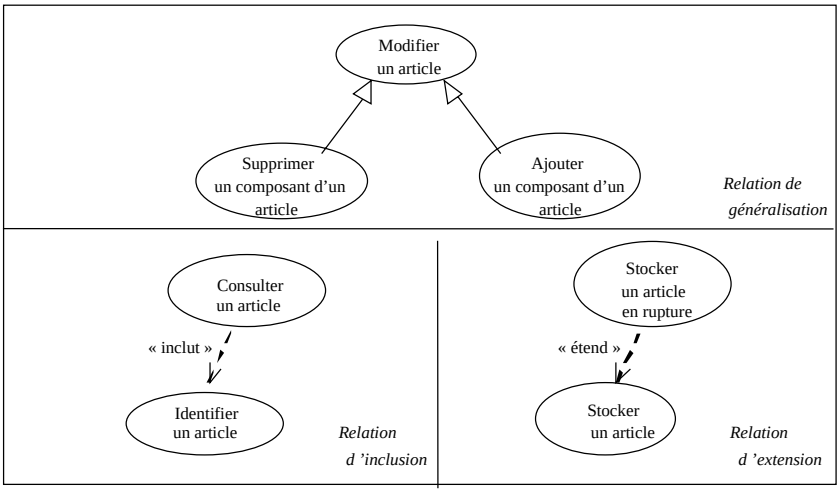


Figure 9 : Relations entre cas d'utilisation

Diagrammes de séquence

Les diagrammes de séquence permettent de représenter les interactions entre objets en précisant la chronologie des échanges de messages. Ils peuvent être utilisés pour représenter les scénarios d'un cas d'utilisation donné :

[170] En UML un paquetage (package) permet de regrouper des éléments de modélisation (des classes, des cas d'utilisation, etc.). Il constitue une technique d'organisation des modèles.

un diagramme de séquence associé à un cas d'utilisation spécifie les échanges de données ou d'événements entre un utilisateur et le système. Un message reçu par un objet déclenche l'exécution d'une opération et en général renvoie un message qui correspond au résultat de l'opération. Nous donnons à titre d'exemple, à la figure 10, le diagramme de séquence d'un scénario du cas d'utilisation « Ajouter un composant », demandé par le responsable des nomenclatures. Ce diagramme de séquence met en jeu deux objets des classes Magasin et Article. L'article correspond au composite à qui le responsable des nomenclatures veut rajouter un composant. Le responsable des nomenclatures demande au magasin d'ajouter un composant (de numéro : noseriecomposant) à un composite (de numéro : noseriecomposite). Le magasin accède aux objets à partir de leur numéro de série : *ci* identifie le composite alors que *ca* identifie le composant. Le magasin demande à l'objet *ci* de prendre en compte *ca* comme composant (appel de Ajouter composant ()).

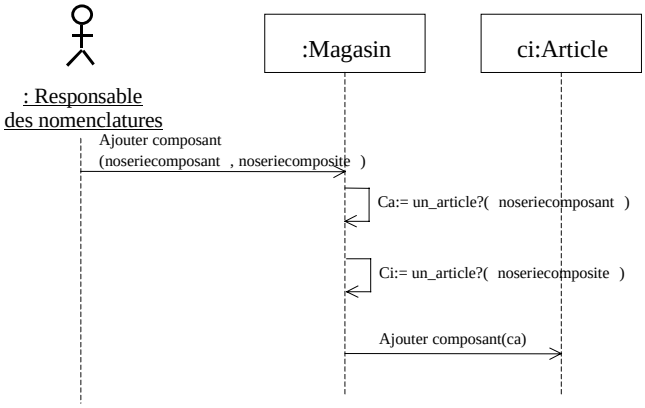


Figure 10 : Diagramme de séquence

- **Diagrammes d'activités**

Un diagramme d'activités est une variante de diagramme d'états-transitions, organisé par rapport aux actions et principalement destiné à représenter le comportement interne d'une méthode (la réalisation d'une opération) ou d'un cas d'utilisation. Un diagramme d'activités représente l'état d'exécution d'une méthode ou d'un cas d'utilisation, sous la forme d'un déroulement d'étapes regroupées séquentiellement dans des branches parallèles de flot de contrôle. Chaque activité représente une étape particulière dans l'exécution de la méthode englobante ou du cas d'utilisation. Les activités sont reliées par des transitions automatiques, représentées par des flèches, comme les transitions dans les diagrammes d'états-transition. Lorsqu'une activité se termine, la transition est déclenchée et l'activité suivante démarre. Les transitions entre activités peuvent être gardées par des conditions booléennes, mutuellement exclusives. Les gardes se représentent à proximité des transitions dont elles valident le déclenchement. UML définit un stéréotype optionnel pour la visualisation des conditions. Une condition est matérialisée par un losange d'où sortent plusieurs transitions.

Les diagrammes d'activités représentent les synchronisations entre flots de contrôles, au moyen de barres de synchronisation. Une barre de synchronisation permet d'ouvrir et de fermer des branches parallèles au sein d'un flot d'exécution d'une méthode ou d'un cas d'utilisation. Les transitions au départ d'une barre de synchronisation sont déclenchées simultanément. Inversement, une barre de synchronisation ne peut être franchie que lorsque toutes les transitions en entrée sur la barre ont été déclenchées.

Par ailleurs, les diagrammes d'activités peuvent être découpés en couloirs d'activités pour montrer les différentes responsabilités au sein d'un mécanisme ou d'une organisation. Chaque responsabilité est assurée par un ou plusieurs objets et chaque activité est allouée à un couloir donné. Lorsque les objets assurant les activités sont des acteurs, il est possible de les représenter à l'aide d'acteurs UML

Il est possible également de faire apparaître clairement les objets dans un diagramme d'activités, soit au sein des couloirs d'activités, soit indépendamment de ces couloirs. Les objets peuvent être représentés à l'aide de carrés.

Pour augmenter la lisibilité, un objet peut figurer à plusieurs reprises dans les diagrammes ; son état est alors spécifié à chaque occurrence dans une expression entre crochets.
Les flots d'objets sont représentés par des flèches pointillées. Une flèche relie ainsi un objet à l'activité qui l'a créé. De même, une flèche relie un objet aux activités qui le mettent en jeu. Lorsqu'un objet produit par une activité est utilisé immédiatement par une autre activité, le flot d'objets représente également le flot de contrôle ; il est alors inutile de représenter explicitement ce flot de contrôle.
Nous donnons à titre d'exemple, à la Figure 11, le diagramme d'activités d'un cas d'utilisation « Traitement de commande produit ».

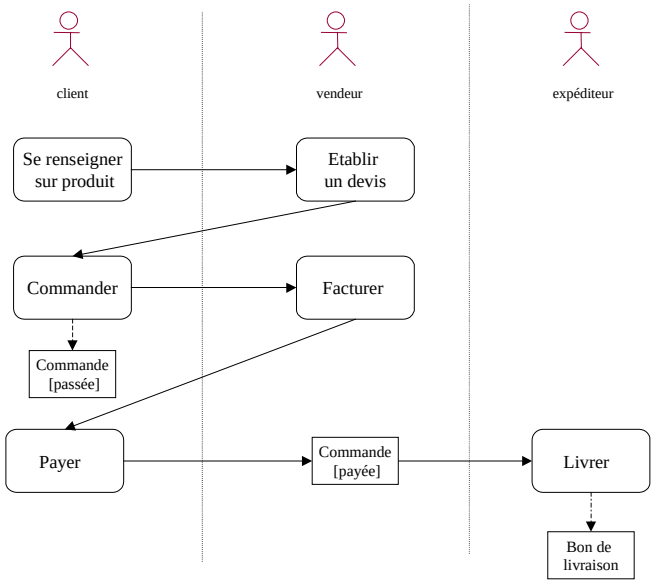


Figure 11 : Diagramme d'activités [Muller, 97]

3. Le langage OCL

Le langage de contrainte OCL fait partie d'UML, il peut être utilisé pour spécifier toutes sortes de contraintes, pré et post conditions sur les opérations de classes des différents modèles. OCL a les caractéristiques d'un langage d'expression , d'un langage de modélisation et d'un langage formel.

OCL est un langage navigationnel permettant :

La spécification des invariants sur les classes et les types dans les modèles de classes ;
La spécification des invariants de types pour les stéréotypes ;
La description des pré et post conditions appliquées aux méthodes et opérations ;

La présentation qui suit décrit les notions et les concepts de base du langage OCL. Elle est largement issue de la description proposée par I. Hassine [Hassine, 00].

Notions de base d'OCL:

- **Principe de modélisation par contrat :**

Une manière de spécifier des opérations et des méthodes consiste à exprimer des pré et post conditions. Dans UML, on peut utiliser les expressions OCL pour définir les pré et post conditions dans toutes les classes, types et

interfaces. Leur utilisation se réfère souvent au principe de modélisation par contrat. Ce dernier peut être utilisé dans toute méthode de développement orientée objet.

Définition de Contrat :

En termes de modélisation orientée objet, un contrat définit les responsabilités d'un objet d'une façon claire et non ambiguë. Un objet est responsable de l'exécution de services (obligations) si et seulement si certains droits sont satisfaits. Les objets qui sont disposés à utiliser ces services offerts, sont appelés des clients ou consommateurs. Les objets qui offrent ces services sont dits des serveurs (supplier).

Dans la technologie orientée objet, un contrat est offert par un objet indépendamment de la présence ou non des clients. Mais si un client demande un service, il sera soumis aux conditions du contrat.

Le contrat décrit les services fournis par un objet. Pour chaque service, on spécifie :

- Les conditions sous lesquelles un service est offert
- La spécification du résultat du service offert dans le cas où les conditions précédemment décrites sont satisfaites.

Un exemple de contrat :

Un Circuit-microprocesseur est remplacé par un autre si et seulement si il est mis hors tension.

- **Pré et post conditions :**

L'interface d'un objet décrit toutes les opérations qui sont exécutées par l'objet lui-même. Pour chaque opération, les droits et obligations de l'objet qui offrent le contrat sont spécifiés par des pré conditions et post conditions. Une pré condition doit être vérifiée avant l'exécution de l'opération. Les obligations sont spécifiées par des post conditions. Celles ci doivent être vraies juste après la fin d'exécution de l'opération. Si une pré ou post condition n'est pas vérifiée le contrat est rompu.

- **Invariant :**

En addition aux pré et post conditions, un troisième type de contraintes a été défini : les **invariant**. Les invariants sont toujours couplés aux classes, types ou interfaces. Un invariant exprime une contrainte qui doit être vérifiée par toutes les instances d'une classe. Un invariant est décrit au moyen d'une expression qui doit être vraie chaque fois que l'invariant est rencontré. Les Pré et post conditions doivent être vérifiées dans les intervalles de temps respectifs avant et après l'exécution d'une opération, alors qu'un invariant doit être vrai tout le temps.

En UML, ces trois contraintes sont prédéfinies comme étant trois stéréotypes standards :

<<invariant>>
<<precondition>>
<<postcondition>>

Concepts de base en OCL:

- **Contexte d'un invariant**

Le contexte d'un invariant est toujours une classe, une interface ou un type. Une seule expression peut suivre une déclaration de contexte. Si cette dernière est utilisée comme un invariant, elle doit toujours être évaluée à vrai pour toutes les instances et tout le temps.

- **Contexte d'une pré et post condition :**

Le contexte d'une pré ou post condition est une opération ou méthode. Les paramètres d'une opération contextuelle sont accessibles dans l'expression OCL définissant la contrainte.

- **Invariants sur les classes associées :**

On peut définir un invariant sur les attributs d'un objet d'une classe ou sur ceux des classes associées. En OCL, on utilise rolename pour se référer à l'objet de l'autre bout de l'association sinon c'est le nom de la classe qui le remplace dans l'expression.
Comme exemple , on prendra le modèle de classes du circuit intégré et on ajoute une relation d'association entre les deux classes microprocesseur et carte mère, sur qui on définit l'invariant qui vérifie l'égalité des nombres de broches.

Circuit-Microprocesseur
Self.Broche_process = Self.carte-mère.broche_support_process

Traitement avec collection d'objets

Souvent, la multiplicité d'une association est supérieure à 1. Elle peut lier un objet à un ensemble d'objets de la classe associée ; pour les traiter OCL dispose d'une multitude d'opérations appelée collection utilisées chaque fois que le résultat d'un lien de contrainte est constitué de plusieurs collections ou ensemble d'objets. Dans le cas de notre exemple, une carte mère peut contenir un ou plusieurs microprocesseur qu'on fixe la borne supérieure à 2 :

Carte mère
Self.Microprocesseur → size <= 2

Dans OCL, on distingue trois différents types de collection : *set*, *sequence* et *bag*. Un set est un ensemble mathématique ne contenant pas d'éléments dupliqués. Un bag est un set mais qui peut contenir des copies d'élément. Une *sequence* est un *bag* où les éléments sont ordonnés. Les trois types précédemment introduits peuvent être spécifiés par un littéral OCL :

```
Set {1, 2, 74, 19, 4, 10}
Bag {1, 2, 3, 2, 3}
Sequence {1, 3, 9..20, 35}   /* 9..20 : intervalle.
```

- **Héritage**

L'avantage de l'héritage est qu'un objet utilisant une superclasse ne cherche pas à connaître les sous-classes de cette dernière. Mais parfois, on a besoin de les différencier. Dans notre exemple, si on ajoute une classe micro ordinateur en relation avec la classe circuit mémoire et deux sous-classes de cette dernière : mémoire morte (ROM) et mémoire vive (RAM) et on définit les invariants suivants :

La somme des capacités de blocs mémoires d'un micro-ordinateur doit être inférieure ou égale 100 KO.

Micro-ordinateur
Self.Circuit mémoire → Collect(capacité_mémoire)→ Sum < 100 * 1024

- **Traitement des énumérations :**

L'énumération est définie comme un type d'attribut dans le modèle de classe UML. Dans une expression OCL les valeurs d'un attribut de ce type sont précédées par le symbole #.
Si on prend le cas de la résistance, il y a une bande colorée qui indique l'incertitude sur la valeur de la résistance, elle ne doit pas dépasser 5% (la couleur de la bande est silver).

Résistance
Self .Bande_incertitude = #silver

- **Pré et post conditions**

Dans la description du principe de modélisation par contrat on a expliqué l'utilisation des pré et post condition. En effet, elles spécifient les effets d'une opération sans indiquer la méthode algorithmique utilisée ou une implémentation. Quand les pre et post-conditions doivent être précisés OCL est un outil adéquat pour le faire.

contexte ::operation(parametre1 : type1, parametre2 …) : ReturnType
pre : parametre1 > ……
post : result =

- **Instances et types :**

Les types en OCL sont subdivisés en deux catégories :

- Types prédéfinis :
- Types de base (Integer, Real, String, Boolean)
- Type de collections prédéfinis : Collection, Set, Bag et Sequence. Ils sont utilisés pour spécifier le résultat exact de la navigation à travers les associations entre classes.

Types définis par les modèles utilisateurs : les types comme composant électronique et circuits intégrés sont définis par le modèle UML. Chaque classe, interface ou type dans UML est automatiquement un type dans OCL auxquels on ajoute le type énumération qui peut être utilisé comme type d'attributs.

- **Types définis dans le modèle :**

Les contraintes font toujours référence à des objets définis dans le modèle UML. Tous les éléments d'un modèle définissent des types dans OCL. Le développeur d'un modèle UML peut créer des nouveaux types qui seront utilisés comme étant un type prédéfini dans OCL, appelés model types. Les model types sont désignés par leurs noms dans le modèle UML. Les différents model types sont : type, classes, interfaces, associations classes, acteurs, cas d'utilisation et les types de données définis dans le modèle UML. Comme exemples de types dans le modèle de circuit intégré, on peut citer : Résistance, microprocesseur … et toutes les autres classes comme l'énumération {silver, gold} définie dans la classe résistance. Les propriétés d'un tel type sont :

- Attributs et opérations : si une classe C est spécifiée dans un modèle UML, alors C constitue un type valide en OCL. Tous les attributs, les opérations et les méthodes de C dans le modèle le sont dans OCL. Parce que OCL est un langage a effet de bord, les opérations qui font changer l'état d'un objet ne sont prises en compte. Seules les opérations dites query operation qui retournent une valeur mais ne changent rien, peuvent être utilisées dans les expressions OCL.
- Navigations dérivées des associations ou des agrégations : pour une classe ou un type, chaque association qui lui est attachée définie une navigation. La navigation porte le nom du rôle de la classe de l'autre bout de l'association sinon c'est le nom de la classe qui est pris en compte.

