الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي

BADJI MOKHTAR-ANNABA UNIVERSITY
UNIVERSITE BADJI MOKHTAR-ANNABA

جامعة باجي مختار- عنابة

Année : 2016 / 2017

Faculté des Sciences de l'Ingéniorat
Département d'Informatique

THESE

Présentée en vue de l'obtention du diplôme de
DOCTORAT EN SCIENCES

MODELISATION MULTI-PERSPECTIVES DES SYSTEMES EN LIGNES DE PRODUITS

FILIERE
Informatique
OPTION
Informatique Industrielle

Par
DJEBAR YACINE

DEVANT LE JURY

Président	:	LASKRI Mohamed Tayeb	Prof. Univ.Badji Mokhtar- Annaba
Rapporteur	:	GUERSI Nouredine	Prof. Univ.Badji Mokhtar- Annaba
Co-Rapporteur	:	KIMOUR Mohamed Tahar	Prof. Univ.Badji Mokhtar- Annaba
Examinateurs	:	CHAOUI Allaoua	Prof. Univ.Constantine II.
		LAFIFI Yacine	Prof. Univ.8 Mai 1945- Guelma
		MBAYE Sene	Prof. UCAD.Dakar, Sénégal

الملـــخـــص

الإشكاليات الجديدة التي فرضها اختراق تكنولوجيا المعلومات لعالم الصناعة و على الخصوص المتعلقة منها بالأنظمة المحمولة ترتبط بالنمو المذهل و المعقد للبرمجيات. من بين الحلول الأحسن تكيفا لعملية التصنيع هذه و المتعلقة بالبرمجيات برزت تقنية التصميم و التطوير ضمن إطار خطوط المنتجات البرمجية.

هذه التقنية المعتمدة على استعمالية البرمجيات المطورة سالفا تأخذ بعين الاعتبار القواسم المشتركة – المشتركية – و القواسم المتغيرة – التباينية – المتعلقة بالبرمجيات التي تتيح تخفيض تكاليف التطوير وكذا مدة إنجازها و الحقبة اللازمة لتسويقها.

لغرض الاستجابة لهذه الاهتمامات و انتاج برمجيات مكينة و موثوق بها ، يتحتم على خطوط المنتجات الاعتماد في نفس الوقت على نماذج قوية و متمددة تتوفر على تعددية الانماط المرتبطة بهذه التقنية .

غير انه في الممارسة الميدانية لا توجد مناهج او انماط عملياتية يمكن اتباعها لتطوير هذا النوع من النماذج.

تندرج هذه الرسالة ضمن هذا الاهتمام المزدوج الذي يبحث على اقتراح مقاربات و مكانزمات جديدة تتكفل بفعالية بالجوانب المتعددة الابعاد المرتبطة بتطوير و تمديد النماذج الخاصة بخطط المنتجات البرمجية.

لغرض الاحاطة بكل الجوانب المرتبطة بهذه المقاربات اضطررنا الى تنظيم عملنا هذا وفقا لمساهمتين على شكل نماذج مرتبطة كلها بأنماط التطوير و تمديد التباينية .

- ***النمذجة الاولى تعتمد على مسار يستند إلى مجموعة من النماذج (تباينية، تحليلية، ارتباطية، تقييمية وتحويلية) و مقاييس متعددة. هذه الاخيرة تمكن من استكشاف إمكانيات إنشاء البنية الامثل لخط المنتجات من خلال انشاء مخطط خصائص ابتداءا من عدد قليل من المنتجات المماثلة فقط ، كما يقدم لمختلف المستعلمين نظرة متعددة لنماذج الخط طبقا لمعايير تحليلية عدة.***
- ***النمذجة الثانية، من خلال تكييف مفهوم << الدور >> المرتبط بلغة*** **UML** ***المصمم في الاصل لنظام واحد للاستعمال ضمن خطط المنتجات المتعددة الانظمة تم اتاحة للمصمم امكانية تمثيل و تسيير تطور التباينية الخاصة بالسمات المتغيرة باستخدام الادوار المتمددة . لأجل تحقيق هذه المهايأة ، كان من الضروري الحاق رابطين جدد – الدور و التمدد – ، و كذا طرازين جديدين للتمدد. تم كذلك اضافة شكل جديد للتمثيل : " نموذج التمدد " .*** **هدا الاخير يقسم بيان الخصائص الى مجموعة من الاجزاء المتغيرة و المتمددة . كل منها يربط عن طريق خانتين الاقسام المتغيرة المتعلقة بدور الخاصية (f-role) حسب صورية بيان الاقسام ، بالعناصر المماثلة حسب صورية الة الحالات. كل المفاهيم و المكانزمات المقترحة تم ادماجها ضمن النماذج الإجمالية لبرنامج UML.**

الكلمات المفتاحية : *خط انتاج البرمجيات ،تطوير موجه حسب خط انتاج البرمجيات، بيان الخصائص ، تحليل متعدد المعايير ، مقاييس التمدد ، تباينية متمددة، نموذج التمدد*

Résumé

Les nouvelles problématiques introduites par la pénétration de l'informatique dans l'industrie notamment celle relative aux systèmes embarqués sont axées essentiellement sur la croissance et la complexité des logiciels. L'une des solutions les plus adaptées à cette industrialisation du logiciel est la conception et le développement dans le cadre de lignes de produits logiciels.
Cette technique basée sur la réutilisation du potentiel logiciel existant prend en compte les facteurs communs –commonalité- et variants –variabilité- des applications et permet de minimiser les coûts et les temps de réalisation ainsi que les durées de mise sur le marché des produits logiciels développés.
Pour répondre à de telles préoccupations et générer des produits fiables et robustes, une ligne de produits se doit d'être dotée simultanément de modèles performants et évolutifs capables de couvrir l'aspect multi-perspectives de cette technique.
Cependant, dans la littérature, il n'existe pas encore de méthodes opérationnelles à suivre dédiées pour le développement de tels modèles.
C'est dans cette double-préoccupation que s'inscrit cette thèse en ciblant cette problématique par la proposition de nouvelles démarches et de nouveaux mécanismes pour une prise en charge efficace des aspects développement et évolution des modèles d'une ligne de produits logiciels.
Pour bien cerner tous les volets liés à cet objectif, une organisation de la thèse autour de deux contributions sous forme de modélisations des aspects associés au développement et à l'évolutivité de la variabilité, a été nécessaire.

- *La première modélisation utilise un processus basé sur un ensemble de modèles (variabilité , analyse, corrélation , évaluation et mapping) et de métriques .Elle permet d'explorer les possibilités de créer une structure optimisée d'une ligne de produits à travers son diagramme de caractéristiques et à partir seulement de quelques produits similaires .Elle offre également aux différentes parties prenantes des vues variées des modèles de la ligne dans différentes configurations et selon plusieurs critères d'analyse .*

- *La deuxième modélisation cherche à travers une adaptation, du rôle UML -initialement conçu pour les systèmes uniques- aux lignes de produits multi-systèmes, à représenter et à gérer l'évolution des variants par l'usage de rôles évolutifs. Pour cela, l'introduction de deux nouvelles associations (role et evolution), de deux nouveaux scénarios d'évolution et d'une nouvelle représentation, le modèle d'évolution a été nécessaire. Ce dernier scinde le diagramme des caractéristiques en un ensemble de partitions variables évolutives. Chacune d'entre elles associe par l'usage de deux compartiments, les classes variables de f-roles selon les formalismes du diagramme de classes à leurs éléments équivalents selon les formalismes d'une machine à états .Tous les concepts et mécanismes proposés ont été intégrés aux Méta-modèles UML.*

***Mots Clés**: Ligne de produits Logiciels, Développement orienté lignes de produits logiciels, Diagramme des caractéristiques, Analyse Multi-critère, Métriques d'évolution, Variabilité Evolutive, Modèle d'évolution.*

Abstract

New issues introduced by the diffusion of computer science technologies in the industry particularly that relating to the embedded systems are primarily focused on the growth and complexity of software. One of the most-adapted solutions to this software industrialization is the development in the context of software product lines.
This technique based on the reuse of existing software potential takes into account the common factors-commonality- and variants –variability- of the software and minimizes the development completion, charges and time to market.
To address such concerns and generate consistent and robust products, a product line has to be provided simultaneously with effective and scalable models that cover multiple-perspectives aspect of this technique.
However, in the literature, there is no still of operational methods devote to develop such models. It's in this context that this thesis targets such a problematic and propose new approaches for an effective support and a management of a development and en evolution models of a software product line. In order to understand all the related parts to this approach, an organization of the thesis in two contributions in the form of modeling of all associated dimensions to the development and the extensibility of variability were necessary.

- *The first modeling uses a process based on a set of templates (variability, analysis, correlation, assessment and mapping) and metrics .It allows to explore the possibilities of creating an optimized structure of a product line through its feature diagram from only a few similar products .It also provides stakeholders various views of the product line models in different configurations according to several analysis criteria*

.

- *By adapting the UML initially single-system role to product lines, the third modeling allows represent and manage the evolution of variants by the use of evolving roles, two new associations (role and evolution) and two new scenarios of evolution. A new representation, the evolving model, divides a feature model into a set of evolving variable partitions .Each of them associates by two compartments, variable classes of evolving roles (f-roles) in class diagram formalisms to their equivalent elements in state machine formalisms. All of these proposed concepts have been integrated to the UML Meta-models.*

***Keywords**: Software Product Line, Software product-line oriented development, Feature Diagram, Multi-criteria Analysis, Evolution Metrics, Evolving variability, Evolution Model.*

Dédicaces

À la mémoire de ma défunte Mère

À titre posthume,
Mon père et Mes sœurs Nabila et Masria

À Mes frères et Sœurs,
Et en particulier, à mon frère Réda et à son épouse Houria,
À Ma Chère épouse Amel,
Et à Mes Adorables Enfants,

Yacine…

Remerciements

Mes premiers remerciements vont naturellement aux professeurs Noureddine Guersi et Mohamed Tahar Kimour, respectivement mon Directeur de thèse et mon Co-directeur de thèse qui, par leur direction, m'ont permis de réaliser ce travail et en même temps de découvrir l'antichambre du milieu de la recherche. Je tiens également à les remercier pour tout ce qu'ils ont bien voulu (et veulent bien encore) me procurer comme soutien, aussi bien sur le plan scientifique que sur le plan moral. Qu'ils sachent que leur gentillesse, leur entière disponibilité, leur précieux conseils et encouragements m'ont profondément marqué et m'ont permis de continuer à mener l'ensemble de ce travail à bien, y compris durant les inévitables périodes de doute. Leur grande rigueur scientifique et encadrement de qualité exceptionnelle ainsi que leur perpétuel souci quant à mon avenir professionnel ont grandement contribué à l'accomplissement de cette page de ma vie. Pour tout cela, et pour tout le reste, je tiens à les remercier le plus chaleureusement possible.

Ma plus grande gratitude va également à Mon frère: le Professeur Réda et à son épouse le Professeur Houria Djebar qui n'ont cessé de m'encourager et donner de précieux conseils pour poursuivre ce travail et aller le plus loin possible dans mes études en dépit de mes nombreux engagements professionnels et familiaux.

Je tiens à remercier également et chaleureusement Laskri Mohamed Tayeb , Professeur à l'Université Badji Mokhtar d'Annaba qui m'a fait le grand plaisir de présider mon jury de soutenance ainsi que Chaoui Allaoua, Professeur à l'université de Constantine et Lafifi Yacine, Professeur à l'université 8 Mai 1945 de Guelma sans oublier Mbaye Sene, Professeur à l'université de Dakar qui m'ont fait le grand bonheur de bien vouloir examiner cette thèse et apporter leurs jugements sur ses contributions . Je veux également les remercier pour leur disponibilité et leur immense gentillesse.

Mes remerciements vont également au Maitre-Conférencier en Mathématiques Mounsar Mohamed pour son assistance dans la validation des définitions formelles des différents concepts utilisés dans la thèse.

Je tiens aussi à remercier du plus profond de mon cœur et de mon âme tous les membres de ma famille, **et notamment ma défunte Mère. Qu'elle sache que sa présence à mes côtés tout au long de sa vie a été des plus précieuses.**

Je remercie également mon Épouse et mes Enfants qui ont partagé et supporté avec moi les charges et les engagements de ces dernières années. Qu'ils sachent aussi que leur présence à mes côtés tout au long de ces années a été des plus réconfortantes.

N'ayant pu dresser une liste exhaustive de toutes les personnes ayant contribué de près ou de loin à ce travail, Je remercie toutes celles et ceux qui m'ont aidé à le réaliser.

Table des matières

Chapitre 2. Modélisations en Lignes de Produits Logiciels

II Deuxième partie. Contributions

III ***Troisième partie.*** *Conclusion Générale et Perspectives*

Table des figures

Liste des tableaux

Introduction Générale

Introduction Générale

De nos jours, les systèmes informatiques enfouis qui intègrent ensemble du matériel électronique et du logiciel, ne cessent de prendre de la place dans le quotidien des citoyens et des organisations. Une dépendance cruciale vis à vis du maniement de ces systèmes s'en est même développée [1],[2].

Eu égard à cet essor, la communauté du système embarqué s'est vue concédée un rang prépondérant dans tous les secteurs d'activité et en particulier celui de l'industrie [3],[4],[5].

Cette situation est devenue de plus en plus imposante du fait que l'ensemble des taches pouvant être assurées sous des contraintes très dures par ce type de systèmes, ne peuvent l'être que péniblement ou pas du tout par l'être humain.

Réussir une bonne conception de ces environnements, nécessite outre des moyens financiers, un effort dans le domaine de la recherche lié notamment aux aspects de modélisation, de développement et de maintenance [5],[6],[7].

Répondant à cette préoccupation, plusieurs approches de modélisation et/ou de développement de systèmes intégrant le logiciel et le matériel ont été proposées dans la littérature à travers nombre de domaines informatiques tels l'informatique embarquée et le génie logiciel [8],[9],[10],[11]. L'une des plus imposantes en pratique est sans doute la ligne de produits (LdP) [12],[13],[14]. D'ailleurs, une étude menée par [15], a confirmé que durant la dernière décennie, plus de la moitié du logiciel enfoui a été développé en ligne de produits.

Cette dernière basée essentiellement sur le principe de la réutilisation à grande échelle [16],[17],[18] exploite les variations –variabilité- contenues dans les familles de systèmes logiciels similaires tout en prenant avantage de leurs aspects communs-commonalité- [9],[10].

A partir de ces deux concepts, une nouvelle ingénierie dédiée aux lignes de produits a vu le jour.

Cette ingénierie scinde la manipulation de la LdP en deux niveaux d'abstraction, le premier étudie le domaine en vue de préparer un ensemble de composants basiques appelés Assets (parties de codes, exigences similaires, documents spécifiques,…) qui seront utilisés (ou réutilisés) dans le second niveau pour construire les produits de la ligne appelées applications [1],[18],[19],[20]. En procédant ainsi, cette technique a permis de minimiser les coûts de réalisation, alléger l'effort de développement, diminuer les temps de réalisation et réduire les durées de mise sur le marché [21],[22][23].

Ce sont principalement ces apports qui confèrent à cette discipline la puissance et la fiabilité , essentiels dans le domaine du développement .

Toutefois, parvenir à de tels apports passe forcément par une modélisation efficace de la ligne de produits c.à.d. à travers la conception de modèles performants [24],[25],[26]. Ces derniers doivent être capables de produire un maximum d'applications à partir de l'exploitation optimale des avantages de ce mode de développement en matière de modélisation, à savoir :

La réutilisation accrue [27],[28],[29], La généricité maximale par le paramétrage à grande échelle [30],[31], Une capacité d'évolution [32],[33] et une fiabilité des modèles notamment le modèle clé des caractéristiques [34],[35],[36].

En plus de ces performances, les modèles de la ligne doivent convoiter l'acquiescement d'un maximum de représentations en multi-perspectives c.à.d. selon un maximum de vues en faveur des parties prenantes [37],[38].Ces dernières représentent l'ensemble de toutes les catégories d'utilisateurs de la ligne de produits (concepteurs, développeurs, architectes système, spécialistes en marketing,…).

Cependant, dans la littérature, la majorité des approches dédiées aux LdP proposent des améliorations dans les étapes et/ou les concepts de l'ingénierie de la ligne [39],[40],[41] .Peu d'entre elles se sont fondées sur la génération et l'exploitation de métriques et même dans cette catégorie, la focalisation ne s'est pas fixée sur la construction de véritables modèles de développement de lignes de produits.

Le constat est que ces dernières convoient en particulier deux directions :

La première cherche à développer des métriques pour sélectionner et évaluer des parties appropriées du logiciel hérité [69],[72],[75] dans le but d'estimer l'opportunité d'un développement en LdP ;

La seconde, vise à travers les métriques proposées à évaluer la performance des modèles existants et notamment le modèle des caractéristiques selon divers aspects tels le comportement de la ligne, l'effort de développement, la qualité des applications, la teneur de la ligne en matières de généricité ou de réutilisabilité [68],[69],[70],[71].

Outre l'aspect développement de modèles performants, une autre dimension des lignes de produits lui promulgue un aspect tout aussi probant, C'est la dimension évolutivité de ses modèles.