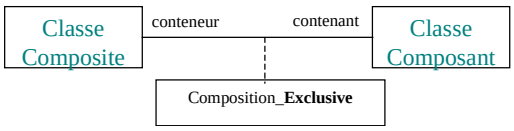
Annexe B : Sémantique des liens de composition

1. Propriétés du lien de composition

Les propriétés attachées aux liens de composition permettent d'enrichir la représentation des objets composites et de simplifier leur manipulation. Les propriétés d'un lien de composition sont définies au niveau des classes [Oussalah, 97]. Ces propriétés permettent de décliner diverses constructions de la nomenclatures de base, correspondant à une composition récursive d'éléments.
Les propriétés les plus utilisées et les plus significatives dans les modèles objets sont les suivants [Oussalah, 97]:

1. Exclusivité et partage

Un objet référencé par un lien exclusif ne peut être référencé que par ce seul lien de composition et ne peut donc faire partie que d'un seul objet composite.
A l'inverse, un composant référencé par un lien de composition partagé peut être référencé par un nombre quelconque de liens de composition partagés et peut donc faire partie de plusieurs objets composites simultanément. Les objets composites référençant un même objet composant peuvent être des instances d'une même classe ou de classes différentes.
Dans [Hassine, 00], les contraintes imposées par le lien de composition exclusif sont exprimés en OCL comme suit :



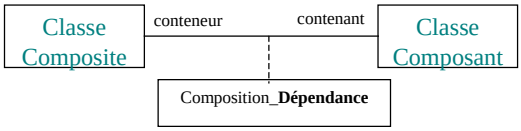
OCL :

```
Context composant
Inv : (self.conteneur->size=1)or(self.conteneur->size=0)
```

/* On vérifie que l'ensemble des pères (conteneurs) d'un composant est soit vide soit formé d'un seul singleton */

2. Dépendance et indépendance

La dépendance permet de spécifier que l'existence d'un objet composant est directement liée à celle de son composite. Dans ce cas, la destruction de l'objet composite entraîne automatiquement la destruction du composant. La notion de dépendance est particulièrement utile dans la manipulation des objets composites. En effet, si un objet composite est détruit et si ses composants doivent l'être également, la notion de dépendance évite à l'utilisateur de rechercher les composants séparément et de les détruire un par un.
Dans [Hassine, 00], les contraintes imposées par le lien de composition dépendant sont exprimés en OCL comme suit :



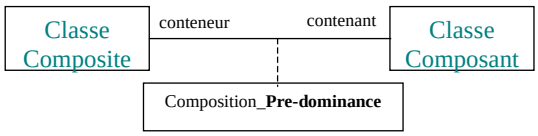
OCL :

```
Composite::Supprimer ()
post:(self.contenant->isEmpty)and(self.oclAsType(Composant).conteneur->forall(c|
c.contenant=c@pre.contenant->excluding(self))
```

/* Dans la post condition on vérifie que les fils ont été bien supprimés, puis on vérifie que le composite (à supprimer) a été supprimé des listes de fils de ses parents */

3. Prédominance et non prédominance

A l'instar de la précédente propriété, le lien de prédominance spécifie une dépendance existentielle sauf que dans ce cas, l'existence d'un objet composite est à liée celle de ses composants, et donc la suppression de l'objet composant entraîne de l'objet composite. Lorsque le composant est détruit, l'objet composite perd son identité.
Dans [Hassine, 00], les contraintes imposées par le lien de composition pré-dominant sont exprimés en OCL comme suit :



OCL :

Composant::Supprimer
post:(self.conteneur->forall(c|c.conteneur->forall(c.contenant=c@pre.contenant-excluding(c))

/* lorsqu'on veut supprimer un composant, il est nécessaire, dans un premier temps de supprimer l'élément ascendant, puis d'éliminer les liens de cet ascendant qui le rattachent à ses parents. */

4. Combinaison des propriétés

Un lien de composition peut toujours avoir l'une des propriétés citées précédemment, mais il peut aussi supporter une combinaison de celles-ci. C. Oussalah [Oussalah, 97] souligne que par défaut, selon les modèles, les liens de compositions sont exclusifs, dépendants et non prédominants (cas de la modélisation des hiérarchies des compositions physiques) ou partagés, indépendants et non prédominants (dans le but de minimiser les contraintes par défauts).
La richesse expressive d'un objet composite dépend fortement des différentes propriétés qui peuvent être associées aux liens de compositions. C. Oussalah précise tout de même qu'un compromis est nécessaire entre la charge sémantique et la puissance fonctionnelle afin de ne pas alourdir la représentation et la manipulation des objets composites [OUSS 97].

2. La sémantique du lien de composition des nomenclatures produit

Le tableau suivant présente la sémantique du lien de composition dans les nomenclatures produit du SIP :

	certains nœuds sont des variantes d'autres nœuds	certains nœuds sont optionnels	lien de composition exclusif	lien de composition partagé	lien de composition dépendant	lien de composition indépendant	lien de composition pré-dominant	lien de composition non pré-dominant
N. organique générique	*	*		*		*	* s'il n'existe pas des composants optionnels	* s'il existe des composants optionnels
N. organique type	*	*		*		*	* s'il n'existe pas des composants optionnels	* s'il existe des composants optionnels
N. organique physique			*		*		* s'il n'existe pas des composants optionnels	* s'il existe des composants optionnels
N. fonctionnelle		*		*		*		*
N. géométrique générique	*	*		*		*	*	
N. géométrique type	*	*		*		*	*	

Annexe C : Analyse des mécanismes d'évolution des objets métiers du SIP

L'objet de cette annexe est d'analyser la terminologie des divers termes proposés dans la littérature pour désigner l'évolution des concepts produit, afin de dégager une terminologie commune.

3. Version – Révision – correction

Diverses définitions de ces mécanismes sont proposées dans la littérature [CIMdata, 97] [AIT, 97a] [ISO, 94c] [EDS, 98] [Maurino, 93]. Nous retenons les suivantes :

- Version : les versions présentent entre elles des différences de définition du produit qui répondent à des fonctions différentes pour les utilisateurs (bloc d'identification[171] différent) [BNAE, 90].
- Révision : une révision est une modification d'une donnée produit après que cette donnée a été validée pour utilisation[172] [CIMdata, 95].
- Correction : une correction est une évolution de la représentation des informations de définition n'ayant aucune répercussion sur la définition du produit, ni sur le produit lui-même. Il peut par exemple s'agir d'une amélioration d'un graphe, d'une correction d'une erreur matérielle manifeste (faute d'orthographe, désignation non normalisée, etc.), d'un changement de désignation si l'ancienne et la nouvelle désignation s'appliquent à deux objets identiques. Une correction entraîne obligatoirement le changement de l'indice de validité significatif de la correction et le bloc d'identification du produit ne subit aucun changement [BNAE, 90].

L'étude de ces définitions fait constater que :

- Les corrections sont réservées aux évolutions de représentations. Elles caractérisent des modifications éditoriales (changement de forme tel que les fautes d'orthographe, les désignations non normalisées) sur des représentations et donc sans impact sur les objets environnants.
- Les versions et les révisions peuvent concerner des articles ou des représentations. Dans le cas d'article, elles correspondent à des modifications de définition de l'article. Dans le cas de représentation, elles correspondent à des modifications sémantiques (changement de fond tel que le changement de valeur de données, l'ajout de nouvelles

[171] Un produit est identifié à partir de son bloc d'identification composé du code fabricant et de la référence fabricant qui est une combinaison de caractères attribuée par le concepteur responsable de la définition pour identifier un produit de sa responsabilité [BNAE, 90].

[172] Traduction de : revision is a modification of any product data after that data has been released for use.

données). Toutefois, aucune distinction entre les deux mécanismes de version et de révision n'apparaît dans les définitions rencontrées. C'est ce qui explique l'utilisation indifférente dans certaines entreprises de l'un ou de l'autre de ces deux mécanismes pour caractériser les évolutions de définition des articles ou des représentations. Notons toutefois que certains auteurs choisissent d'inclure les versions à l'intérieur des révisions[173].

4. Modification mineure, majeure et critique

La deuxième famille de types d'évolution souvent rencontrée dans la littérature attache un caractère de criticité aux modifications. [Maurino, 93] classifie les modifications en mineure, majeure et critique selon le type et l'interchangeabilité[174] des objets modifiés, comme suit :

Pour les articles et les fonctions	Pour les produits
modification mineure : évolution qui conserve l'interchangeabilité de l'objet initial et de l'objet modifié. Seul évolue le mode d'obtention de l'objet. Par exemple un changement de processus de fabrication d'un article sans impact sur la définition. ***modification majeure*** : évolution qui rompt l'interchangeabilité de l'objet initial et de l'objet modifié. Par exemple un changement de matière d'un article qui doit être assemblé avec un autre article de même matière (l'évolution de l'objet doit être synchronisée avec celle d'autres objets).	***modification mineure*** : seul le processus de réalisation du produit est impacté, à l'exclusion de la définition, des fonctions et des processus d'utilisation. Seuls les dossiers industriels de fabrication et de contrôle du produit évoluent. ***modification majeure*** : la définition et les processus de réalisation et d'utilisation sont seuls impactés. ***modification critique*** : les fonctions du produit sont impactées ce qui correspond à une répercussion sur les CdCF et sur les spécifications techniques du produit.

Tableau 1 - Modifications majeure et mineure [Maurino, 93]

Alors que les trois premiers concepts étudiés (version, révision, correction) sont définis en terme de nature de la modification apportée à l'objet modifié (éditoriale, sémantique, etc.) et selon sa nature (article ou représentation), les trois derniers concepts (modifications mineure, majeure, critique) sont plutôt définis en terme d'impact de l'objet modifié sur ses objets environnants, indépendamment de la nature de l'objet modifié. Par objets environnants, nous désignons deux types d'objets :

– les cas d'emplois d'un objet : ils s'obtiennent par une analyse ascendante en explorant les liens de dépendance (ou de cause à effet, ou liés par des contraintes de précédence dans le processus d'acquisition de cette donnée) *aboutissant* à cet objet. Ce concept est plutôt

[173] Dans [EDS, 98], il est précisé que "les donnée peuvent être versionnées plusieurs fois dans une révision donnée.

[174] Dans [Maurino, 93], *deux objets A et A' sont interchangeable si A' peut être substitué à A sans que cela n'entraîne l'évolution d'un quelconque cas d'emploi de A.*

Soulignons que dans cette définition l'interchangeabilité est totale : elle concerne tous les cas d'emploi de l'objet substitué. Dés qu'un cas d'emploi évolue au titre de l'évolution de l'objet A, l'interchangeabilité de celui-ci est rompue.

utilisé pour les articles dans une nomenclature où les cas d'emploi correspondent aux composés utilisant ces articles dans leur composition.

– les cas d'utilisation d'un objet : ils s'obtiennent par une analyse descendante en explorant les liens de dépendance (ou de cause à effet) *issus* de cet objet. Dans le cas d'un composé, les cas d'utilisation sont les articles entrant dans sa composition[175].

Nous pouvons alors définir ces types d'évolution, indépendamment du type de l'objet modifié, comme suit :

- une *modification mineure* d'un objet est une évolution de l'objet qui conserve l'interchangeabilité de l'objet initial et de l'objet modifié. Le mode d'obtention de l'objet peut évoluer mais la modification est sans aucune conséquence sur l'utilisation de l'objet. L'objet évolue sans impact sur les objets l'utilisant c'est à dire ses cas d'emploi.
- une *modification majeure* sur un objet est une évolution de l'objet qui rompt l'interchangeabilité de l'objet initial et de l'objet modifié. C'est une évolution qui par ailleurs exige l'évolution synchronisée de certains objets environnants avec l'objet modifié (par exemple un changement de matière d'un article qui doit être assemblé avec un autre article de même matière). Ce n'est pas uniquement le mode d'obtention qui change mais aussi le mode d'utilisation de l'objet.

5. Correspondances entre les types d'évolution

Comme nous l'avons énoncé dans le chapitre IV, il n'existe pas de relation précise entre les termes version/révision/correction et les termes modification mineure/majeure. Cependant, certains auteurs ont proposé des correspondances. Ainsi, dans [Jensen, 96], les révisions sont associées à certaines ou à toutes les modifications mineures. Dans [Maurino, 93], les modifications mineures ou majeures sont réservées aux modèles déjà validés et utilisés dans l'entreprise alors que les termes révision et version sont réservées aux éventuelles reprises survenant au cours de la procédure de définition de ces modèles.

Nous proposons de définir les mécanismes de correction, révision et version, en s'inspirant des définitions de modification mineure/majeure proposées dans le §4, comme suit :

Article	Représentation
Version : changement de définition du produit qui n'assure pas son interchangeabilité Révision : changement de	Correction : modification éditoriale (modification de la forme du contenu telle qu'une faute d'orthographe) sans impact sur les objets qui dépendent de la représentation. Elle peut ou non être diffusée. Révision: modification sémantique (modification du sens du contenu) qui assure l'interchangeabilité de l'objet (initial et modifié) donc sans impact sur

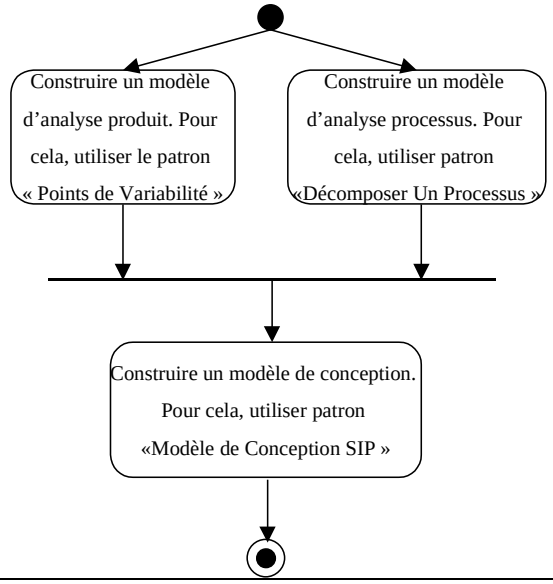
[175] Les définitions proposées dans la littérature pour les concepts de cas d'emploi et de cas d'utilisation sont centrées sur les nomenclatures organiques où un cas d'emploi est un composé d'un article et un cas d'utilisation est un composant d'un article. Nous étendons cette définition à tout type d'objet (article, représentation, fonction, …) et nous entendons dire par cas d'emploi (respectivement cas d'utilisation) un objet utilisant (respectivement *utilisé par*) l'objet concerné. Ils peuvent être de même nature que l'objet interchangeable ou de nature différente.

définition du produit qui assure son interchangeabilité	l'ensemble des cas d'emploi de l'objet. Une diffusion de la nouvelle révision à une faible échelle suffit. Version : modification sémantique qui n'assure pas l'interchangeabilité de l'objet (initial et modifié) donc ayant un impact sur au moins un cas d'emploi de l'objet. Une diffusion de la nouvelle version à une large échelle est souvent nécessaire.

Tableau 2 - Version, Révision et Correction

Annexe D : Les Patrons du Catalogue SIP

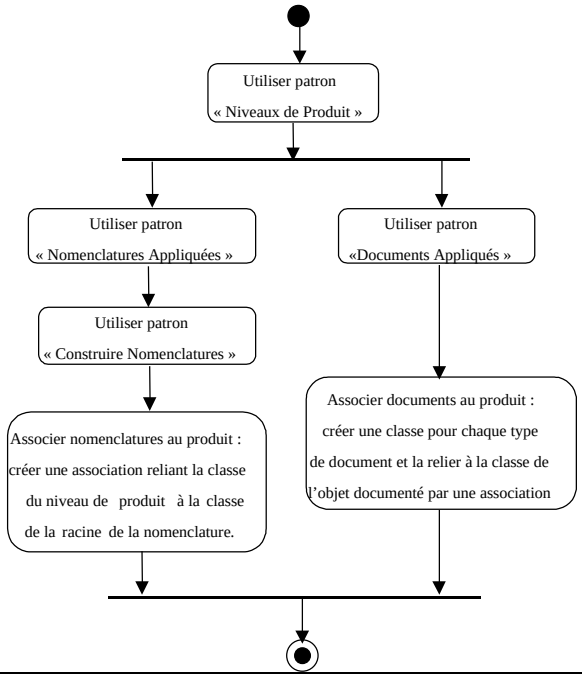
Point d'entrée du catalogue : le patron "Spécifier un SIP"

Nom	Spécifier un SIP
Problème	Guider le concepteur du SIP dans l'utilisation des divers patrons du catalogue SIP.
Motivation	Une démarche de spécification de SIP suit un ensemble plus ou moins ordonné d'étapes. Ainsi, pour construire un modèle de conception illustrant les objets mis en collaboration dans le SII nécessite de connaître les objets métiers du SIO ainsi que les traitements attendus du SII. Ces derniers ne peuvent être identifiés qu'à l'issue de la description des processus métier du SIO. Par ailleurs la description des processus métiers du SIO peut nécessiter la connaissance des objets métiers mis en jeu dans le SIO.
Force	Ce patron guide l'analyse et la conception du SIP d'une façon rigoureuse.
Contexte	Ce patron ne nécessite aucun autre patron ou modèle pour être appliqué.
Solution-Démarche	4. identifier les objets métiers associés au produit et gérés dans le SIP. Les structurer dans un modèle d'analyse produit. Pour cela appliquer le patron "Points de Variabilité" 5. représenter les processus gérés dans le SIP. Pour cela, appliquer le patron "Décomposer Un Processus". Pour illustrer les objets entrants\sortants des activités, les objets métiers associés au produit doivent être préalablement identifiés et décrits (patron "Points de Variabilité). 6. Construire le modèle de conception représentant les objets du systèmes d'information informatisé (SII) associé au SIO. Pour cela appliquer le patron "Modèle de Conception SIP". L'application de ce patron nécessite la description préalable des objets métiers associés au produit ainsi que les processus métiers du SIP. 
Solution-Modèle	A l'application de la solution-démarche, on obtient le modèle d'analyse produit, le modèle d'analyse processus et le modèle de conception du SIP.
Utilise	{Points de Variabilité, Décomposer Un Processus, Modèle de Conception SIP}

1. Patrons d'Analyse Produit

Patron "Points de Variabilité"

Nom	Points de Variabilité
Classification	Patron d'analyse produit.
Problème	Identifier tous les blocs du modèle SIP et fixer les points de variabilité à l'intérieur de chaque bloc selon la spécificité de l'entreprise.
Motivation	Chaque organisation gère des produits et associe diverses nomenclatures et des documents à ces produits. Toutefois, le nombre de niveaux d'abstraction de ces produits varient d'une entreprise à une autre. Certaines organisations gèrent des types de produits indépendants, d'autres gèrent en plus des produits génériques souvent appelés lignes ou gammes de produits. Dans ces deux cas, les exemplaires de produit peuvent ou non être gérés selon l'aspect critique des exemplaires produit et du service après-vente (dans l'industrie aéronautique par exemple, chaque exemplaire d'avion est géré tout au long de sa vie pour garder trace des conditions de sa fabrication, des opérations de maintenance, etc. alors que dans la production de masse tel que l'industrie de stylos, les exemplaires ne sont pas gérés). Par ailleurs, le nombre et le type de nomenclatures utilisées pour décrire le produit varient aussi d'une entreprise à l'autre. Dans certaines organisations, un type de produit a une seule nomenclature organique alors que dans d'autres organisations, on distingue la nomenclature d'étude, la nomenclature industrielle, la nomenclature logistique pour un type de produit. Il en est de même pour les documents dont le type (le contenu) et le nombre varient d'une entreprise à une autre.
Force	Ce patron guide la modélisation du SIP d'une façon rigoureuse et uniforme. Toutefois, il ne permet pas d'introduire de nouveaux concepts. L'apparition de nouveaux concepts exige une révision du catalogue de patrons et en particulier ce patron.
Contexte	Ce patron ne nécessite aucun autre patron ou modèle pour être appliqué.
Solution-Démarche	4. Fixer le nombre et le type de niveaux de produit (générique, type, exemplaire). Pour cela, appliquer le patron "Niveaux de Produit". 5. Pour les Nomenclatures : ▪ Fixer les types de nomenclatures appliquées. Pour cela, appliquer le patron "Nomenclatures Appliquées"; ▪ Construire les Nomenclatures décrivant votre produit. Pour cela, appliquer le patron "Construire Nomenclature"; ▪ Associer chaque Nomenclature au niveau de produit approprié. Pour cela, créer une association qui lie la classe du niveau de produit considéré à la classe qui correspond à la racine de la nomenclature considérée. 6. Pour les documents : ▪ Fixer le type de documents appliqués dans votre entreprise. Pour cela appliquer le patron "Documents Appliqués"; ▪ Associer les types de documents identifiés aux objets appropriés. Pour cela, créer une classe pour chaque type de document identifié et créer une association qui lie cette classe de document à l'objet qu'il documente (qui peut être un niveau donné de produit, un article, une fonction, un feature). Selon que le type de document est de nature "dépendant" ou "non-dépendant", associer un stéréotype à la classe associée (dépendant, non-dépendant) et baptiser l'association par "décrit" (cardinalité 1 du coté de la classe associée à l'objet documenté) ou "documente" (cardinalité 0..* du coté de la classe associée à l'objet documenté).