En pratique, mis à part le fait qu'une application soit générée à partir d'un modèle performant d'une ligne de produit, elle doit dans certaines situations répondre à un besoin émanant d'utilisateurs particuliers après sa dérivation. Il est parfois nécessaire pour le concepteur de prendre en compte ce besoin qu'il n'est pas possible de définir dans l'ensemble des

caractéristiques identifiées à partir desquelles les applications ont déjà été dérivées. Il en découle que les concepteurs doivent spécifier, concevoir puis maintenir un grand nombre de composants logiciels qui constituent les produits de la ligne ainsi que leurs inter-dépendances et contraintes. Ceci implique soit, la génération d'une grande diversité de produits, soit la maintenance d'un très grand nombre de caractéristiques, obtenues à travers une identification exhaustive des exigences du domaine de la ligne, avant leur transformation en caractéristiques [45],[61].Cependant, l'expérience a montré que ces deux activités demeurent complexes et difficilement réussies.

Pour répondre à ces questions liées à la prise en compte des nouvelles exigences, bon nombre de travaux se sont orientés vers la proposition de modèles d'évolutivité ou d'extensibilité des caractéristiques de la ligne.

L'extensibilité d'une caractéristique exprime sa capacité à couvrir toutes les exigences supplémentaires non considérées par le concepteur et propres à des utilisateurs particuliers. Il peut s'agir d'un nombre restreint d'utilisateurs présents ou d'un nombre considérable d'utilisateurs potentiels [44],[61],[68].

Partant du principe fondamental de la LdP relatif à la variabilité et la commonalité, prévoir un modèle d'évolutivité pour la ligne revient à concevoir un modèle d'évolution de la variabilité qu'elle renferme. Représenter un tel modèle s'est canalisé dans la majorité des approches proposées à trouver des mécanismes d'évolution des points de variation de la ligne de produits [49],[61].Ce dernier identifie un emplacement dans un modèle d'éléments composant la ligne de produits (tel que le modèle de caractéristiques) au niveau duquel la variation est exprimée.

Les insuffisances souvent rencontrées dans ces approches concernent en plus de la complexité des modèles d'évolution proposés, la difficulté de représenter les contraintes de transition des caractéristiques évolutives ainsi que leurs dépendances.

Il existe souvent une confusion entre le point de variation originel et ses formes d'extension. Les efforts d'amélioration dans cette direction ont engendré un autre problème, celui de la visibilité dans les modèles proposés. Cette insuffisance est perceptible à travers l'obligation de manipuler plusieurs modèles en même temps (modèles initiaux de représentation et modèles d'évolution) en plus de la complexité des structures des modèles d'évolution proposés (représentation chargée) et la difficulté rencontrée dans leur lecture.

La seule approche significative [50] qui s'est basée sur le principe de séparation entre le point variant et ses extensions a conduit à la gestion parallèle de deux modèles de la ligne, le modèle représentant les éléments de la ligne et le modèle d'évolution. Ce dernier est basé sur

la manipulation de 3 espaces, un espace des problèmes, un autre pour les solutions et un autre qui exprime la relation entre les deux (l'inter-espace) . Cette complexité dans la gestion de ce modèle d'évolution ajoutée à l'absence de représentation des contraintes de transition et des dépendances entre les caractéristiques a compliqué sa manipulation.

C'est pour réagir à cette double-préoccupation relative aux dimensions développement et évolutivité que s'inscrit notre thèse. A travers notre étude, nous nous sommes fixés comme objectif principal de proposer des solutions aux limitations et insuffisances liées aux aspects suscités plus haut de la ligne de produits. Nos propositions ont visé les objectifs suivants :

- Pour la dimension développement. Concevoir un modèle de caractéristiques performant à partir d'un processus de développement d'une ligne de produits basé sur un ensemble de métriques multicritères et une gamme de modèles multi-perspectives pour la variabilité, l'analyse, la corrélation , le Mapping , et l'évaluation des différentes vues générées pendant le déroulement du processus.
- Pour la dimension évolutivité . Représenter et gérer l'aspect evolutivité de la variabilité à travers une modélisation à base d'une adaptation du role UML aux contexte des multi-systémes et une perception plus appropriée de la ligne de produits .

Pour notre première contribution [73] , les métriques proposées peuvent être produites progressivement tout au long du déroulement du processus de construction du modèle optimal des caractéristiques à travers l'enchainement des modèles successifs suivants: pour la variabilité (au niveau duquel nous construisons le diagramme initial des caractéristiques) , pour l'analyse (qui utilise 4 critères scindés en différentes granularités pour lesquels le processus génère un certain nombre de métriques et de vues d'analyse), pour la corrélation (basé également sur les 4 critères d'analyse , développe un ensemble de métriques et de vues après application du Mapping du modèle d'analyse vers le modèle de corrélation), pour l'évaluation (basé lui aussi sur les 4 critères , génère une série de métriques d'évaluation et un ensemble de vues en forme de rapports, puis optimise les vues obtenues par l'usage des métriques et des rapports d'évaluation dans le but d'identifier la vue optimale pour chaque critère) . Un deuxième Mapping intervient pour convertir les 4 vues d'analyse optimales en 4 diagrammes de caractéristiques. L'intégration de ces derniers selon une technique appropriée [116], donnera le diagramme des caractéristiques optimal et global de la ligne de produits. Dans le but de montrer les points forts de notre démarche, nous avons focalisé nos illustrations sur le domaine des maisons intelligentes utilisé par [115] et plus précisément la partie de son diagramme des caractéristiques qui permet de contrôler l'accès à la maison.

Le deuxième mécanisme traite la dimension évolutivité du modèle des caractéristiques de la ligne en prenant en charge les deux directions actuelles [46],[47],[48] à savoir : une extensibilité à partir de modèles avec une meilleure représentation de la variabilité évolutive .
Pour cela, nous avons proposé un modèle évolutif de la variabilité basé principalement sur une adaptation du rôle UML.
Le constat actuel est que l'évolution de la variabilité dans la plupart des approches existantes a été réalisée à travers celle des éléments variants de la ligne selon différents modèles [61]. Parmi ces derniers, beaucoup utilisent comme base de représentation le diagramme de classes UML. Cependant, les tentatives de représentations des conditions d'évolution dans ce diagramme ont provoqué une surcharge de ce dernier. De ce fait, ces techniques se sont heurtées à un problème de visibilité [42],[43],[44],[45].
Cette contrainte a rétracté l'activité d'évolution à la charge du concepteur et de l'architecte système qui ont souvent recours à l'usage de mécanismes appropriés pour modéliser et gérer la variabilité.
Pour apporter une solution à cette problématique, nous avons proposé l'expression de la variabilité évolutive à travers une adaptation du rôle UML (destiné traditionnellement aux systèmes uniques) aux lignes de produits (dédiées aux multi-systèmes).
Au lieu du rôle UML tel que défini, nous proposons l'intronisation à UML de deux nouvelles associations proches de la généralisation désignées « role » et « evolution » à travers lesquelles nous représentons non seulement la variabilité mais aussi son évolution.
Pour la gestion de l'aspect evolution de la variabilité , notre modèle représente la variabilité dans une ligne de produits logiciels non pas par des variants mais par des rôles UML adaptés aux lignes de produits (f-roles). Le rôle UML n'étant conçu initialement que pour un système unique [88],[89],[90] , notre modèle en vue de le voir supporter une stratégie de développement multi-systémes (la ligne de produits) [91],[92] , a introduit un nouveau modèle , « le modèle d'évolution » scindé en partitions variables évolutives qui comporte chacune deux compartiments.Le premier représenté en formalismes du diagramme de classes avec une prise en compte des nouvelles associations 'role' et 'association' est consacré à la représentation des points variants sous forme de super-classes et leurs variants en sous-classes de roles évolutifs que nous avons appelé f-roles.Ces derniers peuvent évoluer selon quatre scénarios liés au gain et à la cession d'autres f-roles.
En vue d'éviter d'encombrer et de surcharger ce compartiment, un deuxième compartiment qui représente les mêmes classes mais sous une forme évolutive plus complète par l'intégration des contraintes d'évolution et les scénarios associés à travers les formalismes

d'une machine à état-transitions réduite .Il est important de noter que nous avons intégré dans les Meta-modèles UML tous les concepts et les mécanismes proposés. Pour bien montrer l'efficacité de notre modèle, nous l'avons appliqué à un domaine de l'embarqué celui du contrôle de la consommation en fuel dans un moteur automobile.

En vue de couvrir convenablement tous les aspects liés à cette thèse, nous avons organisé ce document en trois parties.

La première présente un état de l'art ; elle est divisée en deux chapitres:

Le chapitre 1 aborde les systèmes embarqués et leurs concepts fondamentaux compte tenu du fait que nos études sont recommandés pour les domaines de lignes de produits complexes c.à.d. à forte composante tels que le domaine des systèmes embarqués Ceci est dû au fait que ces derniers renfermant du logiciel enfoui avec au moins une fonction spécifique à exécuter, demeurent très évolutifs notamment en terme de technologique.

Le chapitre 2 rappelle les principes et les concepts de la stratégie de développement en lignes de produits et montre leurs aspects multidimensionnels. Pour cela, un état de l'art des modélisations en lignes de produits est présenté.

Les contributions de cette thèse sont détaillées dans la deuxième partie de ce document.

Cette partie comporte deux chapitres qui présentent nos contributions en réponse aux problématiques citées.

Le chapitre 3 aborde l'aspect développement d'une ligne de produits logiciels.

Le chapitre 4 illustre deux aspects liés à l'évolutivité des modèles de la ligne à savoir l'extensibilité par la modélisation des caractéristiques et l'évolution de la variabilité par les rôles.

Ces contributions sont illustrées par différentes études de cas relatives aux systèmes embarqués (automobiles et maisons intelligentes).

La troisième et dernière partie conclut le document et présente les perspectives des contributions présentées dans cette thèse.

Première Partie

État de l'Art

Chapitre I

Les Systèmes Embarqués

1.1 Introduction

De nombreux dispositifs électroniques sont omniprésents dans notre quotidien depuis plusieurs années : du téléviseur, console de Jeux, lecteur vidéo, téléphone mobile, assistant personnel, à la machine à laver, photocopieur, scanner, imprimante et autres.

Ce qui rassemble tous ces dispositifs c'est leur composition à partir d'éléments électroniques fonctionnant à l'aide de logiciels implantés. Ces groupes d'éléments électroniques et logiciels sont communément appelés systèmes enfouis.

Ces derniers si de plus, sont dotés chacun d'une tache bien déterminée, indépendante ou associée à l'environnement du dispositif électronique, sont alors appelés systèmes embarqués (SE).

Les systèmes embarqués se différentient par les applications implantées tant au niveau espace que fonctionnalités ainsi que par les performances requises pour l'exécution de leurs taches en termes d'efficacité, de consommation d'énergie, de réactivité ainsi que leur capacité à communiquer avec leur environnement extérieur [1],[2],[3]

Ces systèmes existent depuis plus de cinquante ans (le système de guidage de la fusée américaine Apollo en 1960 en est un exemple).

La nouveauté par rapport aux premiers systèmes est la convergence de technologies hétéroclites sur un seul système, qui sont : l'informatique, l'électronique, les télécommunications et les réseaux [4],[5].

Le domaine des Systèmes embarqués lié étroitement à ceux de l'industrie et de l'économie, ces derniers revêtent dès lors un caractère stratégique.

L'innovation et le progrès qu'ils apportent donnent une valeur ajoutée sure aux produits et services mis sur le marché [5].

Pour les particuliers, les nouveautés sans cesse arborées dans les fonctionnalités des produits apportent au consommateur un confort notamment à son domicile et à son travail.

L'évolution de ce secteur devrait augmenter surtout par la diversité et l'industrialisation des produits les utilisant [4],[5],[92].

Cette augmentation devra toucher davantage de nombreux domaines en particulier celui des transports (automobile, avionique, transport ferroviaire…), des télécommunications ainsi que les dispositifs électroniques à faible coût tels que ceux de la domotique, les applications multimédia (photo, vidéo, télévision interactive….).Ceci prouve que le domaine des systèmes embarqués demeure vaste et varié. Dans ce qui suit, nous allons donner quelques définitions relatives au système embarqué et les voies de son évolution.

1.2 Définitions et Évolutions

Nous essaierons d'approcher dans cette partie les définitions usuelles d'un système embarqué qui restent liées à ses propriétés, puis d'appréhender les aspects les plus importants de son évolution.

1.2.1 Définitions usuelles

La définition d'un "système embarqué" est fluide et difficile à cerner compte tenu de son évolution permanente attachée aux progrès technologiques et la diminution constante des coûts de production des différents composants matériels et logiciels .

L'embarqué est un terme très général qui regroupe plusieurs définitions selon le contexte utilisé. Au cours de la dernière décennie, le domaine des systèmes embarqués a suranné bon nombre de ses descriptions initiales classiques.

Ainsi, ce terme peut avoir deux utilisations selon les parties prenantes, d'une part les utilisateurs du marché de l'embarqué et de l'autre les spécialistes du domaine [92].

C'est pour ces deux raisons que nous allons nous limiter à donner les descriptions fixant les définitions des systèmes embarqués les plus courantes selon le contexte abordé.

A cet effet, nous avons retenu dans ce qui suit, les définitions liées aux aspects importants de ces systèmes à savoir : la structure, les limitations des composants Matériels/logiciels, les performances, la communication et la réactivité .

- **En termes de structure**

Un système embarqué est un dispositif électronique autonome qui contient du logiciel. Il est conçu pour exécuter une tache dédiée bien définie [1] .Un système embarqué ne possède généralement pas des entrées/sorties standards et classiques comme un clavier ou un écran.

Le système matériel et l'application sont extrêmement attachés, le logiciel embarqué étant enfoui dans le matériel. Cependant, nous rencontrons maintenant des dispositifs tels que l'assistant numérique personnel (PDA) et les téléphones cellulaires portables qui sont des systèmes embarqués conçus pour être en mesure de faire une variété de fonctions primaires.

En outre, les derniers téléviseurs numériques comprennent des applications interactives qui effectuent une grande variété de fonctions générales sans rapport avec la fonction de télévision, mais tout aussi importantes, telles que le e-mailing, la navigation sur internet et les jeux.

- **En termes de limitations de ressources Matérielles/Logicielles**

Les systèmes embarqués sont plus limités dans le matériel et/ou dans les fonctionnalités du logiciel que dans les ordinateurs personnels.

Selon [92], en termes de limitations matérielles, cela signifie des limitations dans la puissance du traitement, de la consommation d'énergie, de la mémoire et des fonctionnalités liées au matériel. Dans le logiciel, cela signifie généralement des limitations relatives c.à.d. avec moins d'applications, ou des applications à échelle réduite, ou encore avec une absence de système d'exploitation (OS) ou bien même avec un OS obtus ou avec moins de code au niveau abstraction.

- **En termes de performances**

Un système embarqué est un système semblable à un système d'ordinateur mais avec des exigences de qualité et de fiabilité plus élevées selon [92].

- **En termes de communication** [92]**:**

Une autre caractéristique importante d'un système embarqué est sa capacité à communiquer avec le monde extérieur. En termes de communications, les composants matériels et logiciels du système embarqué communiquent entre eux.

De même le système embarqué communique avec le monde extérieur dans lequel il évolue.

Cette nouvelle possibilité est liée aux progrès d'intégration de la micro-électronique mais aussi à la généralisation de l'usage des protocoles réseaux.

Les protocoles Internet (TCP/IP), standards de fait, ont ainsi pu être utilisés dans les systèmes embarqués, voire même pour certains, être directement intégrés dans le circuit.

Les processeurs utilisés dans un système embarqué intègrent en standard une interface réseau de type Ethernet. Il est aussi possible d'intégrer une connectivité Internet (filaire ou non filaire) par l'ajout d'un module électronique. Contrôler à distance, par réseau ou même par Internet via un navigateur Web un système embarqué est possible .Toutefois, ceci implique la prise en compte de l'aspect sécurité des communications.

- **En termes de réactivité**

Les Systèmes embarqués sont en général réactifs. Un système embarqué temps réel est un système réactif devant fournir des sorties logiquement correctes tout en respectant strictement des contraintes temporelles explicites [2].Il est considéré comme défaillant s'il ne respecte pas au moins l'une de ses spécifications logiques ou temporelles.

Un système embarqué temps réel fait intervenir selon [2],[93],[94] les dispositifs suivants :

1. Un calculateur dont les activités sont gouvernées par un système d'exploitation spécialement conçu pour les activités temps réel.
2. Un processus physique à commander, depuis lequel le système reçoit de l'information et sur lequel il exécute ses commandes.

3. Des dispositifs d'interconnexion entre les deux composants ci-dessus tels que les Transducteurs et les Actionneurs.

Le calculateur lit les données depuis les transducteurs et répond avec des commandes exécutées par les actionneurs dans des limites temporelles bien précises.

Il est important de constater que la plupart des Systèmes Embarqués sont des Systèmes Temps Réel et que la plupart des systèmes Temps Réel sont embarqués.

1.2.2 Différences essentielles entre les Systèmes embarqués et les Systèmes informatiques

Compte tenu de la frontière très étroite entre un système embarqué et un système informatique, nous avons jugé utile d'en apporter les différences les plus significatives dans leurs définitions.

Les systèmes embarqués se différentient essentiellement des systèmes informatiques d'entreprise (PC, Serveurs…) selon [92] par :

- Le type de réseaux et les types de contraintes matérielles et physiques.
- Le volume de l'encombrement.
- Les types de périphériques d'entrée.
- L'autonomie en énergie.
- La puissance de calcul.
- Le niveau de fiabilité,
- Le débit des flux informationnels véhiculés.
- Le coût.
- Le fonctionnement alterné online/offline,
- La configuration éphémère des éléments réseaux.
- Le poids

Ces contraintes influent sur la conception et l'architecture des systèmes d'exploitation et le développement des applications des Systèmes embarqués.

Etant donné que les systèmes embarqués sont dans leur grande majorité des systèmes temps réel, ils doivent respecter des contraintes temporelles fortes. C'est pour cette raison que l'on y trouve enfoui un système d'exploitation (ou noyau) Temps Réel (en anglais RTOS ou Real Time Operating System)[93],[94].

De plus, le Système embarqué utilise une famille de processeurs différente de celle du PC.

En termes de consommation d'énergie, il en consomme moins par rapport au PC car cet aspect reste primordial pour le SE. A partir de ce constat, un PC standard peut exécuter tout type d'applications car il est généraliste alors qu'un système embarqué n'exécute qu'une

application dédiée et unique. Un système embarqué n'est pas un ordinateur PC bien qu'un PC industriel puisse s'en approcher. En réalité, les ingénieurs de l'embarqué sont divisés quant à la classification de ces conceptions en systèmes embarqués même si actuellement ces dernières sont discutées en tant que tels parmi ces mêmes concepteurs .Ceci est dû au fait qu'il n'y a pas de nouveaux domaines supportés par l'industrie des systèmes informatiques pour les conceptions qui se situent entre le système embarqué traditionnel et les systèmes proches des PC à usage général. Le constat des définitions visant à soulever cette ambigüité dans la définition des SE s'ouvre ainsi sur trois alternatives selon [95]:

1. Se contenter des définitions traditionnelles de l'embarqué pour définir un SE ;
2. Reformuler les définitions des SE selon leurs évolutions technologiques ;
3. Designer un nouveau domaine , pour les Systèmes Informatiques dans le but d'inclure les systèmes plus complexes actuellement confus à définir tels que les PDA.

1.2.3 Evolution des Systèmes Embarqués

L'évolution des Systèmes Embarqués se dirige à grands pas selon [92],[95] vers :

1. Plus de miniaturisation ;
2. Plus d'intégration sur silicium ;
3. Plus de communications entre :
   - Les composants du Système Embarqué,
   - Le Système embarqué et son environnement immédiat
   - Le système embarqué et son environnement extérieur.

A ce titre, une autre discipline très proche des SE a vu le jour .Elle couvre tout ce qui touche à l'informatique diffuse et à l'électronique associée, c'est l'informatique ubiquitaire.

1.3 Les domaines de l'embarqué

Les grands secteurs couverts par l'embarqué sont les suivants :

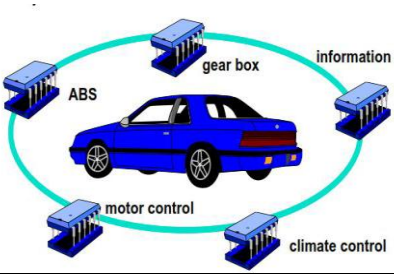
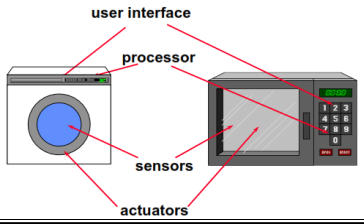
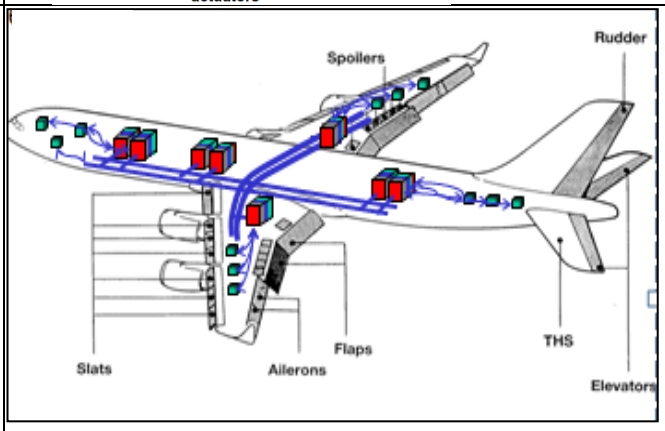
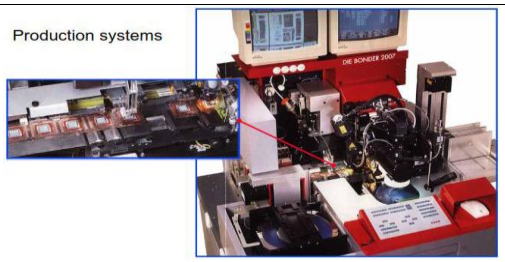
- L'automobile. Exemple : le système de freinage antiblocage (ABS).
- L'électronique grand public. Exemples : Le Téléviseur numérique, PDA, les mobiles..
- Les systèmes de contrôle des dispositifs industriels.

Exemples : la robotique et les systèmes de contrôle de fabrication.

- L'appareillage utilisé en médecine. Exemple : Les Machines de dialyse, scanners,...
- Le réseautage. Exemples : Les routeurs, Les hubs, Les passerelles,…
- L'informatique de bureau. Exemples : Le fax, photocopieurs, scanners, imprimantes,
- Le Traitement du signal. Exemples : les radars, les sonars, la compression vidéo, …

Le tableau ci-dessous illustre les plus importantes localisations par secteur des systèmes embarqués .

Tableau 1.1 Localisation par secteur des systèmes embarqués

Secteur	Localisation des systèmes embarqués
La Robotique	
L'Automobile	
L'électronique grand public	
Le Transport	
La Domotique	
Les Systèmes industriels	

1.4 La Structure type d'un Système Embarqué

En dépit des multiples définitions, selon le contexte, des systèmes embarqués, nous avons retenu dans notre thèse la définition relative à l'aspect structure.

En d'autres termes, un Système Embarqué est considéré comme un système électronique essentiellement numérique doté des propriétés suivantes [95]:

- Le Système Embarqué emploie un processeur embarqué ;
- Le Système Embarqué est dédié à une seule fonction.

Il réalise une fonction bien déterminée par l'exécution d'une seule application logicielle dédiée et enfouie dans le matériel ;

- Le Système Embarqué a des entrées obligatoires : il est doté d'une ou de plusieurs entrées multiformes (petit clavier matriciel, boutons poussoirs,…).
- Le Système Embarqué a des Sorties optionnelles.

Les dispositifs de sortie ne sont pas obligatoires et si elles existent, elles restent limitées.

Exemple : l'affichage pour une sortie typique est limité. Il peut prendre la forme de diodes de type LED, d'écran LCD, … .

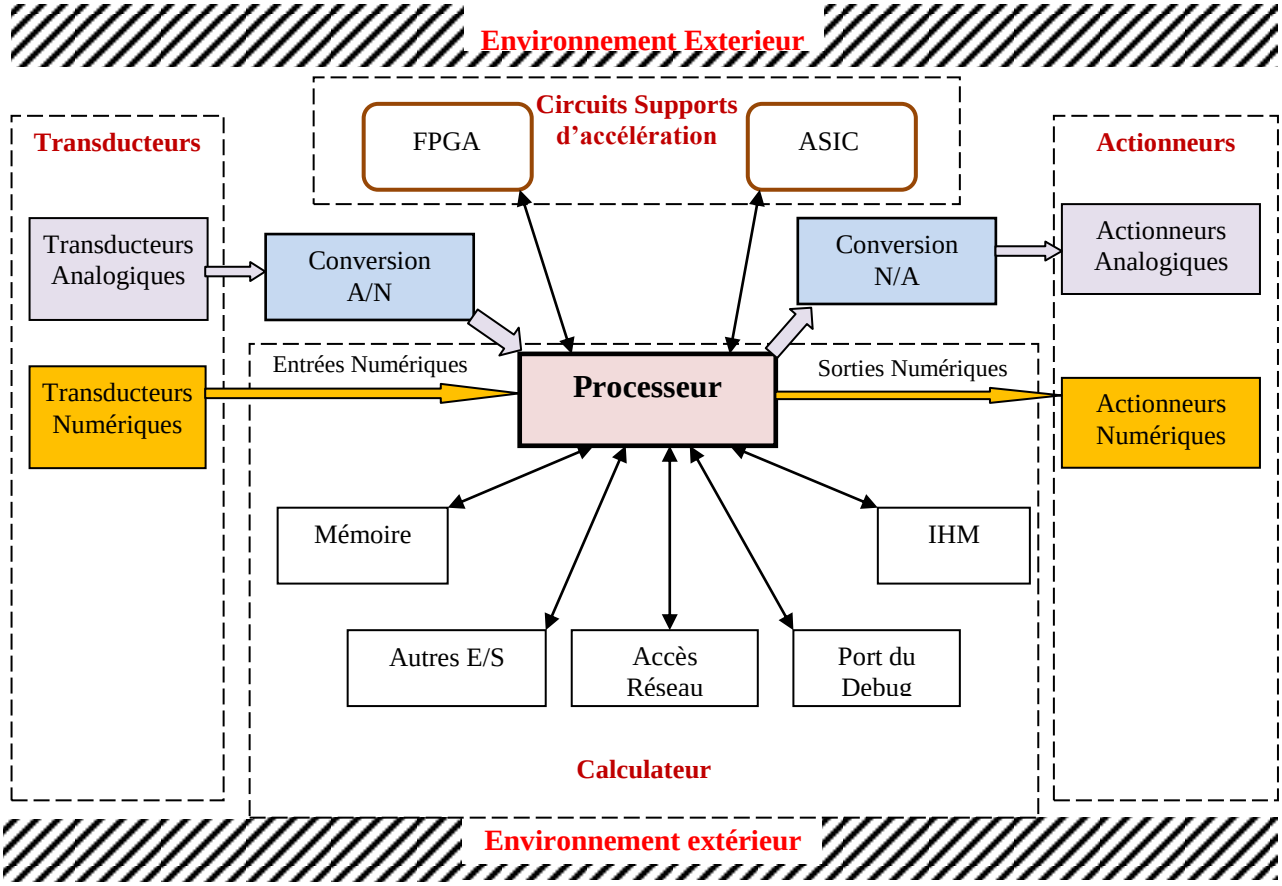


Figure 1.1. Organisation d'un Système embarqué typique

- Des circuits numériques ou mêmes analogiques (FPGA, ASIC …..) peuvent être employés en vue d'augmenter ses performances et/ou sa fiabilité à travers la mise en œuvre des accélérations matérielles du traitement ; le processeur étant chargé quant à lui d'apporter la flexibilité à la programmation logicielle.

La figure 1.1 présente une structure typique d'un système embarqué selon [3], [4] et [96].

1.4.1 Les composants clés

A partir de la structure typique du SE, émergent ses composants clés [3], [4] et [96] à savoir :

- Les Transducteurs :

En entrée du système, il peut y avoir un ou deux types de transducteurs analogiques et/ou numériques : les Capteurs et les Senseurs.

- Les Capteurs analogiques.

La communication avec le processeur du Système Embarqué étant numérique, les capteurs analogiques doivent être couplés à des convertisseurs de l'analogique vers le numérique.

- Les Capteurs numériques (sorties numériques).

Ils sont directement interfacés au processeur car ne nécessitant pas de convertisseurs vers le numérique.

- Les Actionneurs :

Situés en sortie, ils sont de deux types :

- Des Actionneurs généralement analogiques couplés cette fois à l'inverse des capteurs, selon le cas à des convertisseurs numériques ou analogiques.
- Des Actionneurs numériques qui sont directement interfacés au processeur.

- Le Calculateur :

Le noyau central du système embarqué est composé d'un calculateur mettant en œuvre un processeur embarqué avec ses périphériques d'Entrées/Sorties. Ce dernier emploie un ensemble de circuits assimilés à un Coprocesseur support au processeur du système embarqué dont l'objectif principal est de réaliser des accélérations matérielles pour le système.

Ces circuits peuvent être de deux types :

- Des circuits logiques programmables de type FPGA*(Field programming Gâte Array) ;*
- Des circuits intégrés de type ASIC *(Application Specific lntegrated Circuit)*.

1.4.2 Le fonctionnement typique d'un Système Embarqué

Le fonctionnement type d'un système embarqué est axé sur les activités suivantes [97], [98] :

- Les règles et les commandes de fonctionnement du système sont alors implantées dans le calculateur.
- Les capteurs collectent les informations à partir de l'environnement extérieur du SE.

- Les actionneurs exécutent les actions ordonnées par le calculateur.
- Le calculateur donne l'ordre d'exécuter les actions et les contrôle également.
- Le calculateur reçoit les informations recensées par les capteurs, les exploite, les traite puis ordonne et contrôle l'action des actionneurs.
- L'environnement extérieur :

Un système embarqué ne fonctionne pas toujours dans des conditions propices. Il doit dans la plus part des cas subir un ensemble d'environnements défavorables qu'il doit pouvoir surpasser.

Parmi ces environnements, nous citerons les plus importants qui sont liés :

- Aux variations fréquentes de la température ;
- Aux vibrations et aux chocs ;
- Aux variations et/ou aux coupures d'alimentation en énergie;
- Aux interférences de radiofréquences ;
- A la corrosion ;
- A l'infiltration de l'eau ;
- A l'exposition au feu ;
- A l'exposition aux radiations d'ions ;

Ces environnements de fonctionnement du système embarqué ne peuvent tous être contrôlés. Il existe même des environnements qui sont incontrôlables.

C'est pour cette raison que les concepteurs et les fabricants de Systèmes Embarqués sont tenus de bien considérer cet aspect lors de la conception et de la fabrication (Ils prévoient par exemple des parois étanches et ignifuges au système ...).

1.5 Les Exigences et les Contraintes

Les systèmes embarqués fonctionnent souvent dans des environnements physiques caractérisés le plus souvent par des ressources limitées, au niveau de la mémoire, de l'énergie (utilisation de batteries et/ou de panneaux solaires voire de piles à combustible) ou aussi des capacités de traitement.

De plus, compte tenues des fortes interactions des logiciels embarqués avec leur environnement, ces derniers sont confrontés à des contraintes de temps réel ou de sûreté de fonctionnement plus ou moins fortes selon le type d'utilisation du système embarqué.

En matières de contraintes et d'exigences, les systèmes embarqués doivent selon [92] et [93] :

- Satisfaire des contraintes de temps réel provenant des applications, qui peuvent être aussi bien des contraintes de temps réel souple (par exemple dans le cadre d'applications

multimédia) que des contraintes de temps réel strict (par exemple dans le cadre de protocoles de communication).

- S'adapter ; à la nature périssable de certaines ressources (énergie) par le fonctionnement ; aux limites du matériel (surchauffe) ; aux capacités de stockage limitées. Ils doivent également intégrer l'instabilité des systèmes de communication utilisés (déconnexion, bande passante).
- Offrir une puissance de traitement suffisante pour satisfaire les contraintes de temps d'exécution des applications. Les processeurs utilisés dans les systèmes embarqués sont 2 à 3 décades moins puissants qu'un processeur d'un ordinateur PC.
- Revenir au moindre cout surtout lorsque le système est produit en grande série.
- Etre doté d'une grande sureté de fonctionnement surtout pour les situations de défaillance qui mettent en péril des vies humaines où des investissements considérables (explosion de la fusée américaine Discovery).
- Ce type de systèmes sont dits « critiques » et ne doivent jamais faillir c.à.d. qu'ils doivent toujours donner des résultats corrects, pertinents et dans les délais attendus par les utilisateurs que ce soient des machines et/ou des humains.
- Avoir un poids efficient (un juste poids nécessaire au fonctionnement).

1.5.1 Les Contraintes de temps

A cause de la demande toujours croissante de puissance de traitement des applications, la performance globale du système embarqué liée au temps est un critère de conception important.

1.5.1.1 La performance temporelle

Dans la conception d'un système embarqué, la performance temporelle (exécution des taches en un temps court voire très court) des dispositifs est une préoccupation constante. C'est au niveau logiciel que cette performance s'exprime. En vue d'obtenir une meilleure performance temporelle, il y a lieu d'agir sur les algorithmes en vue de réduire leur complexité, ainsi que sur les séquences fréquentes de code en les optimisant (la réutilisabilité).

1.5.1.2 Les systèmes embarqués temps réel

Beaucoup de systèmes embarqués interagissent directement avec leur environnement via des capteurs/actionneurs ou un réseau de communication sans fil. Ces interactions contraignent les temps de réponse du système embarqué de manière plus ou moins forte selon le domaine d'applications visé. Ce sont les systèmes temps réel ou plus précisément les systèmes embarqués temps réel [96],[98],[99] dans le sens où la durée de livraison des résultats d'un calcul fait partie intégrante de la spécification de ce dernier, au même titre que le résultat

global. Il existe deux types de Systèmes Embarqués, les Systèmes Embarqués *temps réel strict* et les Systèmes Embarqués *temps réel souple*.

a. Les Systèmes embarqués temps réel strict

Dans les systèmes 'temps réel strict', ou dur, le non-respect des contraintes de temps du système, le plus souvent exprimées sous la forme d'échéances de terminaison, constitue une défaillance de l'application [99]. Dans le cadre d'applications critiques, telles que par exemple certaines applications dans l'avionique, une telle défaillance peut avoir des conséquences catastrophiques, telles que la mise en danger de vies humaines ou des pertes financières importantes.