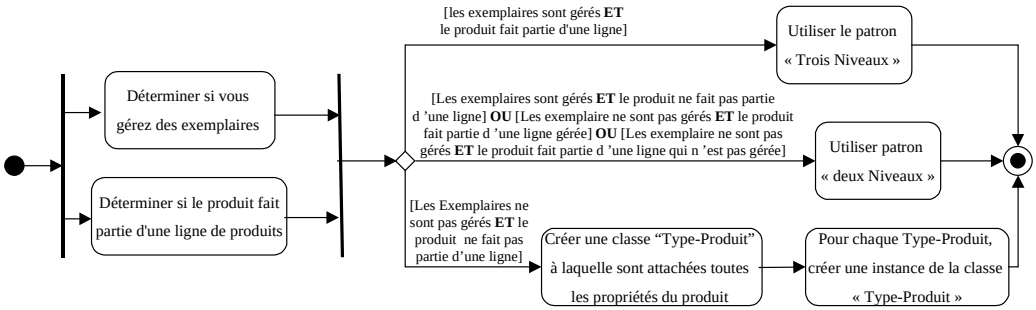
	La figure suivante illustre l'ensemble de la démarche : 
Solution-Modèle	A l'application de la solution-démarche, on obtient un modèle d'analyse produit complet.
Utilise	{Niveaux de Produit, Nomenclatures Appliquées, Construire Nomenclature, Documents Appliqués}

Patron "Niveaux de Produit"

Nom	Niveaux de Produit.
Classification	Patron d'analyse produit.
Problème	Il existe trois niveaux de produit possibles dans un SIP (produit-physique, type-produit et produit-générique). Le patron "Niveaux Produit" permet d'identifier le nombre de niveaux de produit et propose une solution pour les représenter.
Motivation	Le produit "voiture" prend différents aspects. Nous distinguons trois aspects : le produit-physique (c'est le produit livré au client, l'objet physique obtenu à la sortie du processus de fabrication, tel que *ma peugeot-206-S16*), le type-produit (c'est le modèle théorique selon lequel les exemplaires sont fabriqués, tel que le type *peugeot-206-S16*) et le produit-générique (un modèle théorique qui inclue toutes les options et variantes selon lesquelles divers types-produit sont déclinés, tel que la gamme des *peugeot-206*).

Niveau de produit	Produit générique (Générique)	Type-Produit (Type)	Exemplaire-Produit (Exemplaire)
Caractéristiques	Nom PG = peugeot 206 Moteur = thermique ou à essence ou électrique Couleur = vert ou rouge ou bleu	Nom TP = peugeot 206-S16 Moteur = à essence Nb. soupape = 2 Nb. valves = 24 Couleur = vert ou bleu	Nom EP = peugeot 206-S16 Moteur = à essence Nb. soupape = 2 Nb. valves = 24 Couleur = bleu N° série = S16.98.100

Les connaissances associées au produit varient ainsi selon le niveau considéré. Afin d'organiser ces connaissances, il est indispensable de distinguer trois niveaux afin de pouvoir affecter les connaissances aux niveaux appropriés.

Force	ce patron aide à distinguer les divers niveaux d'abstraction du produit et ensuite à construire le modèle correspondant. Il ne permet pas toutefois de considérer plus que trois niveaux de produit.
Contexte	ce patron ne nécessite aucun autre patron ou modèle pour être appliqué
Solution-Démarche	Une organisation a toujours un ou plusieurs types de produit (par exemple le produit peugeot 206-S16). Afin d'identifier les autres niveaux, il faut répondre aux questions suivantes : 1. *Le SIP gère-t-il des exemplaires physiques ?* Cas A : oui. Les niveaux type-produit et produit-physique sont gérés. Cas B : non. Seul le niveau type-produit est géré. *2. Le type-produit fait-il partie d'une gamme de produits ? Si oui, les gammes sont-ils gérés dans le SIP ?* Cas C: oui. Les niveaux type-produit et produit-générique existent et les gammes de produit sont gérés. Cas D : oui. Les niveaux type-produit et produit-générique existent mais les gammes de produit ne sont pas gérés. Cas E: non. Seul le niveau type-produit est géré. **Synthèse :** 1. Si les cas **A** et **C** sont vrais, trois niveaux de produit sont gérés dans le SIP (générique, type et exemplaire). Appliquer le patron "Trois Niveaux". 2. Si les cas **A** et **D** sont vrais ou les cas **A** et **E** sont vrais, deux niveaux de produit sont gérés (type et exemplaire). Appliquer le patron "Deux Niveaux". 3. Si les cas **B** et **C** sont vrais, deux niveaux sont gérés (type et générique). Appliquer le patron "Deux Niveaux". 4. Si les cas **B** et **D** sont vrais ou les cas **B** et **E** sont vrais, un seul niveau est géré (type). Créer une classe "Type-Produit" qui détient toutes les propriétés du type-produit. Ainsi, chaque produit particulier est une instance de la classe "Type-Produit". 
Solution-Modèle	le modèle obtenu est un diagramme de classe avec une, deux ou trois classes représentant respectivement un, deux ou trois niveaux de produit.
Utilise	{Deux Niveaux de Produit, Trois Niveaux de Produit}

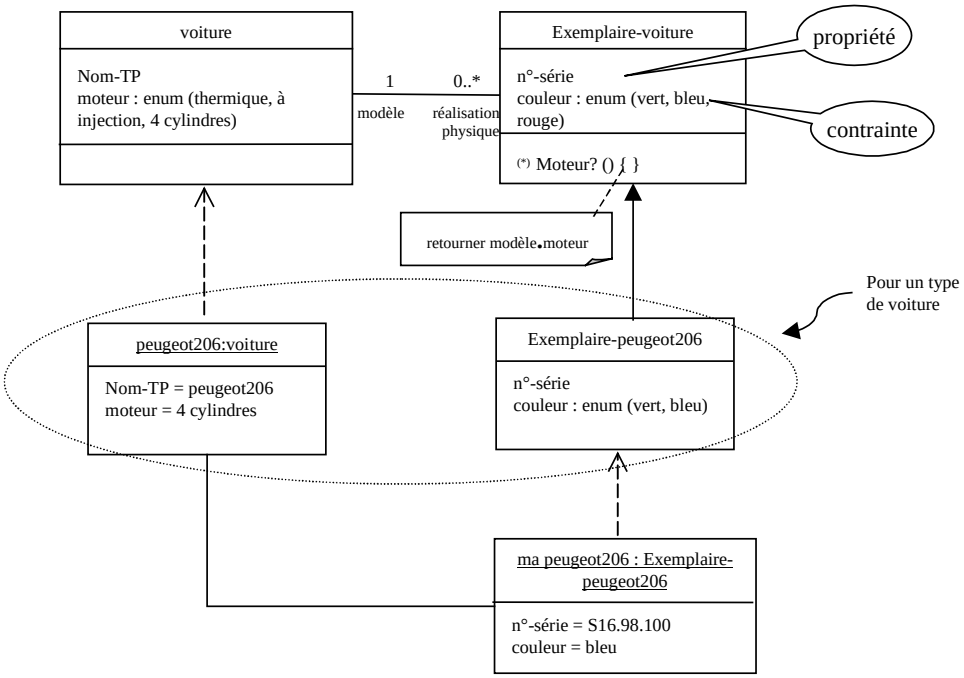
Patron "Deux Niveaux de Produit"

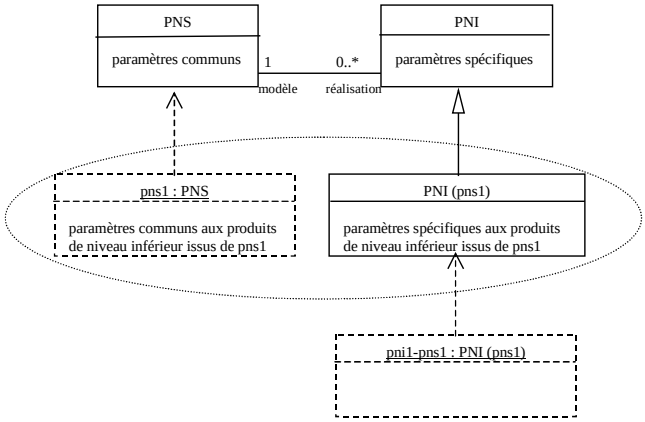
<table>
<tr><td>Nom</td><td>Deux Niveaux de Produit.</td></tr>
<tr><td>Classification</td><td>Patron d'analyse produit.</td></tr>
<tr><td>Problème</td><td>Ce patron permet de représenter le concept de produit en deux niveaux d'abstraction permettant d'une part de partitionner les connaissances relatives aux différents niveaux et d'autre part de définir des relations entre les niveaux afin de permettre la propagation de propriétés entre les niveaux.
Ce patron peut être appliqué pour représenter aussi bien le cas des deux niveaux produit-générique & type-produit que le cas des deux niveaux type-produit & produit-physique.</td></tr>
<tr><td>Motivation</td><td>Nous considérons l'exemple suivant pour le produit "voiture". Nous illustrons le cas des deux niveaux produit-physique et type-produit. Nous supposons ici qu'il n'existe pas de gammes de produits. Par exemple, il n y a qu'un seul type de peugeot 206.

<table>
<tr><th>Niveau de produit</th><th>Peugeot 206 (type)</th><th>Ma peugeot 206 (exemplaire)</th></tr>
<tr><td>caractéristiques</td><td>Nom TP = peugeot 206
Moteur = 4 cylindres
Couleur = vert ou bleu</td><td>Nom EP = peugeot 206
Moteur = 4 cylindres
Couleur = bleu
N° série = 206-98-100</td></tr>
</table>

Les connaissances décrites dans chacune de ces deux entités sont de deux natures:
▪ Propriété d'entité (une entité prend une valeur pour chacune de ses propriétés), telle que la propriété moteur = 4 cylindres pour le type-produit Peugeot 206 ou couleur = bleu pour l'exemplaire Ma peugeot 206. Une valeur d'une propriété, existant à un niveau donné doit être visible au niveau inférieur. Ainsi, la propriété moteur = 4 cylindres de Peugeot 206 est visible dans Ma peugeot 206.
▪ Contrainte exprimée à un niveau donné et portant sur les valeurs d'une propriété du niveau inférieur, telle que la contrainte couleur = vert ou bleu, exprimée dans Peugeot 206 et contraignant la valeur de la propriété couleur de Ma peugeot 206 (couleur = bleu).
Le diagramme de classes ci-dessous organise les connaissances rattachées aux deux niveaux de produit ci-dessus décrits.
▪ La classe "voiture" détient les propriétés de tous les types de voiture. Peugeot206 est une instance de cette classe.
▪ La classe "exemplaire-voiture" détient les propriétés de tous les exemplaires de voitures.

Pour chaque type de voiture, par exemple le type peugeot 206, on définit une sous-classe de "exemplaire-voiture" (la classe "exemplaire-peugeot206") détenant les propriétés spécifiques et les contraintes des exemplaires de ce type (des exemplaires de peugeot206). Tout exemplaire d'un type donné de voiture (par exemple ma peugeot206) est une instance de cette sous-classe.</td></tr>
</table>

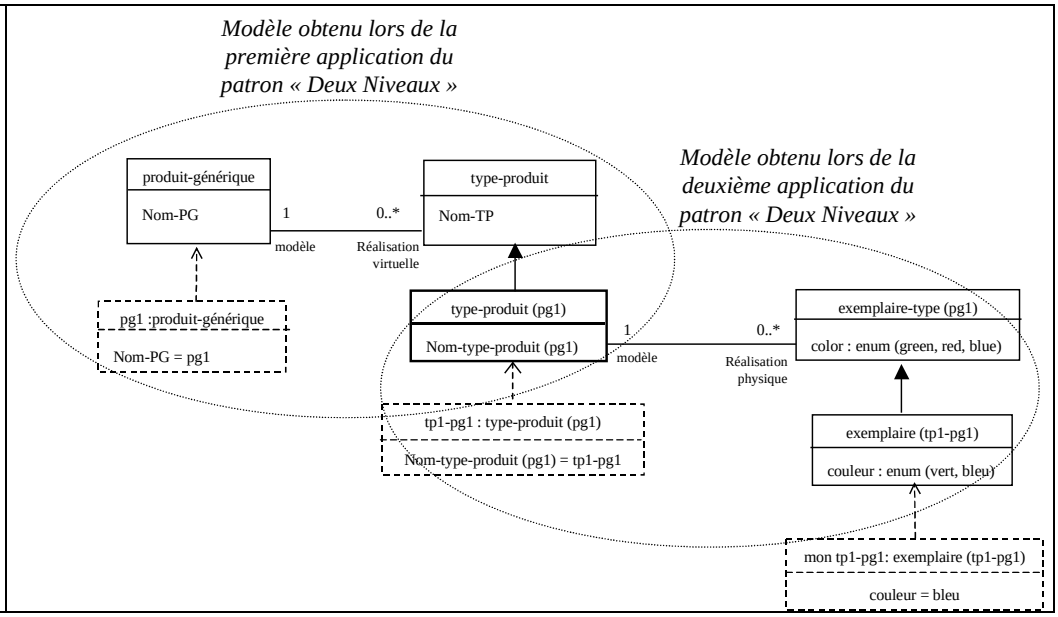
	 *peugeot206* (respectivement *ma peugeot206*) est une instance de la classe "Voiture" (respectivement "Exemplaire-peugeot206"). Les associations entre les niveaux permettent de rattacher chaque exemplaire à son type de voiture. Les propriétés d'un niveau restent visibles au niveau inférieur en définissant des méthodes de consultation permettant de propager une valeur à travers les associations (cf. méthode *Moteur?()*)
Force	Ce patron a les mêmes forces que le patron "Item-description" de Coad ou encore le patron "Materialization" de Dahchour. Il relie une classe de catégories abstraites A (modèles de voitures) à une classe d'objets plus concrets C (voitures individuelles). Sa sémantique est définie en terme des abstractions "est-un" (généralisation) et "est-de" et des correspondances classe/métaclasse.
Contexte	Ce patron n'exige aucun autre patron ou modèle pour être appliqué. Cependant, le concepteur doit déjà connaître le nombre de niveaux de produit. Ce nombre doit être de 2.
Solution-Démarche	4. Une entreprise articule ses activités autour de deux niveaux d'abstraction du produit. Deux classes abstraites interviennent : ▪ Produit Niveau Supérieur "PNS", détenant les propriétés communes des produits de niveau inférieur dérivant de celui-ci et contenant une ensemble de valeurs possibles de certaines propriétés. ▪ Produit Niveau Inférieur "PNI", détenant les propriétés spécifiques de produits de niveau inférieur associés à un produit de niveau supérieur et contenant des valeurs fixes de certaines propriétés. L'association entre ces deux classes a pour cadinalités : 1 du côté de la classe "PNS" et 0..* du côté de la classe PNI. Deux scénarios peuvent exister : "PNS" est un produit-générique & "PNI" est un type-produit ou "PNS" est un type-produit & "PNI" est un produit-physique. 5. Quand l'entreprise décide de créer un nouveau "Produit Niveau Supérieur" (par exemple le nouveau type de produit *peugeot 206*) soit *pns1*. *pns1* est représenté par une instance de la classe "PNS". Ceci se traduit par l'objet *pns1:PNS*. 6. Avec la création du nouveau produit *pns1*, il y aura différents produits de niveau inférieur

	dérivant de *pns1* (par exemple différents exemplaires de *peugeot 206*). On crée alors une classe concrète pour tous les produits de niveau inférieur dérivant de *pns1*, soit la classe "PNI (pns1)". Cette classe contient les propriétés spécifiques aux produits de niveau inférieur dérivant de pns1.
Solution-Modèle	La création d'un nouveau PNS, soit pns1 (par exemple peugeot206), implique la création de divers produits de niveau inférieur issus de pns1 (divers exemplaires de peugeot 206).  Toute création d'un nouveau produit de niveau supérieur génère une instance de la classe "PNS" et une sous-classe de la classe "PNI".

Patron " Trois Niveaux de Produit"

Nom	Trois Niveaux de Produit.
Classification	Patron d'analyse produit.
Problème	Ce patron permet de représenter le concept de produit en trois niveaux d'abstraction (produit-générique, type-produit et produit-physique). Il permet d'une part de partitionner les connaissances relatives aux différents niveaux et d'autre part de définir des relations entre les niveaux afin de permettre la propagation de propriétés entre les niveaux.
Motivation	Nous considérons l'exemple suivant pour le produit "voiture". Nous distinguons trois niveaux de produit : produit-générique, type-produit et produit-physique. **Niveau de produit** / Peugeot 206 (générique) / Peugeot 206-S16 (type) / Ma peugeot 206-S16 (exemplaire) **caractéristique** / Nom-PG = peugeot 206 Moteur = 4 cylindres ou électrique Couleur = vert ou bleu ou rouge / Nom TP = peugeot 206-S16 Moteur = 4 cylindres Couleur = vert ou bleu / Nom EP = peugeot 206-S16 Moteur = 4 cylindres Couleur = bleu N° série = 206-98-100 Les connaissances associées à ces trois entités (produit-générique, type-produit et produit-physique) dépend du niveau d'abstraction (voir motivation du patron "deux Niveaux de Produit").
Force	ce patron a les mêmes forces que le patron "Item-description" de Coad. Il relie une classe de catégories abstraites A (modèles de voitures) à une classe d'objets plus concrets C (voitures individuelles). Sa sémantique est définie en terme des abstractions "est-un" (généralisation) et "est-de" et des correspondances classe/métaclasse.
Contexte	Ce produit n'exige aucun autre patron ou modèle pour être appliqué. Cependant, le concepteur

	doit déjà connaître le nombre de niveaux de produit. Ce nombre doit être de 3.
Solution-Démarche	Une entreprise articule ses activités autour de trois niveaux d'abstraction du produit : ▪ le niveau "produit-générique", détenant les propriétés des produits génériques (tel que "nom-PG) ▪ le niveau "type-produit", détenant les propriétés des types de produits issus de divers produits génériques (tel que Nom-TP). ▪ Le niveau " produit-physique", détenant les propriétés des exemplaires de divers types de produits (tel que n° de série, date-fabrication) Le niveau "produit-générique" est plus haut que le niveau "type-produit", lui-même plus haut que "produit-exemplaire". 4. Appliquer le patron "Deux Niveaux" avec le "produit-générique" comme classe "PNS" et le "type-produit" comme classe "PNI". - Une instance de la classe "PNS" est dans ce cas un nouveau produit-générique (correspondant à la création d'une nouvelle ligne de produit), qu'on nomme *pg1* (par exemple le produit générique peugeot 206). - Avec la création du nouveau produit générique pg1, il y aura différents types de produits dérivant de *pg1* (par exemple différents types de *peugeot 206*). La classe concrète "type-produit (pg1)" représente l'ensemble de ces types de produits issus de *pg1*. 5. Appliquer encore une fois le patron "Deux Niveaux" avec le "type-produit (pg1)" comme classe "PNS" et les exemplaires de tous les types de produit issus de pg1 soit "exemplaire-type (pg1)" comme la classe "PNI". - Une instance de la classe PNS est dans ce cas un nouveau type de produit dans la ligne des produit pg1 (par exemple le type peugeot 206-S16 dans la ligne des produits peugeot 206), qu'on nomme *tp1-gp1*. - Avec la création du nouveau type de produit *tp1-pg1*, il y aura différents exemplaires de ce type de produit (par exemple différents exemplaires de *peugeot 206-S16*). La classe concrète "exemplaire (tp1-pg1)" représente l'ensemble de ces exemplaires de produits issus de *tp1-pg1*. 6. Assembler les deux modèles obtenus des deux phases précédente à l'aide de la classe commune entre ces deux modèle : la classe "type-produit(pg1)"
Modèle-Solution	Le modèle obtenu est le suivant :

Patron "Nomenclatures Appliquées"

Nom	Nomenclatures Appliquées.
Classification	Patron d'analyse produit.
Problème	Déterminer les nomenclatures pouvant être gérées dans l'entreprise, selon les niveaux de produit présents et définir les propriétés de chacune des nomenclatures.
Motivation	Chaque niveau de produit a des nomenclatures spécifiques. Ainsi le produit générique a une nomenclature organique générique alors que le type-produit a une nomenclature organique spécifique. Le produit-physique a par exemple une nomenclature physique. D'autre part, certaines nomenclatures ne sont pas propres à un niveau de produit donné; ils peuvent être rattachés à différents niveaux sans que leur construction soit modifiée. Toutefois, selon le nombre de niveaux présents, ils se voient rattachés à un niveau particulier. A titre d'exemple, la nomenclature fonctionnelle est la même quelque soit le niveau de produit. Elle est toutefois rattaché au niveau de produit supérieur existant. Elle est déduite pour les niveaux inférieurs par propagation de valeurs entre les niveaux. Ainsi, si le produit générique est géré, la nomenclature fonctionnelle est rattachée à ce niveau. Si le niveau générique n'est pas géré, la nomenclature est rattachée au niveau inférieur : le type-produit s'il existe, sinon le produit-physique.
Force	ce patron aide le concepteur à définir les diverses nomenclatures à gérer dans le SIP à construire. - Il fournit l'ensemble des nomenclatures pouvant être gérés dans le SIP, selon le type de niveaux de produit existants, parmi lesquels le concepteur choisit celles qui correspondent à son cas d'étude. - Il permet de caractériser chacune des nomenclatures possibles; ce qui aide le concepteur dans le choix du patron de construction de nomenclature approprié, selon les propriétés des nomenclatures qu'il considère. Ce patron ne permet pas toutefois de considérer d'autres types de nomenclatures, autres que celles indiquées.
Contexte	Pour appliquer ce patron, il faut savoir le nombre et le type de niveaux de produit gérés. Il

<table>
<tr><td></td><td>requiert l'application préalable du patron "Niveaux de Produit"</td></tr>
<tr><td>Solution-Démarche</td><td>

Types de nomenclatures gérées

si le niveau produit-générique existe, les nomenclatures qui peuvent lui être associées sont : nomenclature organique générique, nomenclature géométrique générique
si le niveau type-produit existe, les nomenclatures qui peuvent lui être associées sont : nomenclature organique type, nomenclature géométrique type
si le niveau produit-physique existe, les nomenclatures qui peuvent lui être associées sont : nomenclature organique physique.
La nomenclature fonctionnelle est souvent associée au niveau de produit le plus haut existant dans l'entreprise.

Attachement des Nomenclatures à chaque niveau de produit (en terme de classes)

4. Si le niveau produit-générique existe :

au produit-générique sont attachées la nomenclature organique générique, la nomenclature géométrique générique et la nomenclature fonctionnelle
au type-produit, s'il existe, aucune nomenclature ne lui est rattachée (car ses nomenclatures sont déclinées à partir des nomenclatures du produit générique à l'aide d'opérations sur les classes associés aux niveaux de produit)
au produit-physique, s'il existe, est associée la nomenclature physique

5. si le niveau produit-générique n'existe pas et que le type-produit existe :

au type-produit sont attachées la nomenclature organique type, la nomenclature géométrique type et la nomenclature fonctionnelle
au produit-physique, s'il existe, est associée la nomenclature physique

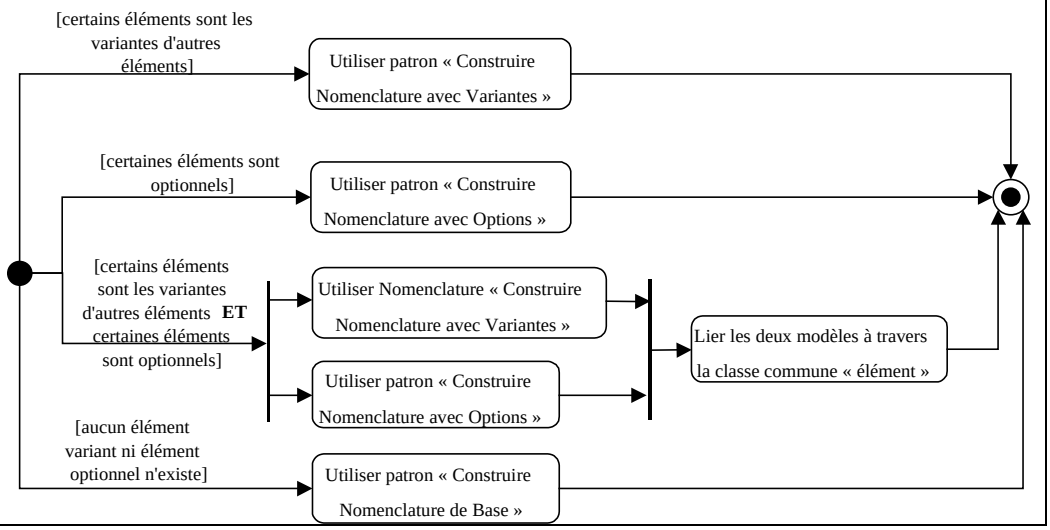
6. si seul le niveau produit-physique existe, alors :

au produit-physique sont associés la nomenclature organique physique et la nomenclature fonctionnelle

Propriétés des nomenclatures

Le tableau suivant associe à chaque type de nomenclature la nature des nœuds qui le constituent et la sémantique du lien de composition entre les nœuds.
<table>
<tr><td></td><td>N. dont certains nœuds sont des variantes d'autres nœuds</td><td>N. dont certains nœuds sont optionnels</td><td>N. à lien de composition exclusif</td><td>N. à lien de composition partagé</td><td>N. à lien de composition dépendant</td><td>N. à lien de composition indépendant</td></tr>
<tr><td>N. organique générique</td><td>*</td><td>*</td><td></td><td>*</td><td></td><td>*</td></tr>
<tr><td>N. organique type</td><td>*</td><td>*</td><td></td><td>*</td><td></td><td>*</td></tr>
<tr><td>N. organique physique</td><td></td><td></td><td>*</td><td></td><td>*</td><td></td></tr>
<tr><td>N. fonctionnelle</td><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td></tr>
<tr><td>N. géométrique générique</td><td>*</td><td>*</td><td></td><td>*</td><td></td><td>*</td></tr>
<tr><td>N. géométrique type</td><td>*</td><td>*</td><td></td><td>*</td><td></td><td>*</td></tr>
</table>
</td></tr>
</table>

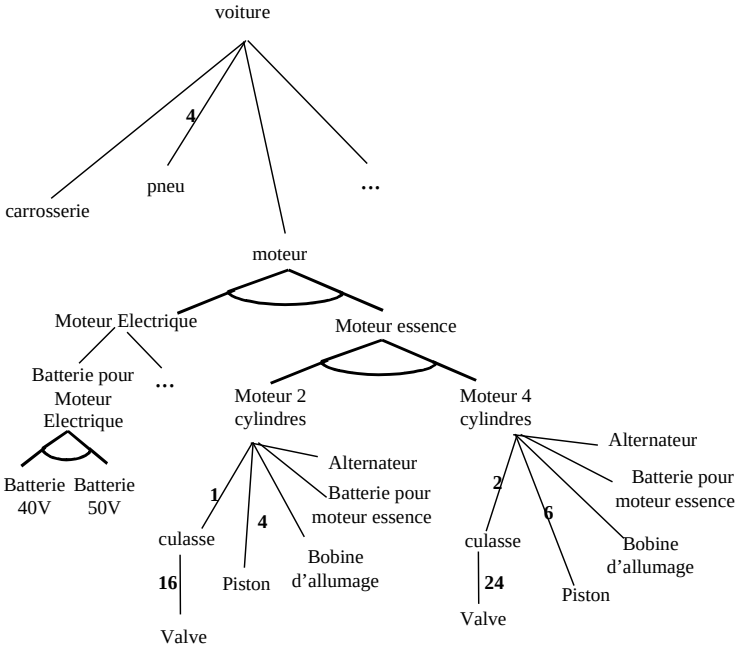
Patron " Construire Nomenclature"

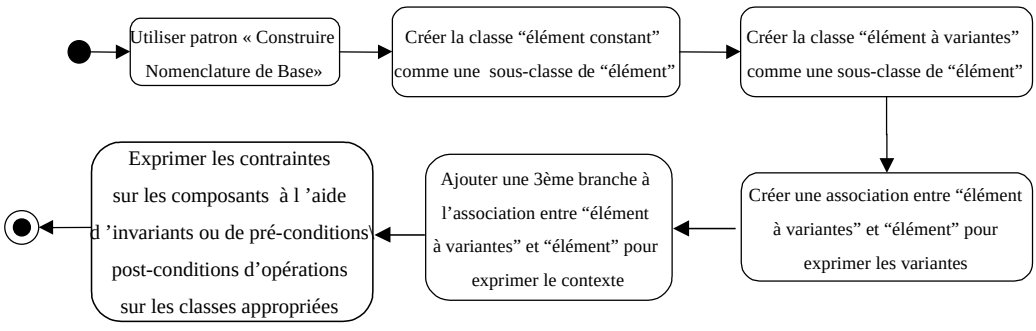
Nom	Construire Nomenclature
Classification	Patron d'analyse produit.
Problème	Ce patron permet de construire une nomenclature selon différentes caractéristiques de celle-ci.
Motivation	Le produit, à ses divers niveaux peut être structuré par plusieurs nomenclatures, organiques (organique générique, organique spécifique d'étude, organique spécifique industrielle, organique physique, etc.), fonctionnelles, géométriques (géométrique générique, géométrique spécifique), etc. La construction de ces nomenclatures ne dépend pas de la nature des constituants de la nomenclature (article, fonction, entité géométrique), mais du rôle joué par les constituants. Certains constituants sont à variantes et donc un ensemble de variantes leur est associé, possédant chacun une composition spécifique. D'autres constituants sont optionnels dans une nomenclature. A titre d'exemple, la nomenclature organique générique ou la nomenclature géométrique générique sont définis à l'aide de constituants obligatoires, optionnels et à variantes. La nomenclature organique physique par contre n'est constituée que d'articles obligatoires.
Force	Ce patron distingue diverses constructions de la nomenclature selon la nature des nœuds constituant la nomenclature.
Contexte	Ce patron n'exige aucun autre patron ou modèle pour être appliqué[176].
Solution-Démarche	▪ Si certains éléments de la nomenclature peuvent être les variantes d'autres éléments, alors appliquer le *patron "Construire Nomenclature avec Variantes"* ▪ Si certains éléments de la nomenclature peuvent être optionnels, alors appliquer le *patron "Construire Nomenclature avec Options"* ▪ Si certains éléments de la nomenclature peuvent être les variantes d'autres éléments et que certains éléments peuvent être optionnels dans la nomenclature, alors : ▪ Appliquer le *patron "Construire Nomenclature avec Variantes"* ▪ Appliquer le *patron "Construire Nomenclature avec Options"* ▪ Lier les deux modèles obtenues par application des deux patrons précédents à travers la classe communes aux deux modèles : "élément". ▪ S'il n'existe ni éléments variantes ni éléments optionnels, alors appliquer le *patron "Construire Nomenclature de Base"*. 

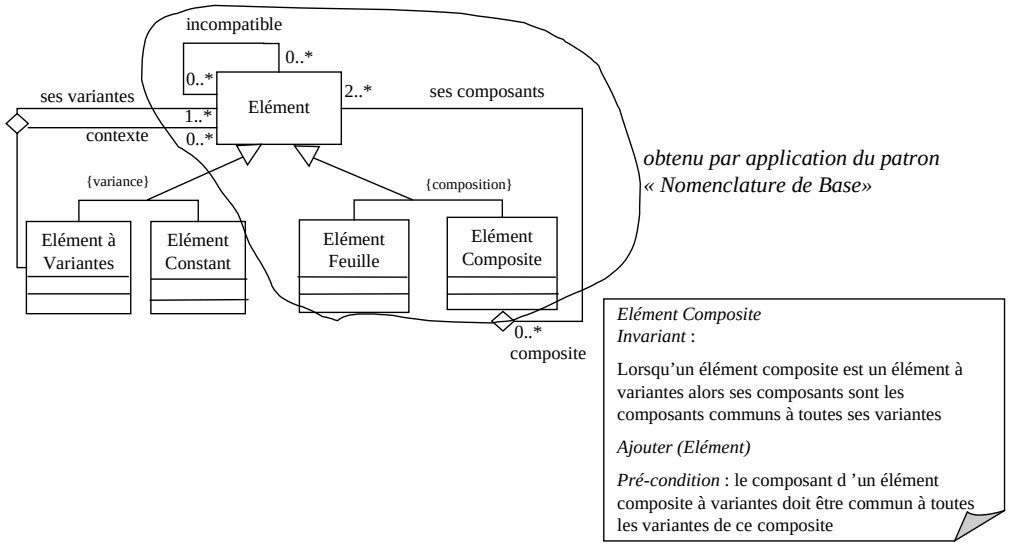
[176] Le concepteur n'est pas obligé par ailleurs d'avoir déjà appliquer le patron "Nomenclatures Appliquées".

Solution-Modèle	le diagramme de classe obtenu représente un modèle d'une nomenclature particulière.
Utilise	{Construire Nomenclature avec Variantes, Construire Nomenclature avec Options, Construire Nomenclature de Base}

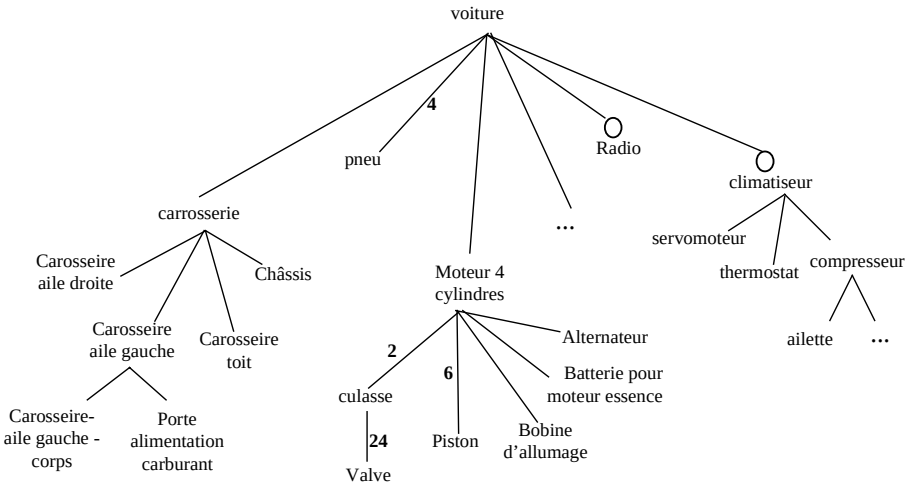
Patron "Nomenclatures avec Variantes"

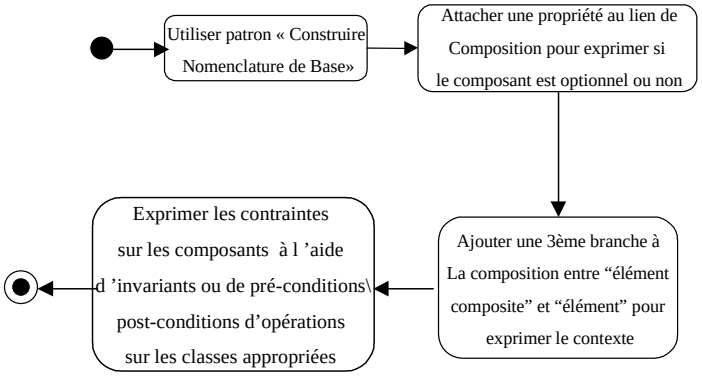
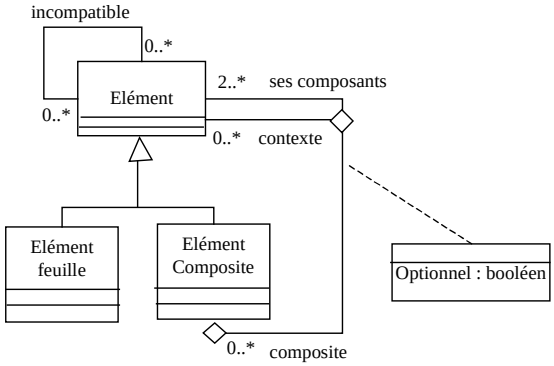
Nom	Nomenclature avec Variantes.
Classification	Patron d'analyse produit.
Problème	Ce patron permet de construire une nomenclature dont certains éléments sont les variantes d'autres éléments de la même nomenclature.
Motivation	Considérons le produit voiture. La composition du produit générique peut être en partie comme suit :  Le composant moteur est un articles à variantes. Une voiture possède soit un moteur électrique soit un moteur essence. Par ailleurs, divers types de moteur essence existent; celui-ci peut être à 2 cylindres ou à 4cylindres. Le composant moteur essence qui était une variante de moteur est elle même un article à variantes. Chacune des variantes a une composition spécifique. Ainsi, le moteur électrique est composé d'une batterie pour moteur électrique, etc. alors que les moteurs essence sont tous composés d'alternateur, de batterie pour moteur essence, de bobine d'allumage, de piston et de culasse. Selon que le moteur essence soit à 2 cylindres ou à 4 cylindres, le nombre de chacun des ces composants varie. Ainsi, dans une nomenclature à variantes, il peut exister une hiérarchie de variantes (une variante est elle même un article à variantes et possède ainsi des variantes). Chacune des variantes d'un composant donné possède sa propre composition. Toutefois, certains composants des variantes sont communs à l'ensemble des variantes (tel que le composant "alternateur" de moteur 4 cylindres et moteur 2 cylindres).