Etant donné les conséquences importantes du non-respect d'une échéance dans les systèmes temps réel strict, il est fortement recommandé pour de tels systèmes de pouvoir vérifier que toutes les échéances soient toujours respectées avant leur exécution.

Pour cela, des méthodes d'analyse d'ordonnançabilité [100],[101] doivent être utilisées afin de procéder à cette vérification.

b. Les Systèmes embarqués temps réel souple

Dans les systèmes 'temps réel souple', bien que l'instant d'obtention du résultat soit important, la violation des contraintes de temps du système est tolérée même si elle demeure exceptionnelle [99],[100],[101].

Cette tolérance est due au fait que les applications concernées ne relèvent pas du domaine des applications critiques. Les exemples typiques d'applications ayant des contraintes temps réel souples sont les applications multimédias à flux continus (transmission télévisée en direct… : le système vise au respect des contraintes de temps dans la délivrance des flux de données afin de garantir la qualité des images et du son). Toutefois, les contraintes de temps peuvent être adaptées [99] puisque une dégradation de la qualité des données ne sera que faiblement perçue par l'utilisateur.

1.5.2 Les contraintes de Consommation énergétique

Une grande majorité des systèmes embarqués (téléphones mobiles,…) est confrontée au problème d'autonomie d'énergie. Aussi, afin d'étendre l'autonomie de fonctionnement de tels systèmes, deux approches sont actuellement utilisées [101] :

- L'approche qui vise à augmenter la capacité de stockage des batteries ;
- L'approche qui vise à réaliser un système embarqué à faible consommation énergétique.

Dans le cadre de cette dernière approche, plusieurs méthodes ont été envisagées qui palpent à la fois le domaine de l'électronique et du logiciel [101].Elles sont axées sur :

- La conception de composants électroniques consommant le minimum d'énergie;

- L'optimisation du logiciel afin de diminuer le coût énergétique de son exécution;
- La conception de stratégies logicielles exploitant au maximum les fonctionnalités du matériel.

L'optimisation du logiciel consiste à privilégier les instructions moins consommatrices d'énergie. Pour l'optimisation d'applications de type protocole réseau sans fil, cette dernière porte sur la réduction du volume des communications afin de diminuer la consommation d'énergie occasionnées par l'utilisation du réseau.

De plus, la conception d'une stratégie logicielle exploitant des fonctionnalités du matériel afin de diminuer la consommation énergétique touche principalement à l'ordonnancement [100],[101].

Certains processeurs offrent différents modes d'exécution. Outre le mode d'exécution nominal, un mode veille permet à la fois d'endormir à faible coût le processeur et également de le réveiller de manière rapide.

1 5.3 Les contraintes de Mémoire

Dans un grand nombre de systèmes embarqués, la mémoire est une source limitée (dans un téléphone portable, elle est de quelques Mégaoctets), et par conséquent une bonne utilisation de la ressource mémoire est cruciale pour ces systèmes.

Une difficulté supplémentaire dans les systèmes embarqués est que la gestion de la mémoire soit compatible avec les contraintes temps réel des applications, qu'elles soient souples ou strictes.

1.5.4 La contrainte de tolérance aux fautes

Certains systèmes embarqués doivent pouvoir remplir leurs fonctions malgré la présence de fautes [92], qu'elles soient d'origine physique ou humaine. Pour le premier type de faute, il est nécessaire :

1. De détecter les erreurs, par l'utilisation accrue de méthodes, telles que les codes détecteurs d'erreurs, les contrôles de vraisemblance ou encore le diagnostic en ligne,
2. D'effectuer un recouvrement d'erreurs. Ceci permet au système de continuer à remplir ses fonctions malgré l'erreur, que ce soit par reprise de son exécution à partir d'un état sauvegardé au préalable (point de reprise) ou par compensation exploitant la redondance présente dans le système (par exemple la duplication active de tâches).

Les difficultés issues du contexte de l'embarqué sont relativement liées aux contraintes de temps ainsi qu'aux ressources limitées de l'architecture. Ceci a contraint les concepteurs à recourir aux méthodes de tolérance aux fautes utilisables dans un contexte embarqué temps

réel. Cependant, dans les systèmes temps réel strict, il est nécessaire d'intégrer les mécanismes de tolérance aux fautes dans l'analyse d'ordonnançabilité du système [100].

1.6 L'architecture des Systèmes Embarqués

Du point de vue de l'ingénierie des systèmes, la conception de l'architecture des systèmes embarqués a pu être approchée selon plusieurs modèles. Ces derniers sont utilisés pour décrire le cycle de la conception du système embarqué.

L'architecture d'un système embarqué est une abstraction du matériel embarqué, c.à.d. qu'il s'agit d'une généralisation du système qui ne montre pas généralement le détail des informations d'implémentation telles que le code source du logiciel ou la conception de circuit matériel[96],[100],[101].

Au niveau architectural, les composants matériels et logiciels dans un système embarqué sont plutôt représentés comme une composition d'éléments en interaction.

Ces éléments sont des représentations de matériel et / ou logiciels dont les détails ont été mis en œuvre abstraitement, laissant accès seulement aux informations comportementales et interrelationnelles.

Les Éléments architecturaux peuvent être intégrés en interne au sein du dispositif embarqué, où à l'extérieur du système embarqué et interagir avec les éléments internes .Typiquement, une architecture embarquée comprend [96]:

- Les éléments du système embarqué ;
- Les éléments qui interagissent avec le système embarqué ;
- Les propriétés de chacun des éléments ;
- Les relations interactives entre les éléments.

L'information de niveau architectural est physiquement représentée sous forme de structures.

Une structure est une représentation possible de l'architecture, contenant les informations sur son propre ensemble d'éléments représentés, ses propriétés et les interrelations entre ses éléments.

Une structure est donc un «instantané» du système en termes de matériel et de logiciel au moment de la conception et / ou de l'exécution du système, pour un environnement particulier et un ensemble donné d'éléments.

Du fait de l'incapacité pour un «instantané» de capturer toute la complexité d'un système, l'architecture sera typiquement composée de plus d'une structure.

Toutes les structures à l'intérieur d'une architecture sont intrinsèquement liées les unes aux autres et l'architecture du dispositif embarqué est la somme de toutes ces structures [96].

1.6.1 Le Modèle architectural du Système Embarqué

Le modèle architectural typique d'un système Embarqué indique que tous les systèmes embarqués partagent une similitude au plus haut niveau [96], c'est-à-dire qu'ils ont tous au moins une couche matérielle dans laquelle tous les composants (matériel, logiciel système et logiciel d'application) convergent. Typiquement, les couches de ce modèle sont :

- La **couche matérielle** qui contient tous les principaux composants physiques situés sur une carte embarquée,
- Les **couches du système ;** et
- Les **applications logicielles** qui contiennent toutes les logiciels, situés sur / et en cours, de traitement par le système embarqué.

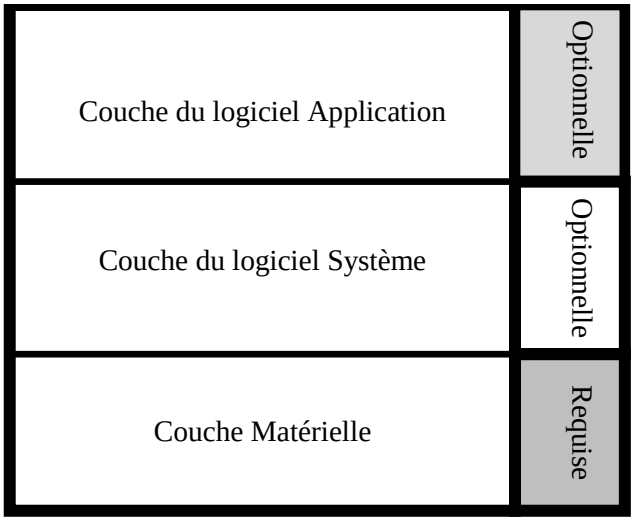


Figure 1.2 Le Modèle architectural type d'un système embarqué

Ce modèle de référence illustré sur la figure 1.2 est typiquement une représentation en couches (modulaire) d'une architecture du système embarquée à partir de laquelle une structure architecturale modulaire peut être dérivée.

1.6.2 L'importance de l'architecture d'un Système Embarqué

Les défis rencontrés lors de la conception de l'architecture d'un système embarqué sont selon [96] :

- La définition et la capture de la conception du système ;
- Les limitations de coûts ;
- La détermination de l'intégrité du système (fiabilité, sécurité ….) et de la capacité de travailler dans les limites de la fonctionnalité élémentaire disponible (la puissance de traitement, la mémoire, la durée de vie de la batterie, …) ;
- L'étude de la commercialisation (l'étude du marché);

- Connaitre toutes les exigences déterministes.

Ainsi, la définition et la compréhension de l'architecture d'un système embarqué deviennent un élément essentiel pour la bonne conception du système. Ceci revient aux avantages qu'elle offre .Nous citerons dans ce qui suit les plus importants :

1. Chaque système embarqué doit être doté d'une architecture , même si cette dernière n'est pas documentée. Ceci est dû au fait que chaque système embarqué est composé d'éléments (matériels et/ou logiciels) en interaction. De plus, la définition de l'architecture doit être réalisée à priori (en début de projet).Ceci évitera au concepteur d'avoir une architecture défectueuse et coûteuse.

Parce qu'une architecture embarquée capte différentes vues qui sont des représentations du système, elle est un outil utile pour comprendre tous les éléments majeurs (Utilité et comportement de chaque composant).

De plus, aucun des éléments au sein d'un système embarqué ne peut fonctionner seul. Il doit toujours interagir avec un autre élément.

A l'extérieur, les caractéristiques visibles des éléments peuvent varier en fonction d'un ensemble différent d'autres éléments.

Sans la connaissance du rôle de la fonctionnalité et de sa performance par rapport aux autres fonctionnalités fournies par chaque élément, il sera difficile de déterminer la façon dont le système se comporterait lorsqu'il sera soumis à une variété de contraintes dans le monde réel.

2. L'architecture exprime les éléments essentiels d'une conception et les relations qui les lient. Elle peut fournir aux membres du projet de fabrication du Système des informations essentielles quant à savoir si ce dispositif est capable de répondre à leurs besoins et comment il peut être fabriqué avec succès.

Cependant, la maitrise de la conception de l'architecture impose la convergence de l'effort d'une équipe pluridisciplinaire de spécialistes en électronique, en développement informatique et en télécommunication.

1.7 Le développement dans l'embarqué

Dans ce qui suit, nous allons donner un aperçu sur les systèmes de développement en matières de langage et de contraintes liées au développement puis nous allons nous étaler sur les principaux modèles de développement et nous terminerons par l'exposition des techniques les plus significatives en matière de développement des Systèmes Embarqués.

1.7.1 Les Langages et les Contraintes

Nous aborderons dans cette partie , les principaux langages et contraintes de développement

ainsi que les systèmes de développements liés aux systèmes embarqués avec un aperçu sur les avantages qu'ils apportent.

1.7.1.1 Les Contraintes de développement

Les systèmes embarqués présentent des aspects spécifiques nécessitant des approches adéquates de développement [99]. Ces dernières doivent répondre aux contraintes de développement suivantes :

1. La disponibilité limitée des ressources (mémoire, processeur, moyens de communication.. …….) ;
2. Les contraintes de temps qui sont souvent imposées.

De telles contraintes ne peuvent être satisfaites que si le logiciel se comporte d'une façon suffisamment prédictible [93].

Les contraintes temporelles strictes et les processus de perturbations fortement liés qui caractérisent souvent les systèmes embarqués, demandent l'utilisation de techniques de développement adaptées.

1.7.1.2 Les Systèmes de développement

Le développement de systèmes renfermant des composants informatiques et électroniques en perpétuelle complexité, nécessite des méthodes puissantes et fiables en vue de s'adapter à des contraintes de développement toujours plus critiques [99].

De plus, la mise en réseau et la complexité augmentent alors que le temps de développement à la disposition des développeurs se réduit.

Toutefois, une grande partie du logiciel embarqué est écrite en assembleur ou en C/C++ , java et autres langages similaires.

Des critères, tels que la modularité, la compatibilité, la réutilisation, la facilité de maintenance, la fiabilité et la documentation, ont pris une importance capitale dans les systèmes embarqués souvent développés pour durer.

1.7.1.3 Les Avantages

Pour les avantages qu'il apporte (réutilisation, encapsulation, modularité), le modèle orienté objet est de plus en plus utilisé pour le développement des systèmes embarqués.

Basées sur UML comme langage standard de modélisation de systèmes informatiques, des méthodes ont été proposées [100],[101] et [102] en plus de SystemC.

Elles permettent d'établir la spécification d'ensemble, la conception générale, et la conception détaillée en sous-modèles décrivant chaque aspect du système (fonctionnel, structurel, comportemental, de traitement; matériel: architectural, de performance, de consommation, temps de réponse, taille mémoire…), indépendamment de sa programmation finale en C, C++

ou Java. Un autre avantage de ces méthodes est l'outillage automatisé, aux termes de la spécification détaillée, par exemple, où les sous-modèles doivent pouvoir être assemblés et affinés pour l'étape suivante par des outils logiciels automatisés [2].

La même logique est appliquée pour la génération et l'exécution des tests de validation, qui doivent se définir et s'exécuter en mode automatique.

1.7.2 Les Modèles de développement des Systèmes Embarqués

Le développement des Systèmes Embarqués peut être basé sur plusieurs modèles. Cependant les deux modèles les plus utilisés lors de la dernière décennie sont :

1. Le modèle en V
2. Le modèle en échelle (CoDesign) ou en Co-conception Hardware/Software (H/S).

1.7.2.1 Les Premiers Modèles

Les premiers modèles avant l'avènement du modèle en V sont organisés dans leur majorité en quatre phases selon [102] et [103] :

1. La création de l'architecture ;
2. L'implémentation de l'architecture ;
3. Le test du système ; et
4. La maintenance du système.

1.7.2.2 Le modèle en V

Dans ce modèle, le développement est réalisé selon des étapes séquentielles (figure 1.3) dans lesquelles les résultats de l'une sont versés dans l'étape suivante [2]. Ces étapes sont :

1. L'analyse des besoins du produit ;
2. La mise en œuvre de l'architecture système ;
3. La conception du système ;
4. La mise en œuvre de l'architecture logicielle ;
5. La conception de l'application ;
6. Les tests unitaires ;
7. Le test d'intégration ;
8. L'intégration finale du système (logicielle et matérielle) ; et
9. La phase d'utilisation du produit.

L'inconvénient majeure de ce modèle est l'obligation de retour-arrière en cas de détection d'anomalies ou dans l'une des étapes puis la reprise non pas à partir de l'étape où l'anomalie a été détectée mais à partir de celle correspondant à la source de l'anomalie.

Le modèle permet aux corrections d'être incorporées dans le processus de développement du fait que toutes les étapes qui suivent cette anomalie soient revues et/ou corrigées. Ceci donne un aspect fastidieux au processus.

Cependant et en dépit de cet inconvénient, ce modèle a été longtemps adopté jusqu'à l'avènement du modèle de Co-conception.

1.7.2.3 Le Modèle en Co-conception Matérielle/Logicielle

Le modèle de conception conjointe ou Co-conception est un modèle plus récent appelé également modèle en échelle dans lequel toutes les étapes relatives au Matériel et au Logiciel sont réalisées chacune d'un côté de l'échelle avec possibilité de communication au niveau de chaque étape et donc pendant le processus de développement.

La définition du Co-conception est le résultat d'un enrichissement progressif dans le temps par les chercheurs du domaine.

La plus ancienne fut donnée par [102] qui le présente comme « *un processus de conception des systèmes qui combinent les perspectives du Matériel et du logiciel dès les premières étapes du processus de conception afin d'exploiter la flexibilité du produit système et une allocation de fonctions efficiente* ».

Un enrichissement important de cette définition fut proposé en 1994 par [103].Cette définition insiste sur « *la conception simultanée du matériel et du logiciel en vue du bon fonctionnement de l'implémentation du produit et du meilleur apport prix-qualité-flexibilité* ».

[104] quant à lui insiste sur « *le contrôle permanent du comportement global du système (matériel et logiciel) tout au long du processus de conception du produit ainsi que sur l'exploration optimale de l'espace des solutions à toute étape* ».