Force	Le modèle de nomenclature avec variantes permet de : - gérer une hiérarchie de variantes de façon à pouvoir éliminer automatiquement certaines variantes lorsque les variantes de niveau supérieur ne sont pas retenues. - Exprimer le contexte dans lequel un élément est une variantes d'un autre élément. Ceci permet d'associer à une variante donnée tous les autres variantes qui lui sont compatibles. - Exprimer la décomposition d'un élément à variantes (qui a différentes variantes) aussi bien au niveau de cet élément à variantes (on n'exprime alors que les composants communs à toutes ses variantes) qu'au niveau de ses variantes (on exprime les composants spécifiques à chaque variante)
Contexte	Ce patron n'exige aucun autre patron ou modèle pour être appliqué.
Solution-Démarche	3. Appliquer le patron " Nomenclature de Base" 4. Le modèle de nomenclature obtenu représente la classification des éléments selon le critère de décomposition (élément feuille ou composite). L'existence d'éléments variantes introduit une seconde classification selon leur variabilité. Ainsi, un élément peut être à variantes ou constant. Puisque un élément peut être feuille ou composite et en même temps peut être à variantes ou constant, alors la double classification devrait être complète et disjointe. Une autre branche relative à la classification selon la variabilité est alors introduite dans le modèle de nomenclature, comme suit: ▪ Créer la classe "élément constant" comme une sous-classe de "élément" ▪ Créer la classe "élément à variantes" comme une sous-classe de "élément" ▪ Puisque les variantes d'un "élément à variantes" sont d'autres "éléments" qui eux même peuvent être à variantes ou constants et feuilles ou composites, alors créer une association entre les classes "élément à variantes" et "élément" qui associe à chaque "élément à variantes" ses variantes. Cette association a comme rôle *ses variantes* du côté de la classe "élément". Les cardinalités de l'association sont de 1..* du côté de "élément" et 0..* du côté de "élément à variantes". ▪ Un élément est défini comme une variante d'un certain élément à variantes dans un contexte particulier (le composant "moteur diesel" est variante de "moteur" dans le produit "peugeot 206" mais il est un élément constant dans la composition d'un "groupe électrogène"). Ainsi, l'association précédemment créée entre "élément" et "élément à variantes" doit expliciter le contexte de telle variance. Ce contexte est un "élément" également (le composite dans lequel il est composant). Ajouter une troisième branche dans l'association reliant encore une fois la classe "élément" et dont le rôle est *contexte*. La cardinalité de cette branche est de 0..* ▪ Exprimer les contraintes éventuelles sur les composants à l'aide d'invariants ou de pré-conditions \post-conditions sur les opérations des classes appropriées. 

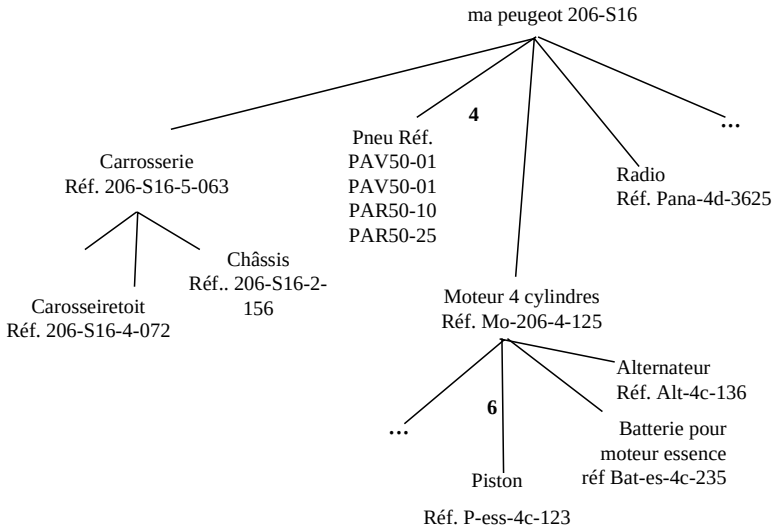
Solution-Modèle	Le modèle obtenu a une forme semblable à la suivante: 
Utilise	{Nomenclature de Base}

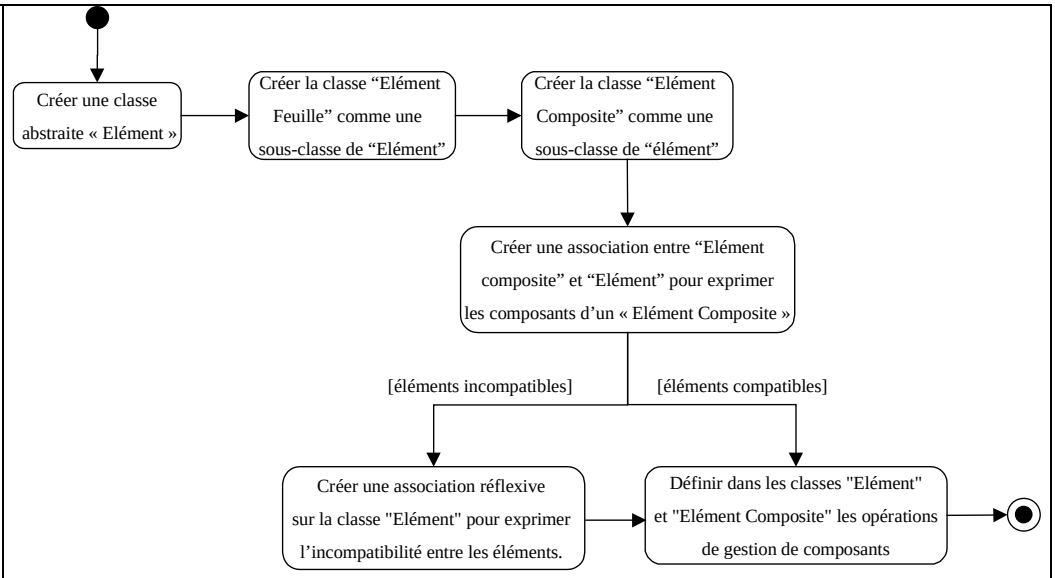
Patron "Nomenclature avec Options"

Nom	Nomenclature avec options.
Classification	Patron d'analyse produit.
Problème	Ce patron permet de construire une nomenclature dont certains éléments sont optionnels dans la nomenclature.
Motivation	Considérons le produit voiture. La composition du produit générique peut être en partie comme suit :  La voiture est composé nécessairement d'une carrosserie, d'un moteur, de pneus et d'un ensemble d'autres éléments nécessaires (volant, boite de vitesse, etc.). certains composants sont toutefois optionnels et sont mis en place dans la voiture selon le modèle de la voiture, selon les souhaits des clients, etc. Un élément optionnel peut être un composite et donc possède sa propre composition.
Force	Le modèle de nomenclature avec options permet de :

	- exprimer le contexte dans lequel un élément est optionnel dans une structure donnée. Ceci permet d'associer à une option donnée toutes les autres options qui lui sont compatibles ainsi que les variantes qui lui sont compatibles, en cas de produit à options et à variantes. - exprimer qu'un composant d'un élément composite est optionnel ou obligatoire.
Contexte	ce patron n'exige aucun autre patron ou modèle pour être appliqué.
Solution-Démarche	5. Appliquer le patron "Construire Nomenclature de Base". 6. Le modèle de nomenclature obtenu représente la composition récursive d'éléments (élément feuille ou composite). L'existence d'éléments optionnels introduit des propriétés supplémentaires sur le lien de composition entre un élément composite et ses composants. Attacher alors une propriété au lien pour exprimer si le composant est optionnel ou non. 7. Un élément est défini comme optionnel dans uns structure dans un contexte particulier (le composant "climatiseur" est optionnel dans le produit "peugeot 206" mais il est un élément obligatoire dans la composition d'un "véhicule frigorifique"). Ainsi, le lien de composition entre "élément composite" et "élément" doit expliciter le contexte de l'option. Ce contexte est également un "élément" (le composite dans lequel il est composant). Ajouter alors une troisième branche dans le lien de composition reliant encore une fois la classe "élément composite" à "élément" et dont le rôle est *contexte*. La cardinalité de cette branche est de 0..*. 8. Exprimer les contraintes éventuelles sur les composants à l'aide d'invariants ou de pré-conditions et de post-conditions sur les opérations des classes appropriées. 
Solution-Modèle	Le modèle obtenu a une forme semblable à la suivante : 
Utilise	{Nomenclature de Base}

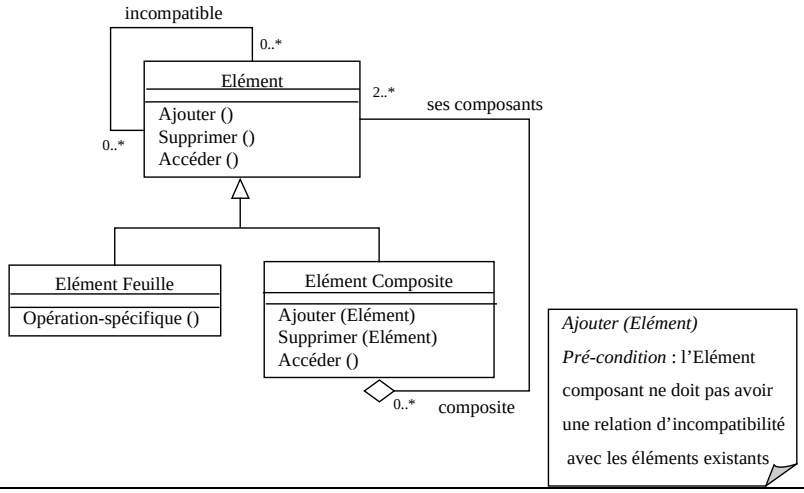
Patron "Nomenclature de Base"

Nom	Nomenclature de base.
Classification	Patron d'analyse produit.
Problème	Ce patron permet de construire une composition récursive d'éléments.
Motivation	Considérons un exemplaire individualisé d'une voiture, par exemple ma peugeot 206-S16. La composition de cet exemplaire est en partie comme suit :  L'exemplaire de voiture étant un choix définitif de composants, déjà mis en œuvre sur les lignes d'assemblage. Tous les composants sont obligatoires.
Force	Le modèle de nomenclature de base permet de définir des hiérarchies d'éléments simples et d'éléments composites et facilite l'ajout de nouveaux composants. Il a les mêmes forces que le patron de conception "Composite" d'E. Gamma.
Contexte	Ce patron n'exige aucun autre patron ou modèle pour être appliqué.
Solution-Démarche	7. Créer une classe abstraite "Elément". Cette classe définit l'interface commune des éléments composites et des éléments feuilles. 8. Créer une sous-classe de "Elément" qui s'appelle "Elément Feuille". Cette sous-classe regroupe tous les éléments qui ne sont pas décomposables. 9. Créer une sous-classe de "Elément" qui s'appelle "Elément Composite". Cette classe regroupe les éléments qui sont décomposables en d'autres éléments de même nature. 10. Créer une relation de composition entre les classes "Elément Composite" et "Elément". Cette composition associe à chaque "Elément Composite" ses composants dans la classe "Elément". Cette composition a pour cardinalité 2..*du côté de la classe "Elément". 11. Dans le cas où certains éléments ne sont pas compatibles entre eux, créer une association réflexive sur la classe "Elément" qui associe à chaque "Elément" les autres "Eléments" qui ne lui sont pas compatibles. Cette association est optionnelle. 12. Définir dans les classes "Elément" et "Elément Composite" les opérations de gestion de composants (accéder, supprimer, ajouter un composant).



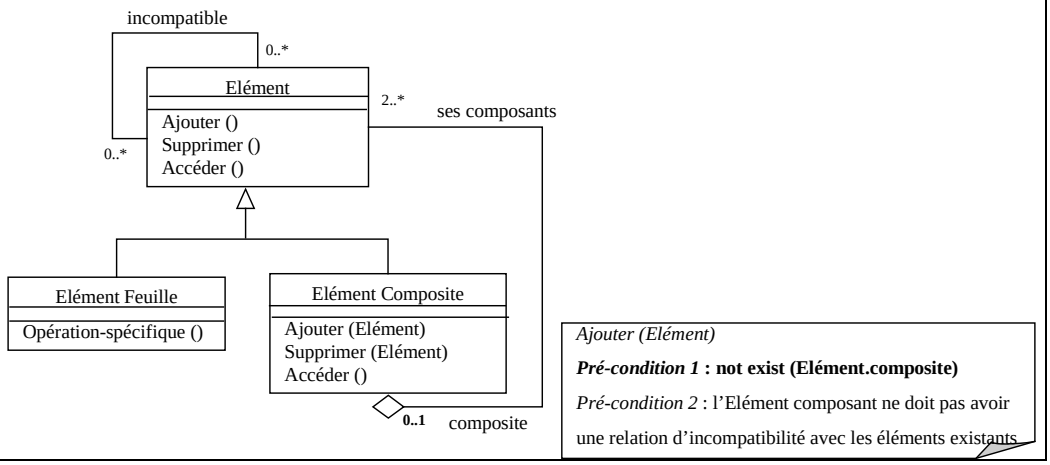
Solution-Modèle

Le modèle obtenu est le suivant :



Patron " Nomenclature de Base avec lien de composition exclusive "

Nom	Nomenclature de Base avec lien de composition exclusive.
Classification	Patron d'analyse produit.
Problème	Ce patron permet de construire une composition récursive et exclusive d'éléments.
Motivation	Dans la nomenclature d'un exemplaire particulier d'une voiture (*ma peugeot 206-S16* par exemple), le *piston réf. P-ess-4c-123* ne peut pas appartenir en même temps au moteur d'un autre exemplaire de voiture (sa peugeot 206-S16).
Force	Le modèle de nomenclature permet de définir des hiérarchies d'éléments simples et d'éléments composites liés entre eux par des liens de composition exclusives.
Contexte	Ce patron n'exige aucun autre patron ou modèle pour être appliqué.
Solution-Démarche	4. Construire une composition récursive d'éléments. Pour cela, appliquer le patron "Nomenclature de Base". 5. Restreindre la cardinalité de 0..* à 0..1 pour le rôle *composite* du lien de composition. 6. Ajouter à l'opération *Ajouter (Elément)* de la classe "composite", les informations suivantes:

	- pré-condition : not exist (Elément.composite) - post-condition : …
Solution-Modèle	

Patron "Documents Appliqués"

Nom	Documents Appliquées.
Classification	Patron d'analyse produit.
Problème	Déterminer les documents pouvant être gérées dans l'entreprise, selon les niveaux de produit présents et les éléments constitutifs du produit et définir les caractéristiques de chacun des documents.
Motivation	Chaque niveau de produit et chaque objet technique décrivant le produit (article, fonction, etc.) peut être documenté par un ensemble de documents. Certains de ces documents sont spécifiques à des niveaux donnés de produit. Ainsi, un produit-générique peut être documenté par un cahier des charges, selon lequel il est conçu. Un type-produit possède par exemple des dessins de définition, des modèles CAO, des notes de calcul, des gammes de fabrication et des instructions de fabrication. Un produit-physique possède par exemple une fiche de contrôle, un rapport d'essai, un compte-rendu de fabrication, un planning d'entretien. D'autres documents sont partagés par plusieurs niveaux, tels que les notices d'utilisation, rattachés aux types-produit mais également à chacun des produits physiques. Par ailleurs, selon qu'un niveau de produit existe ou non, certains documents sont rattachés différemment aux niveaux. Ainsi, un cahier des charges est rattaché au type-produit si le produit-générique n'existe pas. Par ailleurs, chacun de ces documents peut être de deux types : - dépendant de l'objet qu'il documente : il sert à décrire l'objet qu'il documente et son existence est entièrement liée à l'existence de l'objet documenté, tel qu'un dessin de définition d'une pièce mécanique. Un document dépendant est rattaché à un seul objet. - indépendant : il est utilisé pour documenter l'objet qu'il documente et son existence n'est pas dépendante de l'existence de l'objet documenté, telle qu'une norme sur les produits. Un document non-dépendant peut être associé à plusieurs objets. Un document, qu'il soit dépendant ou non-dépendant est enfin soit un enregistrement, soit un modèle. Ainsi, un compte-rendu de contrôle est enregistrement alors qu'un instruction de fabrication est un modèle.
Force	ce patron aide le concepteur à définir les diverses documents à gérer dans le SIP à construire.

	- Il fournit l'ensemble des documents pouvant être gérés dans le SIP, selon le type de niveaux de produit existants et l'objet technique documentant le produit, parmi lesquels le concepteur choisit ceux qui correspondent à son cas d'étude. - Il permet de caractériser les documents identifiés par rapport aux objets qu'ils documentent (document dépendant ou document non-dépendant) mais également par rapport à leur nature intrinsèque (modèle ou enregistrement). Cela aide le concepteur dans l'activité d'association des documents aux divers objets qu'ils documentent (voir solution-démarche du patron "Points de Variabilité"). Ce patron ne permet pas toutefois de considérer d'autres types de documents, autres que ceux indiqués.
Contexte	Pour appliquer ce patron, il faut savoir les niveaux de produit gérés, ainsi que les divers objets techniques rattachés aux produit. Il requiert l'application préalable du patron "Niveaux de Produit" et "Nomenclatures Appliquées".
Solution-Démarche	▪ **Types de documents gérées** Différents types de documents peuvent être rattachés au produit aux niveaux de produit, tels que: - pour le produit-générique : cahier des charges - pour le type produit : modèle CAO, dessin de définition, note de calcul, instruction de fabrication - pour le produit-physique : notice d'entretien, dossier de fabrication, rapport de contrôle, notice d'utilisation, planning d'entretien Pour les articles, il est possible d'associer des documents semblables à ceux décrits pour le type produit. ▪ **Propriétés des documents** Chacun des documents du produit peut être un modèle ou un enregistrement (cela se traduit par un attribut de la classe associée). Chacun des documents peut être dépendant ou non-dépendant. Cela se traduit par des liens différents avec l'objet qu'il documente. Ainsi, un cahier des charges est un document non-dépendant, du type modèle. Un modèle CAO est un modèle, du type dépendant, un rapport de contrôle est un enregistrement du type dépendant.

2. Patrons d'Analyse Processus

Patron "Décomposer un Processus"

Nom	Décomposer un Processus.
Classification	Patron d'analyse processus.
Problème	Décomposer un processus global en sous-processus afin : - De représenter la répartition des tâches entre les acteurs concernés, - De représenter les points de décision et de synchronisation, - Distinguer les activités manuelles des activités informatisées.
Motivation	Prenons l'exemple du processus de ***Gestion des Modifications de produits*** chez notre partenaire industriel (Schneider Electric). Ce processus est récurrent et fait partie intégrante du cycle de vie d'un produit. L'organisation d'un tel processus doit permettre une gestion optimale de la modification tout en intégrant un minimum de mécanismes de contrôle de cette modification de

manière à ce que celle-ci, si elle est validée, soit appliquée sans aucun risque et après toutes les vérifications jugées nécessaires.
Un tel processus de gestion de modification de produit peut être découpé en activités de manière à le rendre modulaire, à mieux cerner les actions nécessaires pour le mener et, ainsi, à mieux définir les acteurs responsables de chacune de ces actions.

Nous préconisons trois critères de décomposition : le changement d'objectif, le changement d'acteur et le changement de type d'activité. Ainsi, nous pouvons décomposer le processus de gestion des modifications de la manière suivante :

4. Dans un premier temps, selon *l'objectif* en décomposant l'objectif global qui est « gérer processus de modification ». Le processus *Gestion des Modifications* est alors décomposé en les activités suivantes :
   - *Emission Demande de Modification* :.
   - *Examen demande :* le processus est initialisé par une émission d'une demande de modification qui est soumise à un gestionnaire du BEGT (Bureau d'Etudes Gestion Technique) pour examiner la pertinence de la demande et décide de la continuité du processus ou pas.
   - *Etude faisabilité :* si la demande est acceptée lors de l'étape précédente, elle est soumise aux responsables des Antennes Techniques (AT) des usines de production, pour une étude de faisabilité portant sur différents critères (techniques, financiers, …).
   - *Application :* si la demande est jugée faisable, elle est mise en application par les membres de l'AT.

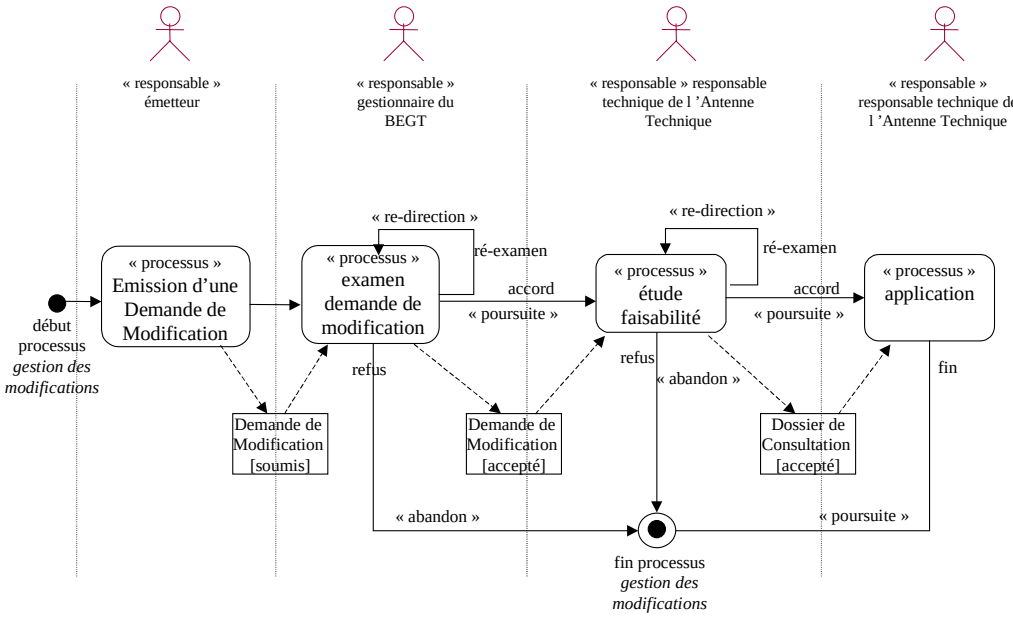


Figure a : Premier niveau de décomposition du processus de gestion de modifications de produits de Schneider

Remarque : Pour ne pas surcharger le modèle, nous n'avons illustré que quelques entrants/sortants dans la figure a.. Il convient de ne faire apparaître les entrants/sortants qu'à un niveau plus bas de décomposition, au niveau des opérations qui les transforment effectivement. Dans la suite de cet exemple, nous ne faisons pas figurer les entrants/sortants. Remarquons par ailleurs que le flot d'objets peut remplacer le flot de contrôle lorsque l'objet produit par une activité est utilisé immédiatement par une autre activité (voir objet "Demande de Modification"

ou "Dossier de consultation").

5. Les activités ainsi obtenues peuvent, lorsqu'elles sont de type 'processus', être découpées à leur tour en d'autres activités par affinement de leurs objectifs. Ainsi, le sous-processus "Etude faisabilité", dont l'objectif est d'étudier la faisabilité de la modification, peut être découpé comme suit :

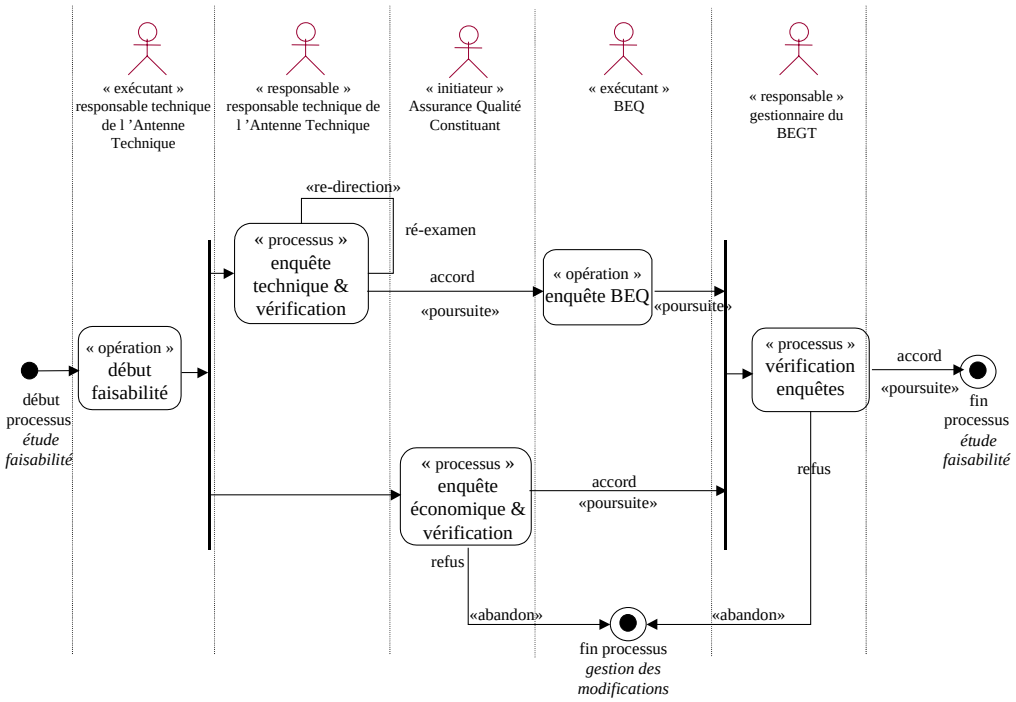


Figure b : Décomposition du sous-processus "Etude faisabilité" de Schneider

Remarque : pour une meilleure illustration, le lien de « re-direction » re-bouclant sur certaines activités dans la figure a (telles que le processus *étude faisabilité*) est représenté pour exprimer aussi bien :

- un re-bouclage sur l'ensemble du processus : un lien externe qui part de la dernière activité du processus et pointe sur sa première activité
- qu'un re-bouclage interne à la décomposition du processus : un lien interne de « re-direction » qui part d'une activité quelconque de niveau inférieur c'est à dire composant le processus et qui y pointe.

Ainsi, en décomposant le processus *Etude Faisabilité*, le lien de « re-direction » figurant sur cette activité dans la figure a ne pointe pas obligatoirement dans la figure b sur la première activité de ce processus et il ne part pas non plus de sa dernière activité. Nous permettons qu'il parte d'une activité interne à ce processus et y pointe (processus *enquête technique & vérification*).

6. La décomposition des sous-processus peut être poursuivie en gardant le même critère de décomposition, en l'occurrence l'objectif, ou en changeant de critère de décomposition, par exemple le changement d'*acteurs* ou de rôles d'acteurs. Ainsi, pour le processus « enquête technique & vérification » nous obtenons :

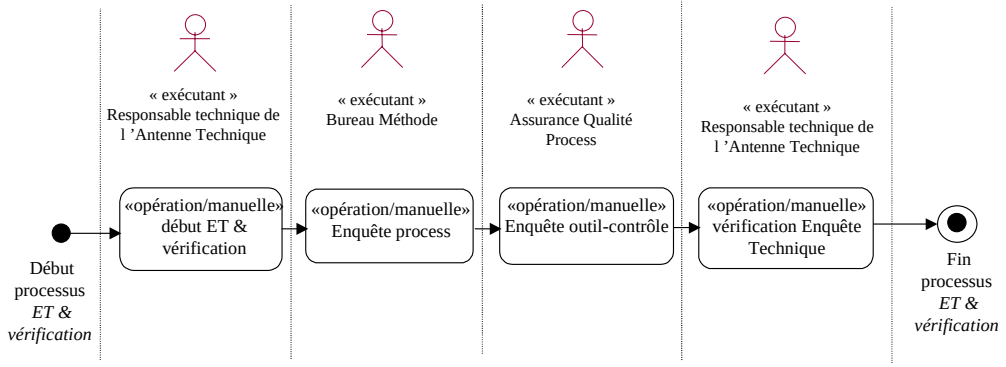


Figure c : Décomposition de l'activité "enquête technique & vérification" de Schneider

Force	Ce patron permet de décomposer un processus organisationnel selon une démarche générique, basée sur un ensemble de critères de décomposition indépendants de la spécificité des entreprises. Il permet en même temps de considérer dans la démarche proposée d'autres critères de décomposition, spécifiques à l'entreprise.
Contexte	Ce patron n'exige aucun autre patron ou modèle pour être appliqué.
Solution-Démarche	6. **Choisir un critère de décomposition.** Nous en préconisons trois : - *Le changement d'objectif :* l'objectif d'un processus peut être trop général et doit donc être réifié afin de permettre sa décomposition. - *Le changement d'acteur ou de rôle d'acteur :* c'est un bon indicateur quant au passage d'une activité à une autre, surtout lorsque c'est le responsable qui change. - *Le changement de type d'activité*[177] *:* si une activité ne peut être affectée d'un unique type (informatisée ou manuelle), ceci est le signe que l'activité peut être décomposée en autant d'activités que de types auxquels elle aurait pu être associée. 7. **Décomposer le processus** en fonction du critère choisi. - Critères suggérés: • *Changement d'objectif* : quand l'objectif du processus est assez général, il est nécessaire de l'affiner en le décomposant. Les sous-objectifs ainsi obtenus sont associés à diverses activités qui correspondent à la décomposition du processus initial. Nous soulignons que la présence d'un objectif dans une activité donnée signifie implicitement l'occurrence d'une action de *mesure* de satisfaction de l'objectif à la fin de l'activité et donc un point de décision[178]. La décomposition de processus selon les objectifs implique donc une décomposition selon les points de décision. • *Changement d'acteur* : un processus est assuré par des acteurs selon différents rôles. Une deuxième façon de décomposer un processus sera par changement d'acteurs. Par changement d'acteur, nous entendons aussi bien le changement d'un acteur par un autre que le changement du rôle d'un même acteur. – *Changement de type d'activité* : à un niveau avancé de la décomposition d'un processus, les activités doivent être typées selon qu'elles soient manuelles ou informatisées. Une troisième façon de décomposer un processus sera selon le type des activités. Si une activité du processus ne peut être caractérisée de façon complète (i.e. on ne peut lui affecter un unique type : informatisée ou manuelle), il convient alors de décomposer cette activité en autant d'activités que de types auxquels elle aurait pu être associée. - Règle de décomposition : en décomposant un processus, • les liens qui y rentraient dans le diagramme d'activités de niveau supérieur pointeront sur la première activité de ce processus, • les liens qui en sortaient dans le diagramme d'activités de niveau supérieur partiront de la dernière activité de ce processus, • les liens de « re-direction » qui re-bouclaient sur le processus dans le diagramme d'activités de niveau supérieur pointeront sur chacune des activités de ce processus ayant des points de

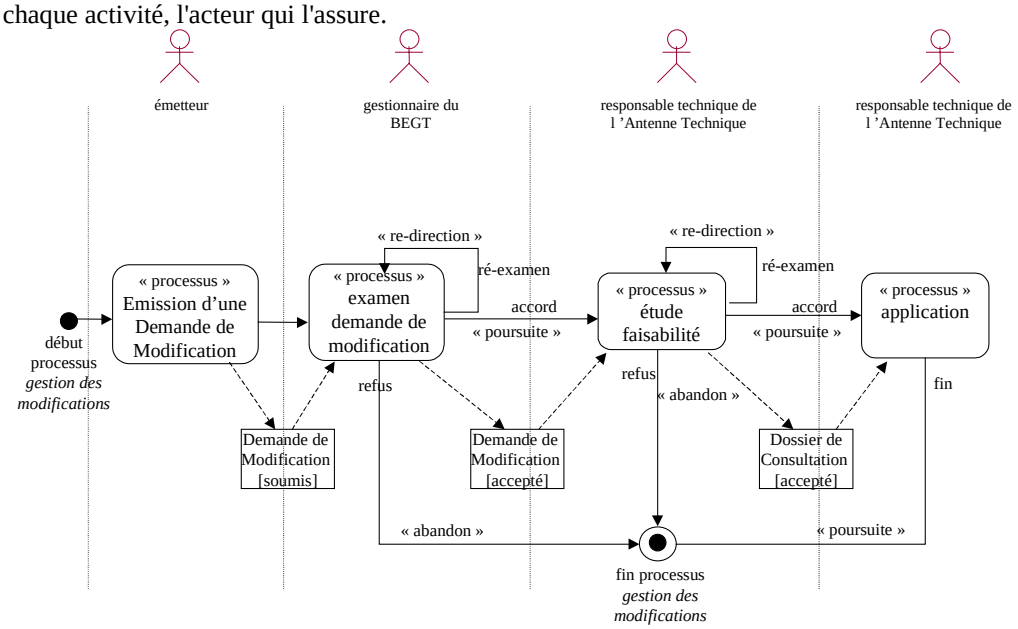
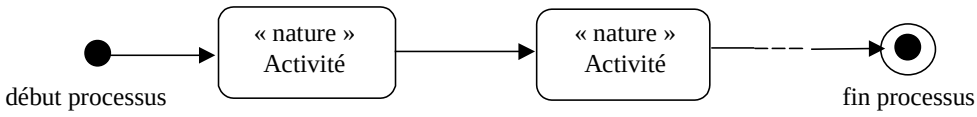
[177] Nous préconisons ce critère à un niveau avancé de la décomposition du processus.

[178] Un point de décision correspond au choix entre plusieurs transitions disjointes lors de la fin d'une activité donnée.

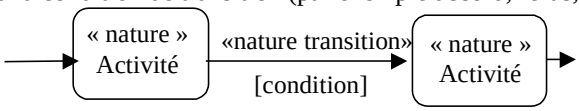
	décision et donnant lieu à des re-directions. 8. **Renseigner les activités** ainsi obtenues. Pour chacune d'elles : - nommer l'activité, - indiquer la nature de l'activité : opération ou processus, - lui affecter, si possible, un initiateur, un responsable et un exécutant (ce peut être un acteur ou un groupe d'acteurs ou une ressource matérielle), - lui affecter, si possible, un type : informatisée ou manuelle, - lui affecter, lorsqu'il s'agit d'une opération, les objets entrants et les objets sortants quand ils existent, - déterminer le type de succession : *ET* / *OU*, - typer les transitions entre activités : *poursuite*, *re-direction*, *abandon* (quand il s'agit de transitions de type OU), - exprimer les conditions de succession sur les transitions (accord, refus, réexamen, …). 9. **Etablir le diagramme d'activités** correspondant au découpage obtenu. Pour cela *appliquer le patron "Représenter un Processus"*. 10. **Repérer les activités non terminales** : une activité terminale correspond à une opération, donc ne pouvant être décomposée. Les activités non terminales possèdent au moins une des caractéristiques suivantes : - son objectif est trop général et peut être décomposé, - il n'a pas été possible de lui affecter *un* exécutant. Ceci est le signe que cette activité doit être décomposée en autant d'activités que d'exécutants qui auraient pu lui être associés. - il n'a pas été possible de lui affecter un unique type (informatisée ou manuelle). Ceci est le signe que cette activité doit être décomposée en autant d'activités que de types auxquels elle aurait pu être associée.
Solution-Modèle	Le modèle obtenu est un diagramme d'activités représentant la décomposition du processus en activités et illustrant les diverses ressources qui l'assurent ainsi que les entrants/sortants de chaque activité.