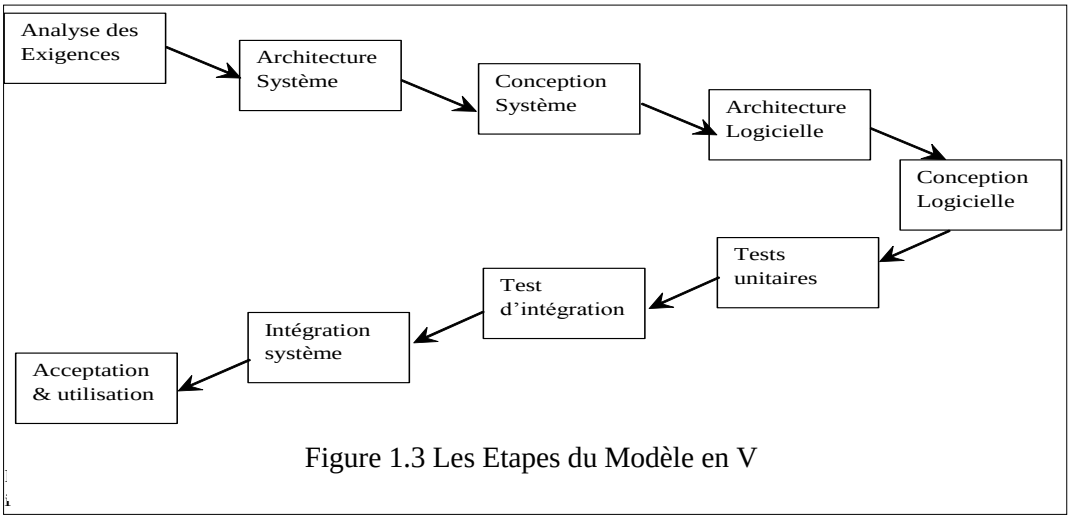


Figure 1.3 Les Etapes du Modèle en V

La figure 1.4 montre l'organisation du modèle de Co-conception.

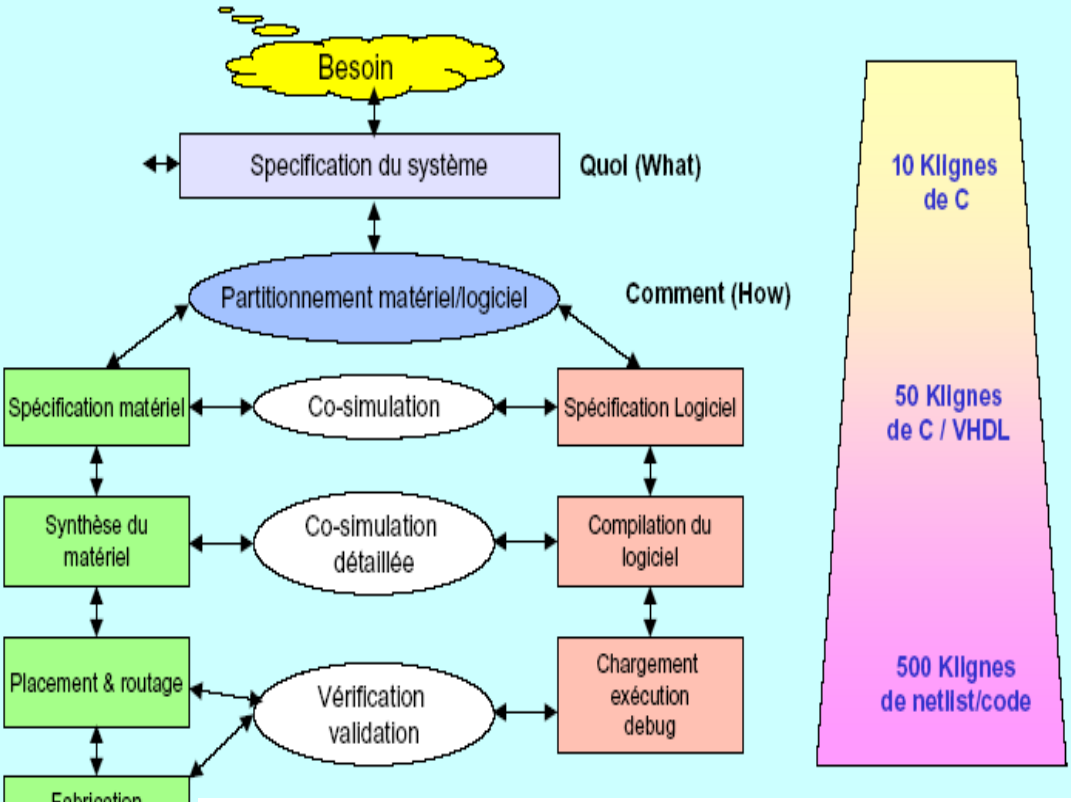


Figure 1.4 les Co-étapes du modèle de Co-conception

Le processus soumis aux contraintes temporelles, de consommation d'énergie et de surface est constitué des Co-étapes matériel/logiciel suivantes :

i. La Spécification du système.

ii. Le Partitionnement Matériel/Logiciel qui consiste à l'exécution des tâches suivantes :
   1- La Sélection des types et nombres des unités de traitement Matérielles et logicielles ;
   2- L'Allocation des fonctionnalités aux unités ;
   3- L'Ordonnancement.

iii. La Co-spécification matérielle et logicielle qui est vérifiée par une Co-simulation.

iv. La Synthèse des interfaces matérielles et la compilation du logiciel sont vérifiées par une Co-simulation.

v. Le Placement-Routage , le Changement-Exécution et le Débogage sont vérifiées et validées.

vi. En fin du processus, la Fabrication est vérifiée et validée.

1.7.3 Les Techniques de développement des Systèmes Embarqués dans le contexte des lignes de produits

Motivés par le succès croissant de l'approche Ligne de produits (LdP), un certain nombre de techniques ont été proposées en faveur du domaine des Systèmes Embarqués. Elles se sont focalisées sur le développement, la maintenance et le déploiement des SE au-delà des approches existantes orientées ingénierie du domaine et basées sur le paradigme Ligne de produits. Nous en citerons les plus significatives.

1.7.3.1 La méthode PuLSE

PuLSE (**P**rod**u**ct **L**ine **S**oftware **E**ngineering) ou l'ingénierie logicielle des lignes de produits [105] est une méthode conçue pour supporter les clients des produits de systèmes embarqués dans le développement en ligne de produits. Elle a été réalisée par le ' Fraunhofer **I**nstitute **E**xperimental **S**oftware **E**ngineering (IESE)'.

Cette méthodologie stipule que le développement de la LdP doit se focaliser sur le produit lui-même et non sur les questions d'organisation des artefacts liés à ce paradigme.

Les avantages de cette méthode reposent sur les apports suivants :

- La disponibilité d'un cadre complet.

Ce denier couvre le cycle de vie entier de développement de la LdP, incluant la construction des infrastructures d'utilisation, de réutilisation et d'évolution.

- La modularité et la personnalisation.

Elles consistent en six composants techniques sélectionnables et instanciés en vue de satisfaire les besoins spécifiques de compagnies.

- L'incrémentabilité.

PuLSE peut être introduite graduellement selon l'étape par l'agrandissement progressif des processus de développement des logiciels déjà existants ainsi que des nouveaux produits et ce à l'aide des aspects spécifiques de la ligne de produits dans chacune des étapes .

PuLSE dans sa version basique est composée de 3 éléments principaux :

- Les résultats des phases de déploiement ;
- Les composants techniques ; et
- Les composants supports.

L'idée principale est que durant les phases de déploiement, les composants techniques et les composants supports sont utilisés et personnalisés en vue d'élaborer une LdP spécifique.

1.7.3.2 La méthode KobrA

KobrA **('Ko**mponent **b**asie**r**te **A**nwendungsentwicklung' pour 'Component-Based Application Development') ou 'Développement d'Applications Basé sur les Composants'

développé par [106] , c'est une méthode conçue par l'institut allemand (IESE) en faveur de l'ingénierie des composants logiciels.

D'un point de vue pratique, KobrA peut être perçue comme une personnalisation "prête à utiliser" de la méthode PuLSE.

Cette méthode systématique est désignée pour l'ingénierie ligne de produits basée sur les composants. L'innovation clé est l'intégration de l'ingénierie LdP avec les approches basées sur les composants qui supportent peu la réutilisation.

L'apport étant la complémentarité des points forts de ces deux types d'approches.

A ce titre, KobrA est principalement scindée en deux activités : "L'ingénierie cadre" et "L'ingénierie application ".

- L'activité de l'ingénierie cadre ou 'Framework Engineering' est basée sur l'approche LdP en vue de concevoir et de maintenir un cadre générique décrivant les variabilités et les commonalités. Le cadre est organisé en un ensemble statique de Komponents ou composants KobrA sous forme d'un arbre.

Les résultats générés sont des modèles cadres définis en termes d'une mixture de textes et de modèles basés sur UML.

- L'activité de l'ingénierie d'application est basée sur l'approche basée composants en vue d'instancier les cadres génériques.

Chaque Komponent est décrit en deux niveaux :

1. Le niveau « Spécification » qui définit les propriétés externes visibles et les comportements des Komponents .
2. Le niveau « Réalisation ». Il décrit comment le composant « Komponent » de niveau supérieur est décomposé en Komponents de bas niveau.

Les résultats générés représentent une application.

A noter que la structure statique des composants KobrA est exprimée par des diagrammes de classes du langage UML.

La variabilité contenue dans la LdP est introduite à l'aide du stéréotype : <<variant>> qui peut être appliqué tout aussi aux paquetages, qu'aux classes, attributs et méthodes.

1.7.3.3 La méthode FAST

FAST (**F**amily-oriented **A**bstraction, **S**pecification and **T**ranslation) ou « abstraction , spécification et translation orientée famille ».

Conçue par [107] , c'est l'une des premières méthodes avec FODA qui est basée sur l'ingénierie LdP (ingénierie du domaine et ingénierie de l'application). Elle a été proposée par Weiss à partir de Lucent.

Cette méthode fournit une approche systématique pour analyser les familles potentielles de la ligne et de développer des mécanismes pour la production efficiente des membres de la famille de produits.

1.7.3.4 La technique d'analyse FODA

FODA (**F**eature-**O**riented **D**omain **A**nalysis) ou analyse du domaine orientée caractéristiques est l'une techniques pionnières dans le contexte des LdP . Elle a été développée par Kang [108]. Cette technique d'analyse du domaine se présente comme une succession d'étapes :

1. **L'analyse du contexte**.

permet l'identification du domaine et la construction du modèle du contexte

2. **La modélisation du domaine**.

permet la sélection et l'analyse des besoins et des tendances dans le but d'identifier la commonalité et la variabilité essentielles de la ligne.

3. **La modélisation de l'Architecture**. est issue à partir des ensembles de caractéristiques.

Elle permet de générer le modèle d'architecture connu aussi sous le nom de model conceptuel de référence ou modèle des caractéristiques qui par son instanciation permet de développer les applications.

1.7.3.5 Le processus RSEB

RSEB (**R**euse-Driven **S**oftware **E**ngineering **B**usiness) ou Métier de l'ingénierie logicielle dirigée par la réutilisation. Conçue par [109], [110] , RSEB est un processus de réutilisation créé pour accomplir des objectifs de métiers clés et améliorer la performance métier .

Ses étapes se concentrent sur :

- L'élaboration des cas d'utilisation,
- La description des exigences de la LdP,
- L'élaboration de l'architecture de la LdP et enfin,
- La mise en place des modèles objet directement liés aux cas d'utilisation initiaux.

1.7.3.6 La méthode FORM

FORM (**F**eature-**O**riented **R**euse **M**ethod) ou méthode de réutilisation orientée caractéristiques. FORM combinée aux architectures de référence du domaine spécifique d'un système embarqué qui ont été développés par [111], est considérée comme une extension de FODA qui cherche à couvrir l'analyse du domaine et le développement des principaux Assets tout en gardant une vue métier sur le développement dans le contexte des LdP et notamment en direction des Systèmes Embarqués. L'Asset étant l'unité de construction élémentaire des composants d'une ligne de produits logiciels. Il est considéré également comme l'élément de base de la réutilisation qui permet de construire tout produit de la ligne. L'Asset est

multiforme. Il peut prendre l'aspect d'un document de spécification, référencer un ensemble d'exigences similaires, décrire un modèle ou un sous-modèle, désigner un module indépendant ou enfin représenter une liste de code de logiciel qui réalise une même tache. Plus de détail sur le rôle de l'Asset sont fournis dans le chapitre 2.

FORM étend FODA aux étapes de conception logicielle et d'implémentation et prescrit comment le modèle des caractéristiques est utilisé pour développer les architectures du domaine et les composants pour la réutilisation.

FORM commence par construire le modèle de caractéristiques. Ce dernier permet de capturer les commonalités telles un « graphe AND/OR » , ou les nœuds « AND » indiquent les caractéristiques obligatoires et les nœuds « OR » indiquent les caractéristiques alternatives sélectionnables pour les différentes applications. Le modèle est alors utilisé pour définir les architectures paramétrées de référence et les composants appropriés réutilisables instanciés pendant le développement d'applications.

1.7.3.7 Le processus FeatuRSEB

FeatuRSEB (Featuring RSEB) ou caractérisation de RSEB a été conçue par [110] .Elle est une combinaison entre la méthode FODA et la méthode RSEB. Le principe est d'ajouter deux autres étapes basées sur FODA pour initier le processus RSEB à savoir : L'ingénierie du domaine et la modélisation des caractéristiques. De plus, une nouvelle notation est proposée.

Le diagramme des caractéristiques de FODA prend la forme d'un arbre ou d'un réseau de caractéristiques qui sont liées ensemble par des dépendances et des raffinements UML.

Cette méthode explicite les points de variation et les variants comme des artefacts se trouvant dans les cas d'utilisation et les modèles objet par l'usage des extensions d'UML.

Le processus de développement est défini, compte à lui, selon les étapes suivantes :

1. L'ingénierie des exigences .

où la variabilité est capturée en structurant les cas d'utilisation. Toutefois, un modèle de cas d'utilisation de haut niveau doit être établi pour décrire le contexte système ;

2. L'ingénierie de la famille d'architecture: où une architecture en couches est développée ;
3. L'ingénierie des systèmes de composants système :

dans laquelle les composants réutilisables sont développés ;

4. L'ingénierie des systèmes d'Applications : où les Applications sont développées.

1.7.3.8 La méthodologie ConIPF

ConIPF (**Con**figuration of **I**ndustrial **P**roduct **F**amilies) ou « configuration des familles de produits industriels » est une méthodologie conçue par [112] qui cherche à intégrer ensemble l'approche LdP et les technologies de configuration orientées structure.

La méthodologie suit un développement descendant, débutant à partir des exigences définies à travers les études de cas des partenaires industriels de systèmes embarqués et qui seront validés à travers des applications concrètes.

L'idée principale est d'appliquer et d'adapter les méthodologies de configuration développés dans le contexte de l'intelligence artificielle dans l'embarqué pour le contexte des lignes de produits en se focalisant sur le développement par la réutilisation.

ConIPF est une approche qui permet de définir et valider une méthodologie de dérivation d'un produit .Elle est surtout applicable pour les applications industrielles.

La méthodologie ConIPF est structurée en deux parties, chacune d'elle traite les taches de la méthodologie ConIPF qui seront réalisées dans les deux niveaux de l'ingénierie de la famille de produits c.à.d. l'ingénierie des applications et l'ingénierie du domaine.

- L'ingénierie des applications .

utilise un modèle de configuration pour créer des produits individuels. Ce processus nécessite deux tâches principales en corrélation qui sont la configuration et la réalisation.

- La tâche de configuration comprend la création des différentes parties de la description d'un produit.
- La tâche de réalisation développe le produit en sélectionnant les artefacts à partir des Assets stockés puis définit les paramètres comme décrits dans la description du produit.

- La deuxième partie est l'ingénierie du domaine.

Elle réalise la création et la maintenance du modèle de configuration. Pour créer ce dernier, les connaissances appropriées doivent être sélectionnées dans la tâche d'élicitation des connaissances puis transformées en modèle de configuration en utilisant les orientations et les directives de la tache de modélisation du domaine.

1.7.3.9 La méthode SPLIT-Daisy

SPLIT (**S**oftware **P**roduct **L**ine **I**ntegrated **T**echnology) ou « technologie intégrée de ligne de produit logiciels ». Conçue par [113], c'est une méthode développée dans un laboratoire commun entre Alcatel et Thales appelé LCAT.

SPLIT-Daisy s'intéresse à la description des architectures de lignes de produits dédiées aux systèmes embarqués par l'usage du langage UML.La variabilité est introduite par la notion de point de variation. Un point de variation est représenté comme une classe UML stéréotypée <<variation-point>>avec un ensemble d'attributs. Un point de variation est associé à un élément particulier de modèle statique (une classe ou un paquetage UML) .Il est défini par un ensemble de classes ou de paquetages variants. Un point de variation est défini aussi par un mécanisme de variabilité ; il est représenté comme un stéréotype appliqué à une association

reliant le point de variation et son élément. SPLIT introduit trois mécanismes de variabilité : **<<extend>>, <<insert>>** et **<<parametrize>>.**

1.8 Conclusion

Ce chapitre présente un aperçu sur notre domaine d'étude : les systèmes embarqués avec une focalisation particulière sur l'aspect état de l'art des méthodologies de développement liées au paradigme 'ligne de produits logiciels'. Ce dernier contexte étant le plus utilisé dans le cadre des systèmes embarqués.

Cette partie de la thèse commence par définir le système embarqué en prenant en compte la plupart des innovations complexes et récentes du marché et ses parties prenantes.