Patron "Représenter un Processus"

Nom	Représenter un Processus.
Classification	Patron d'analyse processus.
problème	Représenter un processus SIP à l'aide des diagrammes d'activités d'UML.
Motivation	Prenons l'exemple du processus de ***Gestion des Modifications de produits***. L'organisation d'un tel processus doit permettre une gestion optimale de la modification tout en intégrant un minimum de mécanismes de contrôle de cette modification de manière à ce que celle-ci, si elle est validée, soit appliquée sans aucun risque et après toutes les vérifications jugées nécessaires. Un tel processus de gestion de modification de produit peut être découpé en activités de manière à le rendre modulaire, à mieux cerner les actions nécessaires pour le mener et, ainsi, à mieux définir les acteurs responsables de chacune de ces actions. Ainsi, nous pouvons décomposer le processus de gestion des modifications en les activités suivantes : - *Emission Demande de Modification* :. - *Examen demande :* le processus est initialisé par une émission d'une demande de modification qui est soumise à un gestionnaire du BEGT (Bureau d'Etudes Gestion Technique) pour examiner la pertinence de la demande et décide de la continuité du processus ou pas. - *Etude faisabilité :* si la demande est acceptée lors de l'étape précédente, elle est soumise aux responsables des AT (Antenne Technique des usines de production) pour une étude de faisabilité portant sur différents critères (techniques, financiers, …). - *Application :* si la demande est jugée faisable, elle est mise en application par les

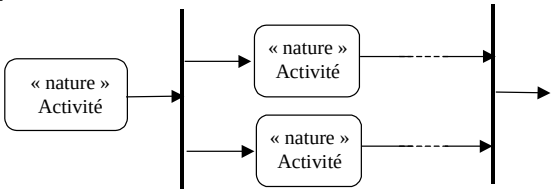
	membres de l'AT. Le diagramme d'activité UML suivant représente la décomposition du processus, en associant à chaque activité, l'acteur qui l'assure. 
Force	Ce patron permet de guider les concepteurs du SIP qui ne maîtrisent pas la représentation de processus avec des diagrammes d'activités d'UML, en offrants les concepts de bases (la manière) pour représenter les diverses entités dans le diagrammes d'activités.
Contexte	Ce patron n'exige autre patron pour être appliqué.
Solution-Démarche	Les entités de base dans la description de processus sont : **activité** (du type opération ou processus); **transition** entre activité, **ressource** et objet **entrant\sortant** des activités. 1. **Activité** : Toute activité qu'elle soit une opération ou un processus sera représenté par une activité UML (un rectangle arrondi), stéréotypé selon sa nature (opération ou ressource) et éventuellement son type (informatisée ou manuelle). La décomposition d'un processus donné est représenté ainsi par un diagramme d'activités. Nous supposons qu'un processus a un état initial (représenté par un gros point noir) qui marque le début du processus et peut avoir un ou plusieurs états finaux (représentés par de gros points noirs encerclés) qui correspondent chacun à une condition de fin du processus différente.  2. **Transition** entre activités : la succession entre activités est modélisée sur les diagrammes d'activités d'UML par des transitions UML, stéréotypées en fonction de leur nature (poursuite, re-direction, abandon). - En ce qui concerne le déclenchement des transitions entre les activités, dans les diagrammes d'activités d'UML, les transitions sont automatiques. Elles ne sont pas déclenchées par des événements externes mais c'est la fin de l'activité précédente qui déclenche la transition et fait démarrer l'activité suivante; en d'autres termes, l'événement est la fin de l'activité précédente. - Lorsqu'il s'agit de transitions gardées par des conditions de succession, c'est la

condition de garde qui valide la transition et la déclenche. Le nom d'un transition correspond à la condition de transition (par exemple accord, refus, …).



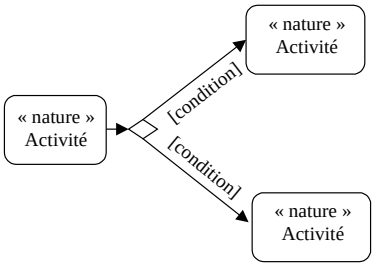
Dans les cas particulier de :

- succession de type ET avec plusieurs successeurs, la synchronisation entre flots de contrôle est représenté à l'aide de barres de synchronisation. Une barre permet d'ouvrir et de fermer des branches parallèle. Les transitions au départ d'une barre de synchronisation sont déclenchées simultanément. Inversement, une barre de synchronisation ne peut être franchie que lorsque toutes les transitions en entrée sur le barre ont été déclenchées.



Transition de type ET

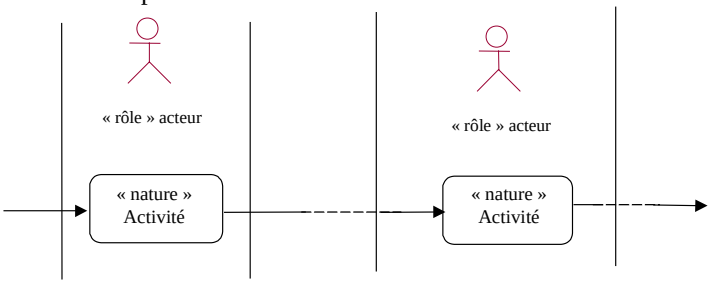
- Succession de type OU, UML définit un stéréotype pour la visualisation de transitions avec des conditions exclusives. Une condition est matérialisée par un losange d'où sortent plusieurs transitions exclusives.



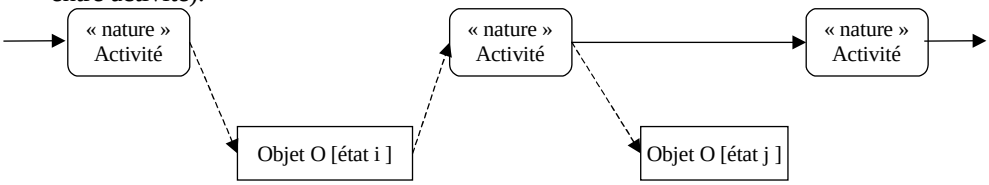
Transition de type OU

3. Ressource : le concept de ressource est représenté dans le diagramme d'activités d'UML par un acteur UML (un acteur UML peut être aussi bien un acteur humain qu'un acteur matériel), stéréotypé selon son rôle.

Le diagramme d'activité peut être découpé en couloirs d'activités (comme une piscine découpée en couloirs de natation) pour montrer les différentes ressources assurants les activités. A chaque ressource, est allouée à un couloir donné. La position relative des couloirs n'est pas significative; les transitions peuvent traverser librement les couloirs.

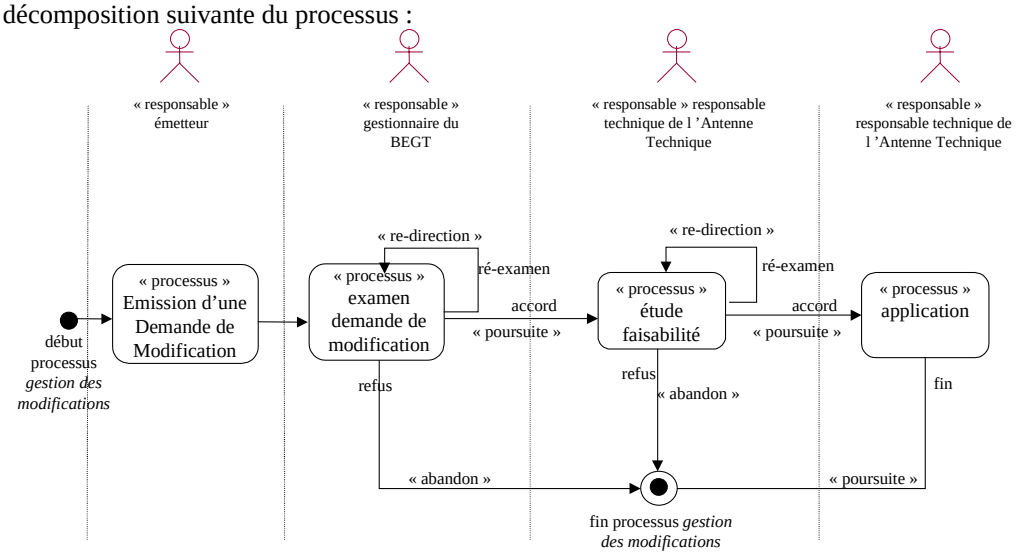


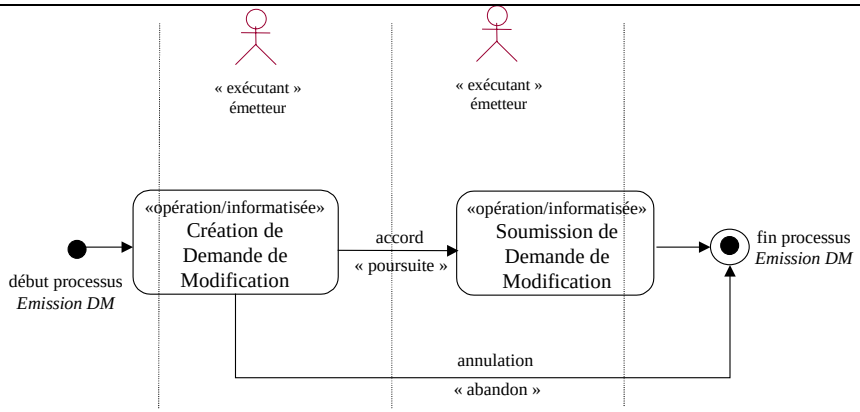
4. Objets entrants\sortants : Dans les diagrammes d'activités d'UML, les objets peuvent être représentés à l'aide d'objets UML (un carré), avec leurs états respectifs. Les états sont spécifiés dans une expression entre crochets à l'intérieur du carré. Les flots d'objets sont alors représentés par des flèches pointillées. Une flèche relie ainsi un objet (en fait un état d'objet) à l'activité qui l'a créé ou qui le met en jeu. Lorsqu'un objet produit par une activité

	est utilisé immédiatement par une autre activité, le flot d'objets représente également le flot de contrôle; il est alors inutile de représenter explicitement ce flot de contrôle (transition entre activité). 
Solution-Modèle	Le modèle obtenu est un diagramme d'activité UML .

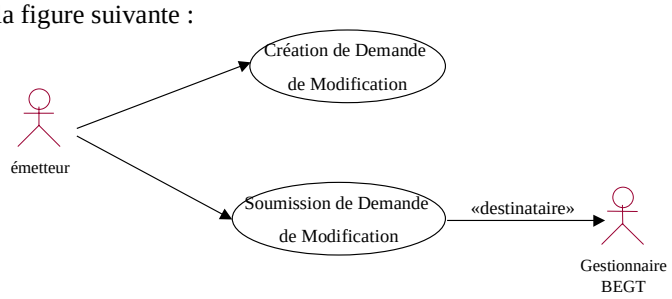
3. Patrons de Conception

Patron " Modèle de Conception SIP"

Nom	Modèle de Conception SIP.
Classification	Patron de conception
Problème	Identifier les objets du Système d'Information Informatisé (SII) associé au Système d'Information Organisationnel (SIO) à partir de la modélisation du produit et des processus métiers du SIP .
Motivation	Nous considérons le processus de gestion des modifications produit. A partir de la décomposition suivante du processus :  et en ne repérant que les activités informatisées de ce processus, telles que pour le premier sous-processus "Emission d'une Demande de Modification", les opérations suivantes :

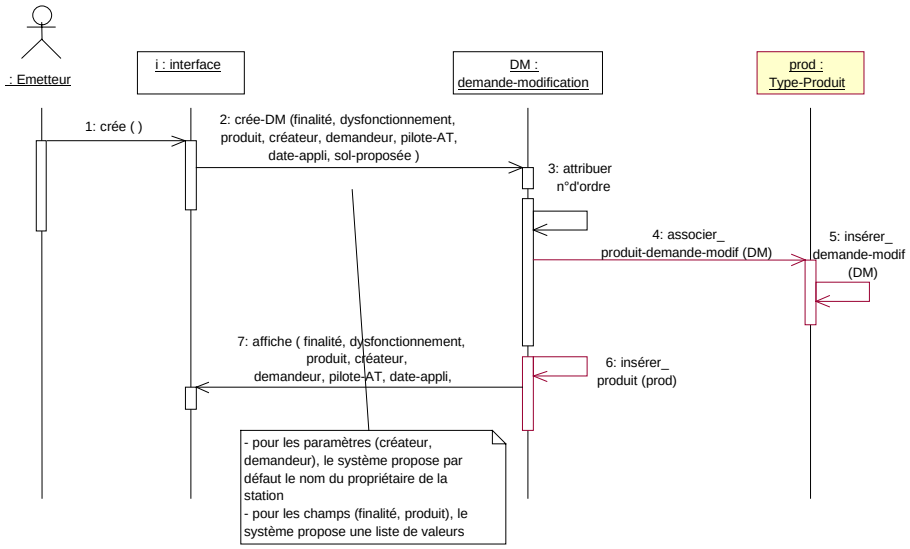


on obtient les cas d'utilisation du SII associé au processus (fonctionnalités attendues du SII), comme l'illustre la figure suivante :

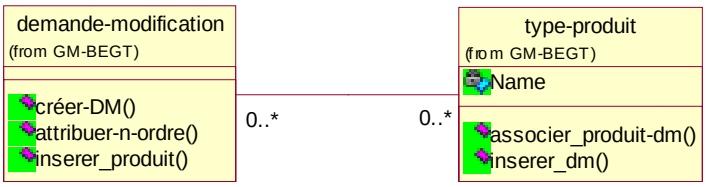


En modélisant dans des diagrammes de séquence, les interactions sous forme de messages entre l'acteur et le système (informatique) nécessaires pour réaliser les cas d'utilisations attendus et par une transformation successive des diagrammes de séquences obtenus, nous identifions les sociétés d'objets du SII collaborants pour réaliser ces cas d'utilisation. Il est alors possible de construire le diagramme de classes associé, modélisant l'ensemble des objets identifiés et qui enrichit le modèle d'analyse produit.

Pour le cas d'utilisation "Création de Demande de Modification", le diagramme de séquence associé se présente comme suit :



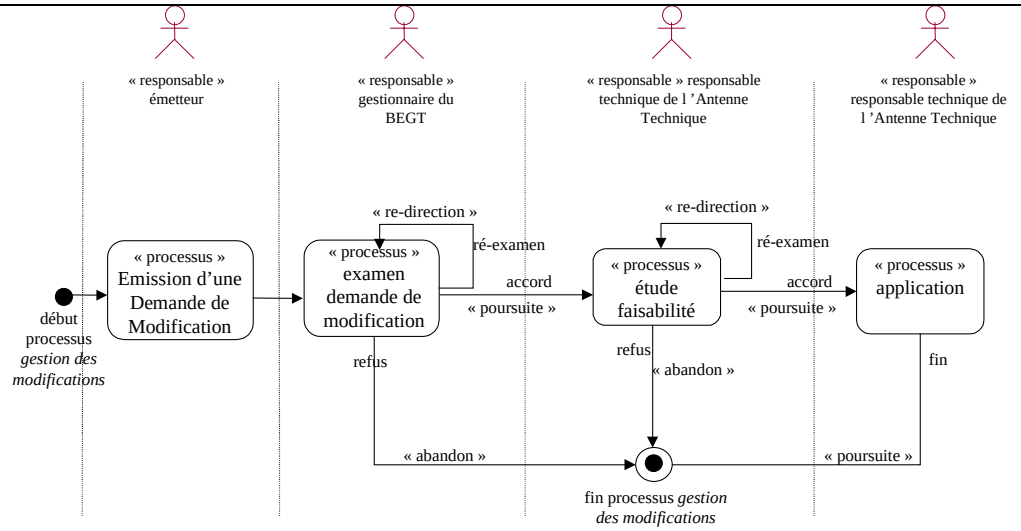
et le diagramme de classes qui en découle est le suivant :

	
Force	A partir des diagrammes d'activité modélisant la décomposition des processus métiers du SIO, on aboutit à un diagramme de classe enrichissant le modèle d'analyse produit en modélisant la dynamique des objets métiers associés au produit engendrée par l'exécution des divers processus métiers (à l'aide d'opérations de classes) et en ajoutant de nouveaux objets nécessaires pour l'exécution des processus informatiques du SII associé.
Contexte	Le concepteur SIP doit avoir appliquer une fois le patron d'analyse produit "Points de Variabilité" et une ou plusieurs fois le patron d'analyse processus "Décomposer un Processus" ou au moins disposer d'un modèle produit et d'un ou plusieurs modèles processus[179].
Solution-Démarche	3. Pour chaque processus métier : ▪ Etablir le modèle d'expression de besoins du SII associé au processus métier considéré. Ce modèle exprime les cas d'utilisation du SII et les documente à l'aide de diagrammes de séquence de haut niveau. Pour cela, *appliquer le patron "Expression des Besoins du SII"*. 1. Etablir le modèle de conception du SII associé au processus métier. Ce modèle met en jeu, par une succession de modèles, des sociétés d'objets collaborants pour réaliser les cas d'utilisation du SII décrites dans le modèle d'expression de besoins du SII. Pour cela *appliquer le patron "Objets de Conception du SII"*. 2. Le diagramme de classe ainsi obtenu illustre les opérations des classes associés aux divers concepts produit mis en jeu dans le processus SIP considéré. Le modèle de conception complet du SIP est obtenu en intégrant les différentes diagrammes de classes obtenues à l'issue de l'étape 1.
Solution-Modèle	Le modèle obtenu à l'issue de la démarche proposée est un diagramme de classe représentant le modèle de conception du SIP. Il modélise la structure et le comportement des objets du SII et complète ainsi le modèle d'analyse produit en explicitant la dynamique des objets métiers du produit engendrée par l'exécution des processus métiers étudiés et en ajoutant de nouveaux objets (de conception).
Utilise	{Expression des Besoins du SII, Objets de Conception du SII}

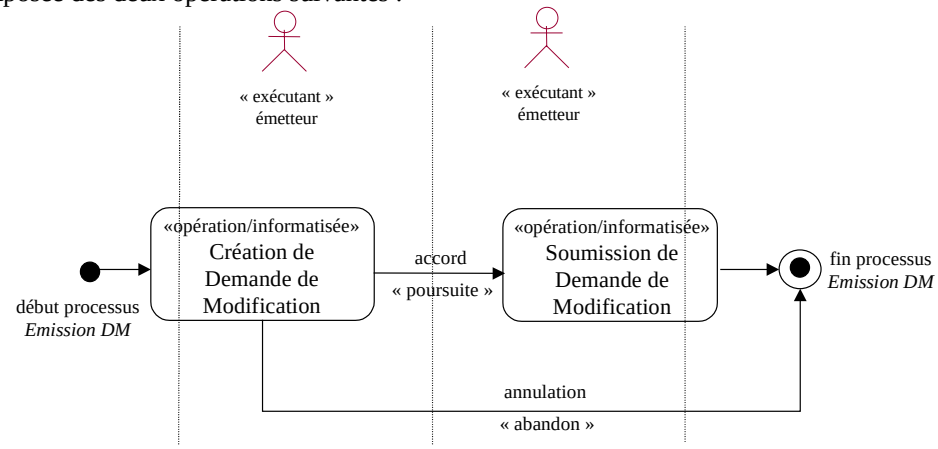
Patron " Expression des Besoins du SII "

Nom	Expression des Besoins du SII.
Classification	Patron de conception.
Problème	Définir et documenter à l'aide de diagrammes de séquence les cas d'utilisation du Système d'Information Informatisé (SII) associé à un processus métier du SII.
Motivation	Nous considérons le processus de gestion des modifications produit. Celui-ci est décomposé à un premier niveau de la manière suivante :

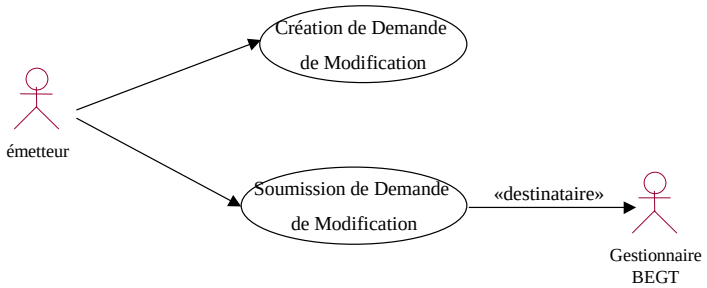
[179] On n'oblige pas le concepteur à appliquer les patrons du catalogue SIP. Il doit dans ce cas disposer d'un modèle produit et d'un ou de plusieurs modèles processus.



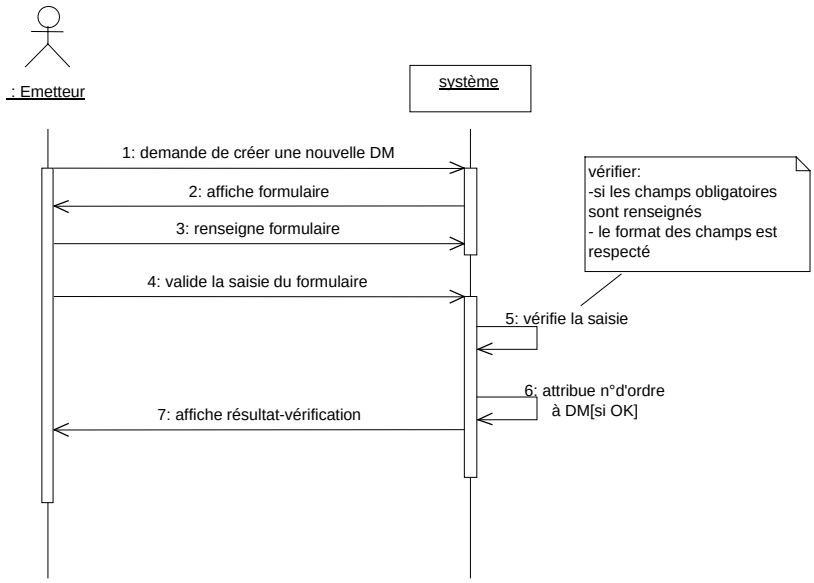
Si l'on se focalise sur la première activité (émission d'une demande de modification), celle-ci est composée des deux opérations suivantes :



Chacune des deux opérations ainsi obtenues est informatisée à l'aide d'un SGDT. Ces opérations correspondent aux cas d'utilisation du SIP informatisé (fonctionnalités attendues du SII). La figure suivante les illustre dans un diagramme de cas d'utilisation.



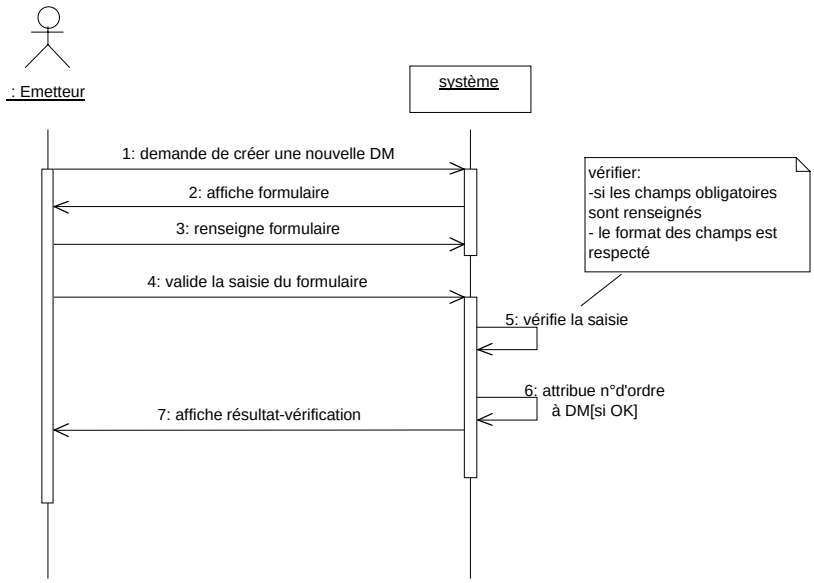
On se focalise sur le cas d'utilisation "Création de Demande de modification". Cette fonction a pour objectif d'assurer la création d'une demande de modification en renseignant un formulaire. Il s'agit uniquement d'une création de la demande sans la soumettre. A ce stade l'objet créé est détenu uniquement par l'émetteur. Si l'on modélise, dans un diagramme de séquence, les interactions sous forme de messages entre l'acteur et le système (informatique) afin de réaliser la fonction attendue, nous obtenons le modèle suivant :

	
Force	Permet de déterminer pour un processus du système d'information organisationnel (processus métier) le processus du système d'information informatisé qui lui est associé (processus informatique). Il permet d'exprimer les besoins attendues du SII associé au processus du SIO étudié dans un diagramme de cas d'utilisation du SII associé et ensuite de documenter chacun des cas d'utilisation à l'aide d'un diagramme de séquence de haut niveau.
Contexte	Le concepteur SIP doit avoir appliquer n fois le patron d'analyse processus "Décomposer un Processus" jusqu'à arriver aux opérations informatisées ou au moins disposer d'un modèle processus décomposé jusqu'aux opérations informatisées.
Solution-démarche	4. Dans la décomposition obtenue du processus métier considéré, repérer les opérations du type "informatisée". Ces opérations constituent les opérations des processus du système informatique associé au SIP, càd le SII (Système d'Information Informatisé). Elles correspondent aux cas d'utilisation de ce système. 5. Représenter ces opérations informatisées dans un diagramme de cas d'utilisation. Pour cela, procéder en suivant les règles suivantes : ▪ Chaque opération du diagramme d'activité est un cas d'utilisation dans le diagramme de cas d'utilisation. ▪ Les ressources intervenant dans la réalisation de l'opération ou de son processus (processus dans lequel l'opération est un composant direct) sont les acteurs du cas d'utilisation associé à l'opération s'ils sont des acteurs du système informatique (càd ses utilisateurs directs). Un cas d'utilisation est une fonction attendue d'un acteur principal mais peut impliquer des acteurs qu'on qualifie de secondaires (ils n'agissent pas dans le cas d'utilisation - ils sont passifs). Désigner parmi les acteurs du cas d'utilisation ainsi identifiés, l'acteur principal (celui qui agit sur le système pour satisfaire le cas d'utilisation - il est souvent celui qui a le rôle de "responsable" dans le diagramme d'activité). Les autres acteurs sont des acteurs secondaires. ▪ Construire le diagramme de cas d'utilisation en mettant en relation les cas d'utilisations entre eux et avec les acteurs. Trois types de relations peuvent exister : - Relation de communication : entre un cas d'utilisation et l'acteur qui lui est associé. Pour distinguer les différents acteurs, associer aux relations cas d'utilisation - acteur secondaire un stéréotype "destinataire".

	- Relation de type "extend" : entre deux cas d'utilisation A et B lorsque le cas A étend le comportement du cas B (il est un cas particulier de ce dernier). - Relation de type "use" : entre deux cas d'utilisation A et B lorsque le cas B comprend (dans ses actions) le comportement décrit par le cas A. D'autres types de relations spécifiques entre cas d'utilisation peuvent être définies, en stéréotypant les liens. Remarque : lorsque le processus métier étudié est de large échelle et fait donc intervenir plusieurs acteurs et cas d'utilisation, il est préférable d'organiser les divers cas d'utilisation associés (ainsi que les acteurs concernés) dans différents paquetages UML. Ceci facilite la lecture des différents modèles. Ce regroupement de cas d'utilisation peut se faire selon plusieurs critères. A titre d'exemple, selon le site industriel concerné par les cas d'utilisation. 6. Chaque cas d'utilisation dans le diagramme de cas d'utilisation obtenu apparaît comme un scénario qui peut être documenté sous forme graphique, au moyen de diagramme de séquence. Pour chaque cas d'utilisation, construire le diagramme de séquence de haut niveau associé. Pour cela, procéder comme suit : ▪ Observer à l'intérieur du cas d'utilisation la séquence, selon un point de vue temporel, des interactions entre les acteurs et le système. Ces interactions se décrivent en termes d'informations échangées pour réaliser la fonctionnalité attendue. ▪ Construire le diagramme de séquence correspondant avec, comme objets les acteurs et le système et comme messages, les interactions successifs entre ces objets. Un message est représenté au moyen d'une flèche horizontale orientée de l'émetteur du message vers le destinataire. Sur chaque flèche, indiquer le nom de l'interaction associée. - Remarque : lorsqu'il existe plusieurs variantes de scénario au sein du cas d'utilisation, il est préférable de construire autant de diagramme de séquence que de variantes de scénario.
Solution-Modèle	Le modèle obtenu est un ensemble de diagrammes de séquence de haut niveau, documentant les cas d'utilisation du SII associé au processus métier étudié.

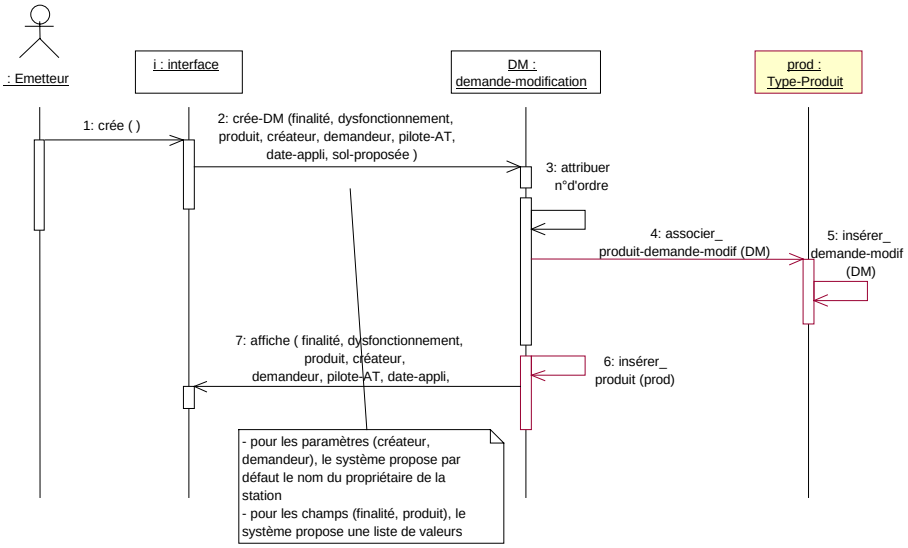
Patron " Objets de Conception du SII "

Nom	Objets de Conception du SII.
Classification	Patron de conception.
Problème	A partir des besoins attendues d'un système informatiques exprimés en terme de cas d'utilisation et documentés à l'aide de diagrammes de séquence de haut niveau, arriver à déterminer les objets mis en jeu dans le système informatique pour répondre à ces besoins, en terme de structure statique (classes + attributs + associations) mais également en terme de dynamique de ces objets (opérations sur les classes).
Motivation	Dans le cadre du processus de modification de produits, le cas d'utilisation "Création de Demande de modification" a pour objectif d'assurer la création d'une demande de modification en renseignant un formulaire. Il s'agit uniquement d'une création de la demande sans la soumettre. A ce stade l'objet créé est détenu par l'émetteur uniquement. Le diagramme de séquence de haut niveau correspondant est comme suit :



L'ensemble de ces interactions constituent des événements extérieurs au système. Ces événements externes se traduisent par des interactions entre les objets du système pour répondre à ces sollicitations, sous forme d'échanges de demandes de service entre les différents objets du système.

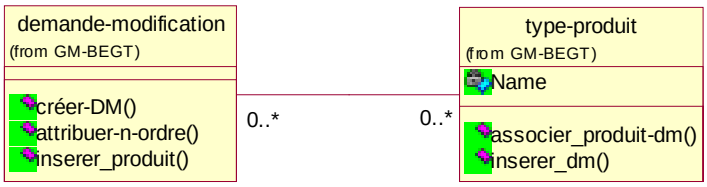
Le diagramme suivant illustre ces interactions entre les objets du système.



Les demandes de services entre objets du système se matérialisent ainsi par des opérations sur les classes correspondants aux différents objets.

Ainsi, par transformation successive des diagrammes de spécification, les objets mis en jeu dans le processus sont déterminés ainsi que les opérations sur les classes correspondants à ces objets.

Le diagramme de classes issu de cette analyse est le suivant :

	 Ce diagramme sert à enrichir le modèle d'analyse produit obtenu par application des patrons d'analyse produit.
Force	Ce patron propose une démarche pour déterminer un diagramme de classes à partir d'un diagramme de séquence de haut niveau associé à un cas d'utilisation du SII, grâce à une transformation continue des modèles UML, garantissant ainsi une cohérence entre les divers modèles. Ce patron permet d'enrichir le modèle d'analyse du SIO en modélisant la dynamique des objets métiers de ce modèle, engendrée par l'exécution des cas d'utilisation associés aux processus métiers du SIO et en ajoutant de nouveaux objets nécessaires pour l'exécution du cas d'utilisation.
Contexte	Le concepteur doit avoir appliquer le patron de conception "Expression des Besoins du SII" ou disposer d'un diagramme de séquence de haut niveau documentant les cas d'utilisation du SII considéré. Le concepteur doit par ailleurs appliquer le patron d'analyse produit "Points de Variabilité" ou disposer d'un modèle d'analyse recensant les objets métiers du SIO.
Solution-Démarche	3. Dans le diagramme de séquence de haut niveau considéré, l'ensemble des interactions entre acteur et système constituent des événements extérieurs au système. Ces événements externes se traduisent par des interactions entre les objets du système (informatique) pour répondre à ces sollicitations, sous formes d'échanges de demandes de service entre les différents objets du système. Il s'agit donc ici de mettre en évidence comment des sociétés d'objets collaborants viennent réaliser les interactions décrites dans les diagrammes de séquences de haut niveau. Ces objets sont initialement ceux qui découlent de la phase d'analyse de besoins (objets du domaine) et ils sont complétés par des objets de conception. Ceci s'apparente à un effet de zoom à l'intérieur de l'objet unique qui représente le système dans le diagrammes de séquence de haut niveau. Construire alors le diagramme de séquence de bas niveau associé au diagramme de séquence de haut niveau considéré : ▪ A chacune des interactions du diagramme de séquence de haut niveau, associer une collaboration d'objets. Une collaboration d'objets réalise l'interaction en échangeant des demandes de service entre objets. ▪ A chacun des objets identifiés, associer un type général, càd une classe. ▪ Les demandes de services entre objets se traduisent par des opérations sur les classes associées à ces objets. En effet, une demande de service entre deux objets est une communication qui déclenche une activité dans l'objet destinataire de la demande, càd une opération sur la classe associée à l'objet en question. Traduire alors les différentes demandes de services en opérations de classes. ▪ Associer aux opérations de classes identifiées leurs paramètres, qui ne sont autre que les données communiqués lors des demandes de service. Ces paramètres constituent les attributs des classes mises en jeu. 4. A la construction du diagramme de séquence de bas niveau, les classes sont identifiées et ils sont décrits par les opérations et les attributs. On peut alors déduire le diagramme de classe partiel issu de cette analyse. Les liens entre les classes se déterminent selon le type d'opérations entre elles. Il est ensuite possible d'affiner ce diagramme de classe en

	introduisant des super-classes pour les classes qui s'avèrent généralisables. Remarque : on peut également déduire le diagramme d'état associé à chaque classe. En examinant la séquence des opérations sur la classe, dans les différents diagrammes de séquences où elle y figure, il est possible de déduire les différents états de la classe. Le diagramme d'état est la représentation de ces différent états avec les événements déclencheurs des diverses transitions et les actions qui s'exécutent dans la classe (qu'on détermine parmi les opérations de la classe).
Solution-Modèle	Le modèle obtenu à l'issue de la démarche proposée est un diagramme de classe représentant une partie du modèle de conception du SIP. Il modélise la structure et le comportement des objets du SII et complète ainsi le modèle d'analyse du SIO en explicitant la dynamique des objets métiers de ce modèle, engendrée par l'exécution des cas d'utilisation associés aux processus métiers du SIO et en ajoutant de nouveaux objets nécessaires pour l'exécution du cas d'utilisation.