Un aperçu est ensuite donné sur les différents secteurs de l'embarqué avant d'appréhender le rapprochement des définitions entre les systèmes embarqués et les systèmes informatiques.

Une présentation de la structure typique d'un système embarqué est ensuite appréhendée. Elle sera suivie par une exposition des contraintes auxquelles sont soumis les systèmes embarqués pendant leurs fonctionnements ont été ensuite appréhendées.

Une définition de l'architecture du système embarqué en termes de somme de différentes représentations d'un même système a été subséquemment abordée.

L'importance de l'architecture a été ensuite traitée en montrant la nécessité de la doter d'une aptitude maximale à fournir des indicateurs précoces sur les capacités de réutilisation du futur système, le fonctionnement de ses composants, ainsi que sur son intégrité, en vue de pouvoir réduire au maximum les coûts de réalisation et de mise sur le marché.

Une argumentation sur l'adoption des techniques de l'orienté objet comme méthodologies cadre pour les systèmes de développement dans l'embarqué a été abordée.

L'aspect « méthodologies de développement » a été consacré aux principaux systèmes de développement de l'embarqué notamment dans le contexte des lignes de produits. Pour montrer en pratique, l'importance de cette dimension en faveur de ce dernier contexte, nous l'avons appuyée par l'exposition de l'éventail des techniques les plus significatives qui l'ont adopté. C'est au vue de cette sollicitude croissante des systèmes embarqués quant aux avantages de la discipline ligne de produits en matière de développement, que nous avons mené avec motivation les travaux relatifs à notre thèse.

Première Partie

État de l'Art

Chapitre II

Modélisations en Lignes de Produits Logiciels

2.1 Introduction

L'objectif de ce chapitre est de situer nos contributions par rapport aux modélisations et techniques existantes dans le contexte des lignes de produits logiciels.

Pour couvrir convenablement cet état de l'art comparatif, nous l'avons organisé en deux grandes sections :

- La première s'intéresse à l'ingénierie des lignes de produits logiciels avec ses principaux avantages et activités.
- La deuxième quant à elle , présente l'importance de l'approche Multi-perspectives des lignes de produits à travers la description des approches de modélisation les plus congrues avec les aspects traités dans notre thèse. Ces approches sont regroupées en fonction de l'aspect abordé.

Nous avons également situé nos contributions par rapport aux mécanismes et concepts introduits dans les travaux connexes.

La Ligne de Produits offre une représentation unique de l'ensemble de systèmes logiciels conceptuellement similaires qui partagent des caractéristiques communes -commonalité- et pourvoient des exigences particulières -variabilité- d'un domaine spécifique [90],[91],[114].

A partir des deux concepts abordés - commonalité et variabilité-, une nouvelle ingénierie a vu le jour, c'est celle des lignes de produits.

Cette dernière inclut 3 activités primordiales selon [9],[85],[86],[114], à savoir :

- L'identification de la variabilité ;
- La dérivation de produits ;
- La gestion des contraintes.

Chacune de ces activités a un rôle essentiel dans le processus de développement d'un produit.

2.2 La ligne de produit pilier de la stratégie de réutilisation

Dans cette partie, nous allons aborder en premier les définitions les plus coutantes de la ligne de produits , pour laquelle , nous en donnerons les principaux apports et le cadre de développement.

2.2.1 Définitions de la Ligne de Produits Logiciels

Une première définition de la ligne de produits logiciels la présente comme « *Un ensemble d'applications développées à partir d'un groupe de caractéristiques communes (à toutes ces applications)- ou Commonalité - et répondant en même temps à des besoins spécifiques - ou Variabilité - dans un domaine particulier »* [9].

Ainsi, le concept de variabilité est utilisé pour regrouper les caractéristiques qui différencient les produits de la même famille.

Une autre définition plus récente liée au marché est celle de [114] que nous présentons telle qu'elle a été publiée : « *la ligne de produit est un ensemble de systèmes intensifs de logiciels qui partagent un ensemble de caractéristiques communes qui satisfont des besoins spécifiques d'un segment particulier ou d'une mission du marché et qui sont développés à partir d'un ensemble commun d'Assets basiques selon un cheminement prédéfini* ».

2.2.2 Apports de la stratégie ligne de produits

Dans cette partie de la thèse, nous allons aborder les plus importants apports de la stratégie lignes de produits à savoir : la réutilisation de masse, la vision multi-systèmes et le paramétrage à grande échelle.

2.2.2.1 La réutilisation de masse

La technique de réutilisation part du principe suivant :'il vaut mieux construire à partir d'un potentiel existant qu'à partir de rien'. Le défi en est d'arriver à le réaliser avec des efforts et des coûts nettement inférieurs à la construction à partir de rien [9].

C'est pour cette raison que la technique de réutilisation pour le développement de logiciels est utilisée dans le contexte des lignes de produits.

Cependant une adaptation de la technique de réutilisation a été nécessaire compte tenu du fait que ce procédé a précédé l'avènement du paradigme ligne de produits logiciels.

Les différentes librairies de codes issues de la réutilisation en sont la meilleure preuve [22].

Cependant, en dehors des lignes de produits, la complexité dans le développement d'un produit réside dans l'identification des composants requis parmi ceux stockés dans les différentes librairies puis une fois détectés à pouvoir les composer ensemble pour créer une nouvelle application [22].

Pour les lignes de produits logiciels, cette technique est utilisée à grande échelle pour la dérivation de produits. Elle permet de réutiliser les composants déjà construits et testés pour dériver d'autres produits. Ces composants peuvent être des documents de spécification, des parties de codes, des modèles….etc.

L'apport étant la capacité de diminuer l'effort de conception, les coûts de développement et l'augmentation de la testabilité et de la qualité des produits.

2.2.2.2 La construction de familles de systèmes logiciels au lieu d'un système unique

La ligne de produits logiciels vise à créer une famille de produits et non un produit unique quel que soit son type de construction.

Avec la réutilisation, la ligne de produit logiciels permet de construire facilement de nouveaux produits à partir d'entités et éléments en entrée ou stockés [9].

Ainsi tous les artefacts (exigences, besoins, composants ….) sont à priori analysés, conçus et modélisés pour être utilisés et/ou réutilisés dans la production d'autres logiciels de la ligne.

2.2.2.3 La construction des produits par le paramétrage à grande échelle

Le paramétrage à grande échelle est une autre singularité des lignes de produits logiciels.

Cette technique permet de configurer la structure d'un produit selon les besoins des parties prenantes c.à.d. des configurateurs potentiels (client, développeur, concepteur..)[9],[85],[86].

Ce procédé représente une forme d'adaptation du produit aux préoccupations de l'utilisateur.

Cette configuration des structures de produits utilise les modèles de la ligne qui renferment comme définis précédemment de la commonalité et de la variabilité. Ces caractéristiques et/ou ces composants sont soient communs et doivent obligatoirement figurer dans la structure de tout produit, soient variables et figurer dans les produits s'ils ont été sélectionnés par les parties prenantes.

La variabilité s'exprime dans la ligne de produits selon trois types [9]:

- Elle est optionnelle si l'élément n'a que deux possibilités, soit figurer soit être absent de la structure d'un produit. C'est ce qu'on appelle l'aspect existentiel ou présentiel.
- Elle peut être également Alternative , si , l'élément variable offre au moins , un choix au configurateur parmi un ensemble d'autres choix (aspect alternatif inclusif ou exclusif).
- Si le choix de l'alternative est introduit à travers un paramètre, nous dirons que le type de la variabilité est paramétré.

Ainsi par la technique du paramétrage associée aux avantages apportés par la composition biforme des produits (contiennent de la commonalité et de la variabilité) , l'ingénierie des lignes de produits a permis de consolider le développement à grande échelle de systèmes logiciels selon les préoccupations des utilisateurs.

Grâce à ces trois apports, l'expérience a démontré le succès de la stratégie ligne de produits dans le développement des systèmes logiciels notamment par la réduction de l'effort , des couts de développement et l'amélioration constante de la qualité des produits [9].

2.2.3 Le cadre de développement de l'ingénierie des lignes de produits logiciels

Le principal objectif d'une Ingénierie LdP est de mettre les mécanismes et les concepts nécessaires pour pouvoir générer des applications (produits) dont les composants peuvent être réutilisables dans les structures d'autres produits.

Cette ingénierie modélise la ligne selon deux niveaux [90],[91] : le niveau domaine ou ingénierie du domaine dans lequel sont construits les composants basiques réutilisables (Assets) des produits de la ligne, et le niveau application qui permet de construire des produits à partir de ces Assets (figure 2.1).

Ainsi, l'objectif du premier niveau d'ingénierie est le développement pour la réutilisation et celui du second est le développement par la réutilisation.

L'unité de construction étant l'Asset. Ce dernier est donc l'élément de base de la réutilisation qui permet de construire tout produit de la ligne. Il peut être un document de spécification, un code de logiciel, un modèle, un ensemble d'exigences … .

2.2.3.1 Le développement pour la réutilisation

Cette étape de développement est réalisée selon trois activités successives: l'analyse du domaine, la conception du domaine et l'implantation du domaine [9],[90],[91].

- L'identification et la modélisation de la commonalité et de même la variabilité de la ligne sont réalisées lors de la première activité.
- La conception du domaine . Cette activité , cherche à mettre en œuvre une architecture logicielle générique de référence de la ligne de produits à partir de laquelle , l'architecture de toute application en est dérivée.Cette dernière comporte les composants, les connecteurs et les contraintes de dérivation des produits.

Il est évident que la commonalité et la variabilité comme identifiées dans la partie analyse devront être spécifiées expressément dans cette architecture.

- L'activité d'implantation . L'activité permet l'implantation de l'architecture sous forme de Composants réutilisables du domaine .Ce sont ces composants qui seront réutilisés dans le niveau de l'ingénierie d'application pour la construction de tout produit de la ligne.

2.2.3.2 Le développement par la réutilisation

Le développement par la réutilisation ou l'ingénierie de l'application utilise les résultats des trois activités de l'ingénierie du domaine :

- Les modèles de caractéristiques ;
- L'architecture générique ; et
- Les composants.

pour la dérivation d'un produit spécifique (ou nouvelle application) de la ligne [9],[90],[91].

Il est à noter qu'à chaque activité au niveau du domaine correspond une activité au niveau de l'application.

Tout produit de la ligne renferme aussi bien de la variabilité à travers les caractéristiques /composants variables que de la commonalité à travers les caractéristiques/composants communs.

La figure 2.1 selon [7],[8],[9] montre l'organisation en deux niveaux de l'ingénierie des Lignes de produits logiciels.

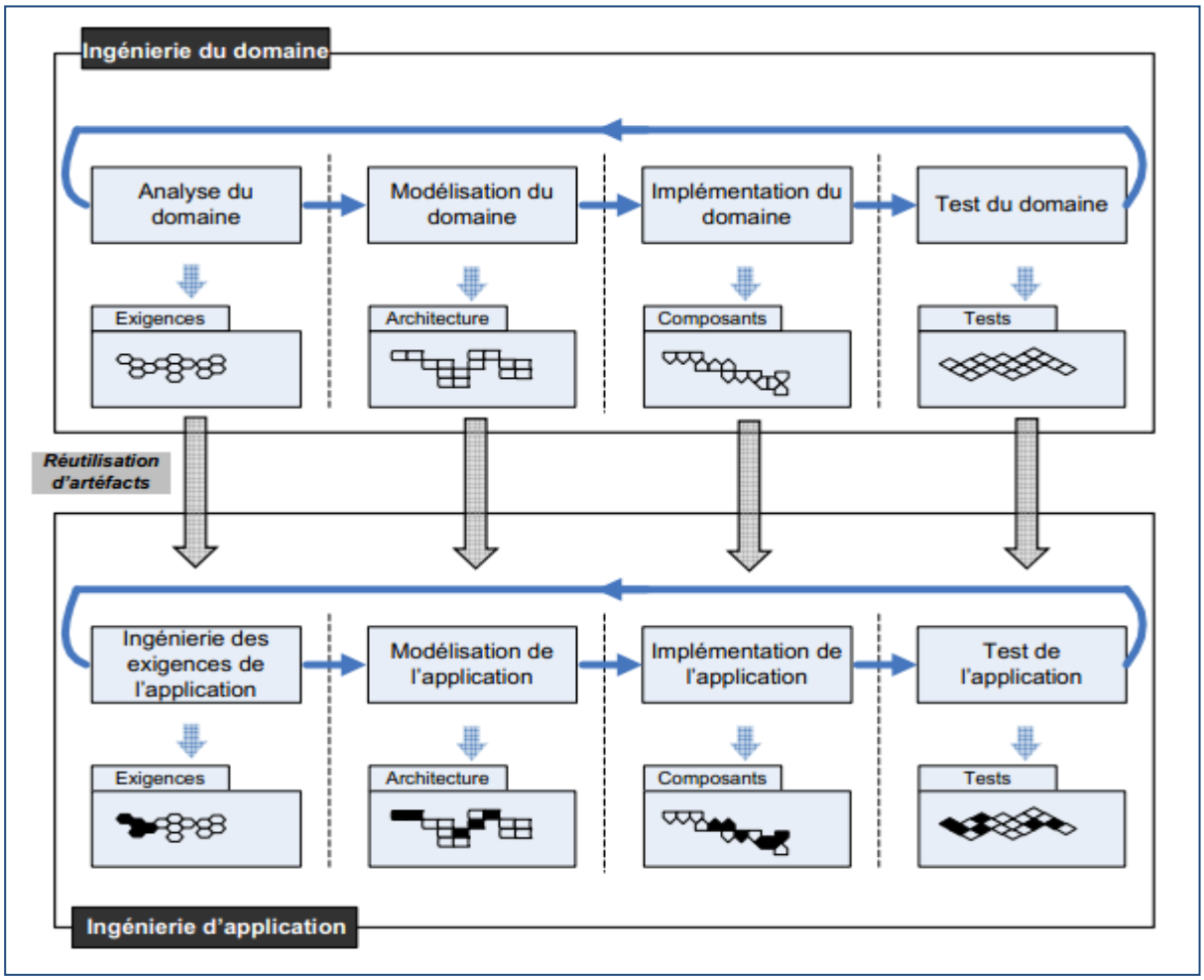


Figure 2.1 Les deux niveaux de l'ingénierie des lignes de produits logiciels

2.2.3.3 L'importance de la Variabilité Logicielle dans l'Ingénierie des Lignes de Produits

Plusieurs aspects dévoilent l'importance de la variabilité logicielle à savoir :

a. L'importance de l'identification et de la gestion de la variabilité.

La commonalité et la variabilité sont les concepts clés dans les lignes de produits logiciels. C'est pour cette raison que les activités de l'ingénierie des Lignes de produits nécessitent davantage d'efforts pour gérer tant les aspects liés à la variabilité que ceux liés à la commonalité.

En effet les éléments communs de la ligne entrent dans la construction des produits selon l'état inchangé de leur identification alors que les éléments variables nécessitent en plus d'une identification, une gestion effective dans la construction des produits bien avant l'étape de leurs délivrances et leurs installations [16],[85].

b. L'étendue de la variabilité en temps et en espace

Selon les critères de présence des éléments de la ligne dans les produits et leurs facultés à offrir au moins un choix en caractéristiques dans les produits, la variabilité a pu être typée optionnelle, alternative ou alternative Paramétrée.

Par contre l'étendue de la variabilité peut prendre deux aspects changeants ou dimensions qui sont le temps et l'espace [15],[16].

- La dimension « Temps » de la variabilité .

  qui a donné l'appellation « variabilité temporelle » à la variabilité, concerne la variation dans le temps d'un produit logiciel unique d'une version à une autre.

- La dimension « Espace » de la variabilité .

  Cette dimension compte à elle, concerne l'existence d'un artefact variant entre plusieurs produits de la même famille en même temps (c.à.d. sans changement dans le temps).

  Les mêmes composants logiciels sont utilisés dans plusieurs applications , et , la variation concerne principalement des variations de fonctionnalités c.à.d. que les produits diffèrent dans les fonctionnalités qu'ils supportent.

  C'est ce dernier aspect de la variabilité qui a donné l'appellation de « variabilité spatiale » à la variabilité.

  Il est à noter , que cet aspect concerne seulement les paradigmes Multi-systèmes (telles les lignes de produits) et ne pourrait être appliqué aux systèmes uniques.

2.3 Modélisations Multi-aspects et Multi-perspectives des Lignes de produits logiciels

Comme montré dans la section précédente, les trois activités des lignes de produits générant des problématiques complexes, leurs prises en charge effectives ne peuvent être réalisées sans une approche de modélisation multi-aspects.

Cette dernière doit couvrir également le maximum de représentations requises pour cette gestion à travers différentes vues (UML et autres) offertes aux parties prenantes. C'est l'approche Multi-perspectives ou Multi-vues.

Dans ce travail, nous avons traité les modélisations en chalenge par rapport à la discipline des lignes de produits selon une double vision Multi-aspects et Multi-perspectives.

Dans les sections suivantes, nous allons en exposer les problématiques associées et les modélisations utilisées puis nous en dresserons un bilan et situeront nos solutions par rapport à ces dernières ; les deux grands aspects abordés étant le développement et l'évolutivité.

2.3.1 Aspect Développement d'une Ligne de produits

Nous aborderons dans cette section les aspects liés au développement d'une ligne de produits en termes de problématique et de formes de modélisation existantes puis nous dresserons un bilan des approches similaires avant de positionner notre démarche proposée par rapport à ces travaux.

2.3.1.1 Problématique et Modélisations existantes

La haute réutilisabilité et la facilité de dérivation réalisée grâce à la modélisation d'artefacts communs et variables étant les points forts les plus importants des LdP, leur exploitation nécessite la conception de modèles de lignes de produits efficaces.

Bien que la plupart des approches traitant les lignes de produits soient fondées sur la modélisation des caractéristiques, elles se basent toutes sur l'analyse des produits similaires existants en vue de créer davantage de réutilisation.

Cependant, dans le contexte de la construction de lignes de produits, les techniques existantes sont caractérisées par un manque d'outils et de démarches claires d'analyse au regard des différentes perspectives des modèles requises par les parties prenantes.

En littérature, Il n y a pas beaucoup de démarches opérationnelles qui ont traité le développement propre des lignes de produits logiciels. Quelques-unes sont basées sur des modèles, d'autres sur des systèmes de métriques et certaines sur la combinaison des de ces deux concepts.

Nous allons nous focaliser dans ce qui suit sur les plus significatives d'entre elles.

Les approches de [69] et de [70] sont basées sur une méthodologie ascendante pour créer une LdP. Elles proposent une série de métriques liées à la similarité existante entre les produits.

L'évaluation des résultats est réalisée selon un modèle unique ; celui des relations, lequel combiné aux métriques proposées permettent d'apprécier l'efficacité du recours à la construction d'une LdP pour développer des produits similaires.

Ainsi, le principal but de ces travaux est l'estimation de l'opportunité d'organiser des produits similaires en LdP et d'opter pour ce type de développement.

[71] par contre, propose un modèle analogue à celui de [69]. Il est basé sur des systèmes de transition modale.

Comparé à l'approche de [69], deux différences sont observées :

L'approche de [71] est basée exclusivement sur des concepts mathématiques .De plus, elle se focalise sur le comportement de la LdP au lieu du développement de ses modèles.

Pour [72], les buts essentiels se concentrent sur la qualité des produits de la Ligne de produits. Pour réaliser cela, il propose un modèle spécifique de variabilité avec un ensemble de métriques. L'objectif de ces derniers étant l'évaluation de la qualité des produits après leur dérivation.

D'autres approches sont dédiées exclusivement à la mesure des systèmes logiciels orientés objet. Parmi ces travaux, celui de [67] qui est basé sur des interfaces de spécification. L'approche de [74] par contre, emploie le concept de 'Service-Utilisation' pour opérer de telles mesures.

Dans le travail de [75], l'approche cherche à construire et appliquer un modèle de qualité pour analyser une application Web et évaluer les qualités structurelles du logiciel en utilisant ce modèle après sa construction.

Ce dernier est basé sur un ensemble de métriques du logiciel associées à un système de prédiction. L'approche fournit un cadre significatif pour construire graduellement des modèles de qualité mais se concentre sur un système unique au lieu d'une famille de logiciels. Une adaptation au domaine des lignes de produits devient alors nécessaire.

[76] présente une approche en vue de développer une architecture pour une famille de systèmes appartenant à un domaine changeant.

Cette approche est basée sur des modélisations de cas d'utilisation et de caractéristiques.

La structure du produit dans le domaine se présente sous la forme de sous-systèmes de composants dans la vue composant d'UML.

[77] développe un ensemble de métriques dédiées aux assets basiques.

Cet ensemble est structuré en une architecture de LdP avec deux modèles :

- Un modèle de décision (sélection de composants, gestion des dépendances..;
- Un modèle de composants.

[78] propose d'utiliser un cadre automatique autour d'un modèle de transformations pour générer automatiquement tous les produits valides construits à partir des caractéristiques existantes dans le diagramme de caractéristiques d'une LdP.

Pour réaliser cela, il utilise une grammaire graphique avec l'outil AToM3. Cette approche n'utilise pas des métriques. Cependant, elle peut être appliquée pour capturer la configurabilité d'une LdP déjà modélisée selon les caractéristiques.

Le travail de [79] a des objectifs proches de notre approche mais ne s'est pas focalisé sur des métriques de réutilisabilité dans le but de construire un modèle performant pour la LdP. Les métriques proposées estiment l'effort requis pour développer une LdP.

Ces métriques sont également utilisées pour intégrer les caractéristiques potentielles et les fonctionnalités dans la LdP existante.

2.3.1.2 Bilan

[69] et [70] sont des approches basées sur des métriques liées à la similarité entre les produits existants qui utilisent pour la tâche d'évaluation , un seul modèle , « le modèle de relations » entre les produits.

Une technique permet de changer la configuration du produit pour essayer d'optimiser le seuil de similarité.

De plus, le principal but de ces travaux est l'estimation de l'opportunité d'organiser les produits existants en LdP et donc de répondre à la question « faut-il opter ou non pour une stratégie de développement des systèmes logiciels en ligne de produits ?».

[71] propose un modèle de systèmes de transition modale proche de [69] mais formalisée avec des concepts mathématiques.

Ce travail se concentre sur l'aspect comportemental de la LdP et non sur la mise en place d'une architecture efficace pour la ligne.

[72] s'intéresse à la qualité des produits après leurs dérivation.

Il se base sur un modèle de variabilité qu'il a défini avec un ensemble de métriques qui ont pour but d'évaluer cette qualité.

Ainsi, le développement d'une LdP n'est pas la vision primordiale de ce travail en dépit du fait qu'il peut être utilisé pour faciliter cette activité.

Le travail de [67] est basé sur des interfaces de spécification et celui de [74] emploie le concept de 'Service Utilisation'.

De plus, cette dernière approche est dédiée uniquement à la mesure des systèmes orientés objet.

Un autre travail peut être lié à cet aspect, c'est celui de [75] qui s'intéresse à la qualité des applications Web en évaluant leurs qualités en termes de structure.

Malgré l'apport certain de ce travail, il ne peut être appliqué que sur un seul système à la différence de la ligne de produits qui englobe des familles de logiciels.

Cette approche nécessite une adaptation pour pouvoir être appliquée au domaine des lignes de produits.

[76] présente une approche pour développer une architecture pour une famille de systèmes appartenant à un système très changeant (domaine en cours de recherche).

Cette approche est basée sur des modèles à base de cas d'utilisation et de caractéristiques qui propose en résultat, une structure du produit sous la forme d'un sous-système dans seulement une seule vue : la vue composant d'UML. Ce travail est beaucoup plus proche de l'aspect extensibilité que celui du développement d'une LdP.

[77] se base sur des métriques liées aux assets basiques seulement, c.à.d. qu'il suppose disposer déjà d'un modèle fiable de composants pour une LdP en début de processus.

[78] utilise un outil existant à base d'un modèle de transformations pour générer automatiquement tous les produits valides construits à partir des caractéristiques existantes dans le diagramme de caractéristiques.

Cette approche ne génère pas de métriques.

Cependant, elle peut être appliquée pour capter la configurabilité d'une LdP déjà modélisée en caractéristiques. Elle n'est toutefois pas recommandée pour développer un modèle efficace pour les caractéristiques d'une LdP.

En dernier, l'approche de [79] pourtant très significative dans le domaine des LdP, génère certes des métriques mais qui ne sont pas liées directement à la réutilisabilité des parties similaires de produits mais utilise ces métriques pour mesurer l'effort nécessaire à la construction de cette structure.

2.3.1.3 Positionnement de notre démarche

Notre approche est basée sur une méthodologie ascendante. Au lieu de commencer la construction d'une LdP à partir des connaissances du domaine seulement, elle utilise un ensemble prédéfini de produits existants tels les travaux de [69] et [70].

Ces produits peuvent être définis par les parties prenantes eues égard de leurs perspectives en matière de projets.

Cependant, à la différence des approches similaires, le principal but de notre travail est de construire une LdP performante à travers un modèle de caractéristiques optimal tout en offrant un maximum de perspectives aux parties prenantes selon différents critères d'analyse.

Pour réaliser cet objectif, notre approche basée sur un processus, développe en premier un diagramme de caractéristiques à partir d'une modélisation de la variabilité contenue dans des produits similaires existants.

Cette étape préalable définit la représentation clé pour les phases suivantes d'analyse, de configuration, d'évaluation et d'optimisation.

De plus, notre démarche n'est pas basée seulement sur la mesure et l'évaluation de la similarité des structures de produits mais consolide cette activité par un sous-processus préalable d'analyse du modèle de caractéristiques qui génère un éventail de métriques.
Dans ce contexte et en vue de donner plus de crédibilité à ce sous-processus, nous l'avons exécuté selon quatre critères d'analyse et en différentes granularités d'unités d'analyse du produit.
D'autres différences comparées aux approches semblables existantes concernent la possibilité de déterminer à la fois les vues optimales selon le critère d'analyse choisi et les diagrammes de caractéristiques optimaux tout aussi bien que les vues désirées et les diagrammes de caractéristiques selon le critère voulu.

Notre processus grâce à un ensemble d'autres métriques, offre également la capacité pour les parties prenantes d'évaluer les modèles résultants sous forme de vues lors de toutes les étapes ainsi que de capter la configurabilité des produits en entrée avant leur dérivation ; l'objectif commun étant d'atteindre une réutilisabilité optimale du modèle des caractéristiques obtenu à chaque phase.
En adoptant une telle démarche, nous soutenons une facilitation de l'obtention d'un modèle de caractéristiques optimal et fiable pour la ligne de produits.

2.3.2 L'Aspect Evolutivité

Nous avons traité cet aspect à travers une modélisation multi-perspectives liée à l'évolutivité à savoir : l'évolution de la variabilité.

2.3.2.1 Problématique et Modélisations existantes

L'évolution de la variabilité dans la majorité des approches existantes est exprimée à travers l'évolution des points de variation de la ligne de produits [61].Un point de variation identifie un emplacement dans un modèle d'éléments composant la ligne de produits (modèle de caractéristiques, modèle de classes, modèle de composants…) au niveau duquel la variation est exprimée.
Un point de variation peut être de type optionnel ou alternative c.à.d. offrant plusieurs choix. Pour ce dernier type, les alternatives possibles sont appelées 'Variants'. Modéliser la variabilité évolutive a toujours été un défi pour les développeurs en lignes de produits logiciels.
Dans ce contexte, les approches les plus récentes se sont focalisées sur l'aspect architectural en offrant des solutions à base de génération de code et de modèles.
Il est à noter également que ces approches ont cherché des mécanismes pour gérer automatiquement la variabilité évolutive.

Nous les avons scindés en trois groupes :

1. Les approches basées sur l'évolution des Modèles de caractéristiques.

Ce type d'approches comporte deux catégories :

Les approches basées sur l'évolution intra-spatiale telles [50], [65] et les approches basées sur un processus de raffinement de modèle architectural telles [62],[81],[82],[83] et les approches basées sur un système de traçabilité [63],[80],[84],[87].

i. Les approches basées sur l'évolution intra-spatiale.

Selon cette vision, l'un des travaux les plus significatifs est celui de [50].

Dans son approche, le modèle de caractéristiques est la seule cible d'évolution dans l'espace des problèmes.

Des modèles , dans l'espace des problèmes , tels que , les cas d'utilisation , et les modèles d'exigences ne sont pas considérés.

Cependant, ce travail propose cinq types d'évolution pour l'espace des Problèmes :

-'*Duplicate feature'*.

Par Duplicate feature', l'approche copie la caractéristique sélectionnée (caractéristique d'origine) et ajoute le clone (la caractéristique copiée) comme cible de la caractéristique sélectionnée au modèle de caractéristiques.

-'*Insert Feature'* :

Par 'Insert Feature', l'approche permet de créer entièrement une caractéristique (nouvelle caractéristique) dans le modèle de caractéristiques comme élément fils de la caractéristique courante sélectionnée.

-'*Split Feature'*:

'Split Feature' est utilisée pour modéliser une caractéristique par sa scission selon plusieurs issues à grains fins pour lesquelles elle a été originalement dédiée et ce en distribuant sa fonctionnalité à un nombre arbitraire de composants.

-'*Owned Assets'* :

Les assets résultant de l'évolution 'Split Feature' sont gérés par la fonction 'Owned Assets' dans le modèle.

-'*Remove Feature'* :

'Remove Feature' peut être appliquée pour supprimer la caractéristique sélectionnée (la caractéristique originale) du modèle de caractéristiques.

-'*Remove Feature'* associée à *'Owned Assets'*

'Remove Feature' associée à 'Owned Assets' représentent un type d'évolution complexe en permettant de supprimer les caractéristiques sélectionnées (Original-

Feature) de l'espace problème et aussi de supprimer tous les assets de L'espace solution qui sont utilisés exclusivement par cette caractéristique mais pas les autres.

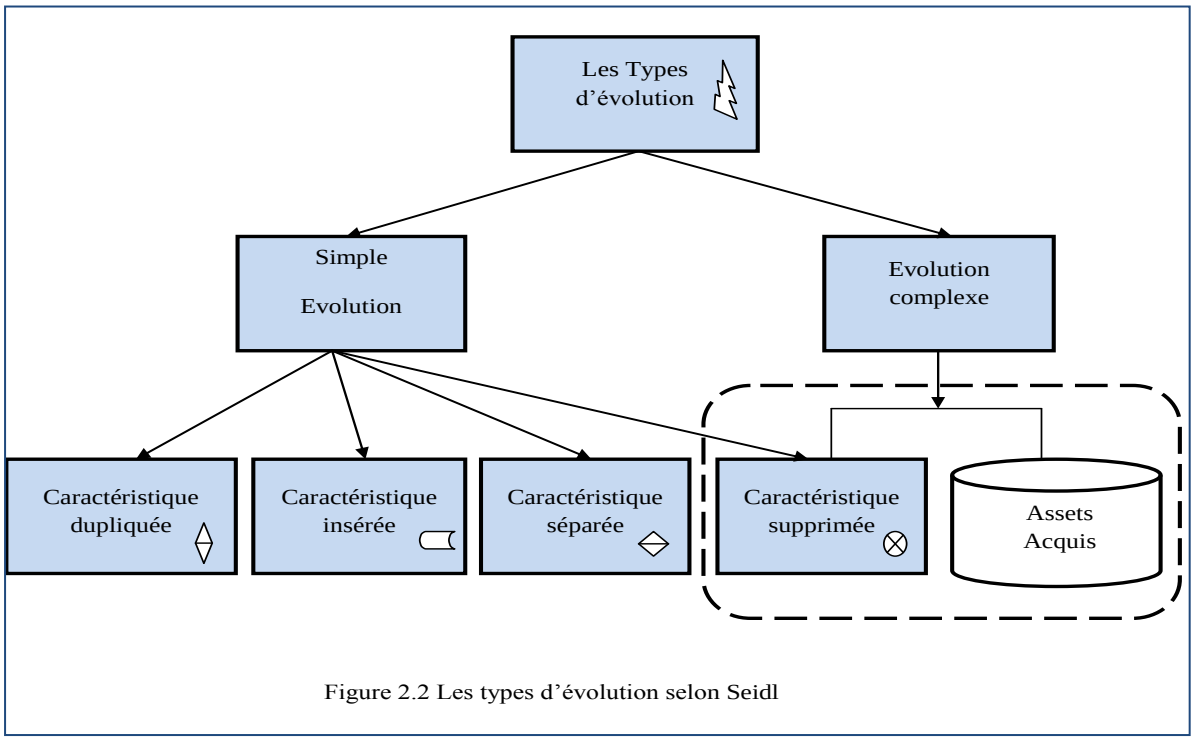


Figure 2.2 Les types d'évolution selon Seidl

Le grand avantage de cette approche concerne l'évolution intra-spatiale qui n'affecte pas le mapping de la caractéristique .Cependant, les contraintes de transition et les dépendances entre les caractéristiques ne sont pas prises en compte.

De plus, l'approche ne supporte pas la multi-instanciation dans la même classe. Le nombre élevé de types d'évolution et les limitations citées précédemment ne facilitent pas la gestion de l'évolution et peuvent mener à des modèles incompréhensibles (problème de visibilité).

ii. Les approches basées sur un Processus de raffinement

Selon cette vision, l'une des approches les plus significatives est celle de [62]. Ses Co-auteurs , ont présenté , une théorie générale pour les descriptions des évolutions de la variabilité selon des raffinements des modèles architecturaux de la ligne de produits et notamment lorsque ces évolutions altèrent la sémantique de la ligne.

Cependant , ils n'ont pas pris en considération la sémantique des Assets individuels de l'espace solution , et , ils n'offrent pas de réelles opérations , pour la prise en charge de l'évolution.

iii. Les approches basées sur un système de traçabilité .Selon cette technique, l'un des travaux les plus significatifs est celui de [63] qui a présenté une approche qui permet de tracer des liens entre les artefacts de la ligne de produits à l'aide de quatre dimensions :

-La dimension 'variabilité'.

La dimension 'variabilité' relie l'espace problème aux artefacts de l'espace solution (et par la même relie ces mêmes artefacts dans l'espace problème).

-La dimension 'temps'.

La dimension 'temps' décrit comment un artefact change à travers son évolution .Il peut être utilisé pour récupérer les modifications apportées par l'évolution.

-La dimension 'raffinement'.

La dimension 'raffinement' permet de capturer les relations entre les artefacts de différents niveaux d'abstraction.

-La dimension 'similarité'.

La dimension 'similarité' capture les relations entre les artefacts de même niveau d'abstraction.

Cependant, la procédure qui supporte l'évolution demeure relativement complexe car nécessitant la gestion de plusieurs dimensions de la variabilité liées à plusieurs aspects.

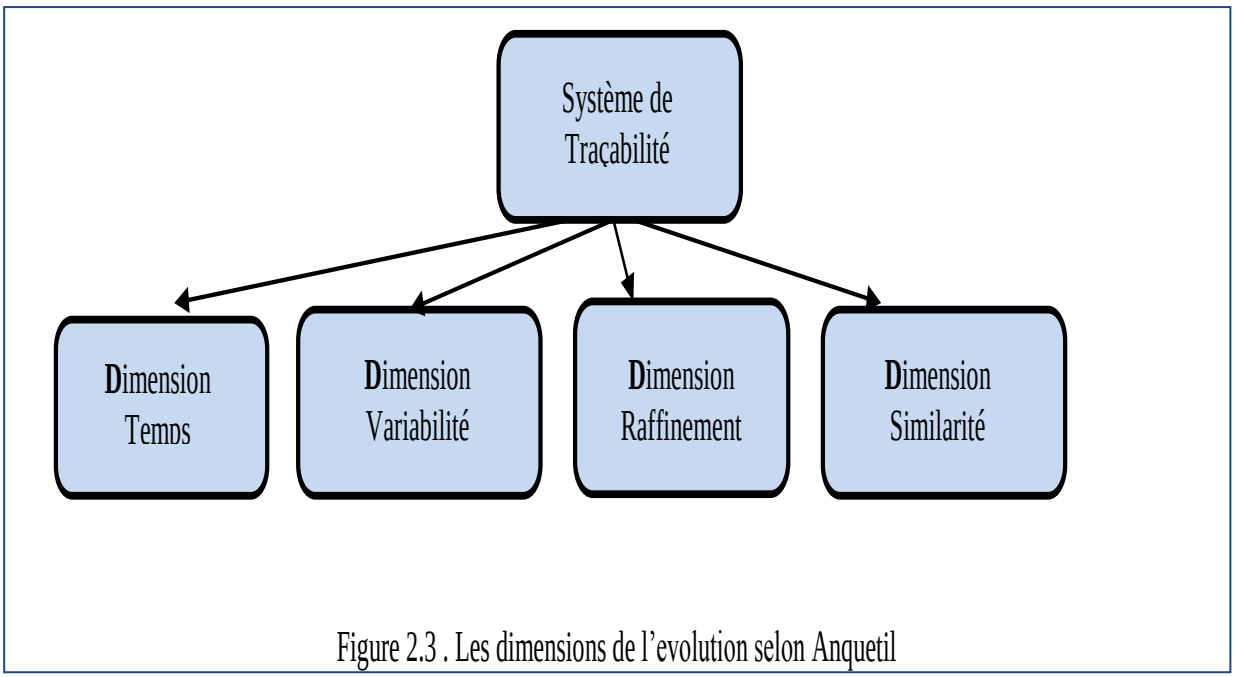


Figure 2.3 . Les dimensions de l'evolution selon Anquetil

2. Les approches qui ont utilisé le rôle UML pour modéliser l'évolution dans le diagramme de classes des systèmes uniques.

Compte tenu du fait que notre approche soit basée sur une adaptation du rôle UML pour gérer l'évolution de la variabilité dans les lignes de produits, nous allons présenter dans ce qui suit les principales approches qui ont utilisé le rôle UML pour modéliser l'évolution dans le diagramme de classes pour un système unique à savoir :

Les approches caractérisées par l'usage des modèles orientés objet, les approches caractérisées par l'usage d'une extension du rôle UML [60], Les approches caractérisées par

l'usage d'une implémentation dans un SGBD relationnel ou orienté objet, les approches caractérisées par l'usage d'une extension canonique du développement orienté objet [66], les approches caractérisées par l'usage d'un développement orienté rôle [51],[64].

- Usage des modèles orientés objet :,

[51] a essayé d'introduire les rôles pour gérer les objets dynamiques dans le diagramme de classes d'UML en présentant des modèles appropriés de données orientés objet qui peuvent également résoudre les problèmes liés à des aspects d'héritage multiples.

- Usage d'une extension du rôle UML :

[49], [52], [53] ont proposé une extension graduelle d'UML avec une nouvelle association rôle. De nouvelles méta-classes et règles OCL ont été ajoutées aux Méta-modèle UML pour capturer la sémantique des rôles.

- Usage d'une implémentation dans un SGBD relationnel ou orienté objet :

[54] a discuté le concept de rôle qui peut supporter l'aspect dynamique de l'objet comme une alternative à une modélisation conceptuelle et définit une structure de données - un sous-objet d'un objet - pour être implémenté dans un SGBD orienté objet ou relationnel.

- Usage d'une extension canonique du développement orienté objet:

[55] a présenté une extension canonique du développement orienté objet à travers l'introduction de rôles, dans laquelle il distingue à partir d'un point de vue syntaxique entre les classes et les rôles .Dans cette approche, la classe contient les rôles et leurs dépendances.

- Usage d'un développement orienté rôle :

[56], [57], [58] utilisent une programmation orientée rôle (RBP) pour représenter l'aspect dynamique des objets.

Avant d'achever cette section, il est important de signaler l'étude détaillée sur le rôle UML et ses problématiques qui a été réalisée par Steinmann [59] et dont se sont inspirés la plupart des travaux cités.

Cependant et en dépit des contributions apportées par ces approches, elles n'ont concerné que les systèmes logiciels uniques.

2.3.2.2 Bilan

Le bilan que nous avons pu dresser se résume dans les points suivants :

1. Pour les approches ayant discuté la gestion de l'évolution de la variabilité.

Pour les approches basées sur un typage de la variabilité selon son évolution, le grand avantage concerne l'évolution intra-spatiale qui n'affecte pas le mapping de la caractéristique.

Cependant, les contraintes de transition et les dépendances entre les caractéristiques ne sont pas prises en compte.

De plus, elles ne supportent pas la multi-instanciation de la même classe.

Ces limitations ne facilitent pas la gestion de l'évolution et peuvent mener à des modèles difficiles de lecture.

i. **Pour la gestion de l'évolution basée sur un système de traçabilité**,

La procédure qui supporte l'évolution demeure relativement complexe car nécessitant la gestion de plusieurs dimensions de la variabilité liées à plusieurs aspects.

ii. **Pour la gestion de la variabilité à travers un processus de raffinement**,

Les approches n'ont pas pris en considération la sémantique des assets individuels de l'espace solution et n'offrent pas de réelles opérations pour la prise en charge de l'évolution.

2. Pour les approches ayant favorisé le contrôle de l'évolution des systèmes uniques à base du rôle UML.

L'évolution des objets à travers les variantes d'objets est spécifiée d'une manière ambigüe dans le diagramme de classes.

A ce titre, la représentation de l'évolution des objets est laissée à la charge du concepteur et/ou de l'architecte système. Ces intervenants sur la ligne de produits ou sur l'évolution des rôles utilisent chacun les techniques et les méthodes qu'ils jugent appropriées pour contrôler cette évolution dans le diagramme de classes.

3. Constat commun

Malgré les contributions apportées par ces deux catégories d'approches, aucune d'entre elle n'a discuté la possibilité de représenter la variabilité par une adaptation du rôle UML aux lignes de produits.

2.3.2.3 Positionnement de notre démarche

Notre approche a cherché à tirer avantage des deux catégories d'approches. Ce compromis se traduit à travers des mécanismes d'expression de la variabilité à travers l'utilisation du concept de rôle UML revu et adapté au contexte des lignes de produits.

Une caractéristique variable peut évoluer simplement en acquérant ou perdant différents rôles à travers son cycle de vie sans changer et d'une façon plus complexe en acquérant des rôles tout en gardant les anciens (rôles déjà acquis) ou en acquérant des rôles tout en perdant les anciens (acquis auparavant).

Face à l'inadaptation du modèle de caractéristiques à représenter l'aspect évolutif de la variabilité, nous avons intronisé un nouveau modèle de représentation que nous avons appelé modèle d'évolution représenté sous forme de diagramme.

Le processus de modélisation mappe le modèle des caractéristiques en diagramme de classes dans une première étape puis dans une seconde, scinde ce dernier en partitions variables et enfin fait correspondre à chaque partition son volet évolution à travers une machine à états correspondante.

Nous avons proposé l'expression de la variabilité au niveau des variants relevant des points variables par une nouvelle association désignée « *role* » .

Dans le modèle, les points de variation sont représentés sous forme de super-classes (classes de f-roles) et leurs variants sous forme de sous-classes roles (classes filles de f-roles).

A chaque super-classe correspond une machine à état-transitions qui représente l'évolution de la variabilité à travers les variant-roles (f-roles) de ce point de variation.

Ces classes de f-roles sous certaines contraintes peuvent aussi évoluer pour jouer d'autres rôles.

La multi-instanciation de la même classe est aussi supportée par notre modèle. L'association « *role* »permet également aux classes Variant-roles d'hériter leurs super-classes.

Pour compléter le niveau d'information de notre modèle, un listing des contraintes d'acquisition des rôles par les instances de la super-classe ainsi que les contraintes d'évolution des rôles peut lui être joint.

Ces deux types de contraintes peuvent être exprimés sous la forme de prédicats dans le langage formel OCL.

Nous avons enfin intégré tous ces concepts dans les Méta-modèles d'UML.

2.4 Conclusion

Nous avons présenté dans ce chapitre deux aspects liés fortement aux points forts des Lignes de produits que nous avons eus à traiter à savoir :

Le développement et l'évolutivité.

Nos solutions dans ce contexte permettent une vision Multi-perspectives des lignes de produits logiciels.

Cet axe de la recherche a été abordé par la présentation des différentes modélisations existantes de la ligne selon l'aspect traité.

Nous avons aussi situé nos contributions par rapport aux travaux semblables existants.

Pour bien mettre en valeur nos apports, nous avons structuré chaque aspect traité en trois parties :

- La première arbore les problématiques de l'aspect et en cite les principales contributions liées ;
- La deuxième dresse un bilan des approches existantes en termes de points forts et insuffisances ; et
- La troisième positionne notre démarche en termes de comparaison et d'apports par rapport aux travaux connexes.

Nous pensons qu'avec une telle démarche, nous sommes parvenu à dresser un état de l'art comparatif des travaux existants les plus significatifs dans le contexte des aspects traités dans le contexte des systèmes en lignes de produits logiciels et à bien situer les apports de notre thèse par rapport à ces travaux .

Deuxième Partie

Contributions

Chapitre III

Un processus basé sur les métriques pour le développement d'une ligne de produits logiciels

3.1 Introduction

Le concept d'ingénierie des lignes de produits (LdP) cherche à rationaliser le développement des familles de produits logiciels similaires. Le résultat principal d'un tel paradigme d'ingénierie est de promouvoir la réutilisation dans tout le processus de développement en vue de réduire sa durée de réalisation, son cout et l'effort nécessaire à son déroulement.

Durant les dernières années, les lignes de Produits sont de plus en plus utilisées dans l'industrie de l'embarqué tout autant que dans le domaine académique [6],[80],[72].

Selon le paradigme « Ligne de Produits », le développement de nouveaux produits logiciels peut suivre aussi bien une méthodologie dévalente qu'une méthodologie ascensionnelle et ce en dérivant un produit à partir respectivement des modèles de la ligne ou par factorisation vers la LdP de l'architecture des applications similaires déjà existantes [80].

Malgré le fait que ce dernier type de méthodologie soit le plus adopté pour développer une ligne de produits en industrie et notamment dans celle de l'embarqué , il n'existe pas dans la littérature une méthode opérationnelle testée et approuvée pour développer et évaluer des modèles de lignes de produits à partir de parties appropriées ou prédéfinies de logiciels existants ou même de sa totalité [69],[70],[72],[116].

Dans ce chapitre, nous décrivons une démarche sous forme d'un processus de développement d'une LdP basé sur un jeu de métriques générées à partir d'un ensemble de modèles. Le résultat est un modèle optimal de caractéristiques de la ligne sous la forme d'un diagramme.

Dans un tel processus, les entrées sont représentées par un ensemble prédéfini de produits existants qui constituent la structure primaire du modèle de la future LdP sous la forme d'un diagramme de caractéristiques initial. Ce dernier est construit à base d'un modèle de variabilité. Une première capture de la configurabilité de ces produits sous forme de vues est également réalisée. Dans le but d'analyser cette structure, un modèle d'analyse selon des critères de différentes granularités est utilisé. Les résultats sont un ensemble de produits potentiels sous forme de vues d'analyse ou analytiques.

Pour capturer la réutilisabilité, la commonalité et la différentiabilité contenues dans les produits potentiels analysés, les vues analytiques sont transformées par un modèle de Mapping en une forme plus appropriée que nous avons appelée vues de corrélation.

Ce faisant, la mesure de la similarité (réutilisabilité plus commonalité) et la différentiabilité des produits potentiels selon les vues de corrélation est effectuée.

A partir de ces vues et de ces métriques, nous établissons des évaluations des configurations obtenues en vue de déterminer les vues optimales selon différents formats (vue et diagramme). Ces dernières sont utilisées pour déterminer le diagramme optimal des

caractéristiques de la ligne pour chaque critère d'analyse ainsi que le diagramme des caractéristiques global de la ligne en fin de processus.

Nous allons donner dans la section suivante un aperçu sur le déroulement de ce processus et les métriques qu'il génère.

3.2 Aperçu de notre Processus

En plus du modèle de variabilité, les étapes clé et les artefacts de notre processus de développement se basent sur six autres modèles [73]: un modèle d'analyse , deux modèles de Mapping , un modèle de corrélation , un modèle d'évaluation et un modèle d'intégration.

Typiquement, le processus commence par la modélisation de la variabilité qui produit un diagramme initial des caractéristiques à partir d'un ensemble de produits similaires existants, prédéfini par les parties prenantes de la future ligne de produits.

La seconde étape se focalise sur l'analyse à partir du diagramme initial des caractéristiques considéré cette fois-ci comme un graphe, de la variabilité contenue dans la structure interne de chaque produit prédéfini lors de la première phase mais en accordance avec les différents critères d'analyse au nombre de quatre (l'utilité, la profondeur, la couverture et la scannérisation).

Les résultats sont un ensemble de configurations initiales de produits qui sont représentées dans une vue appropriée en forme de table selon l'unité d'analyse et les métriques liées à ce critère. Nous l'avons appelée 'la vue initiale d'analyse selon le critère' ou (VAIC) dans le diagramme d'activités de notre processus de développement de la figure 3.1.

Dans la troisième étape, un modèle de Mapping permet de transformer la vue initiale d'analyse selon le critère en une représentation équivalente dans le modèle de corrélation. Nous l'avons appelée 'la vue Initiale de corrélation selon le critère ' ou VCIC dans le diagramme d'activités de notre processus.

Le processus se concentre ensuite sur l'identification du cadre de variation ou ensemble de variation (EV dans le processus de développement) et de ses contraintes de configuration (CC dans le diagramme d'activités de notre processus de développement), qui permettent de scinder le cadre de variation en des sous-ensembles ou sous-cadres de variation (SEV dans le diagramme d'activités de notre processus).

L'activité d'identification détermine également les configurations de variation (CoV dans le diagramme d'activités de notre processus) requises pour chaque sous-ensemble de variation selon les contraintes de configuration.

Les configurations de variation donnent une vue potentielle sur l'évolution de chaque configuration initiale du produit selon les contraintes de configuration dans le sous-ensemble des variations approprié .Nous avons noté cette forme de variation du produit (i): (ΔSP_i).

En appliquant l'ensemble des cadres de variation à une vue d'analyse initiale selon un critère donné, nous obtenons une nouvelle Vue d'Analyse selon ce même Critère ou (VAC dans le diagramme d'activités de notre processus).

Cette vue d'analyse prend la même forme de représentation que la vue initiale d'analyse selon le même critère mais avec la prise en compte des changements apportés par les cadres de variation. Cette tâche est réalisée par l'activité de configuration.

Le Mapping «Modèle d'analyse – Modèle de corrélation (MA-MC dans le diagramme d'activités de notre processus)» permet de transformer chaque vue d'analyse selon ce même critère en une Vue de Corrélation selon ce Critère (VCC dans le diagramme d'activités de notre processus).

A l'aide d'un modèle d'évaluation, le processus par la génération d'un ensemble de métriques appropriées, permet d'évaluer d'une part, la corrélation entre les parties internes des produits dans le modèle de corrélation et de l'autre, les configurations de produits en termes de similarité, commonalité, réutilisabilité et différentiabilité.

Dès lors, les mesures de la similarité (réutilisabilité plus commonalité) et de la différentiabilité des produits selon la vue de corrélation peuvent être réalisées. Les métriques associées à la réutilisabilité selon chaque critère seront utilisées pour déterminer les Vues de Corrélation Optimales (et / ou désirées par les parties prenantes) selon le Critère ou (VCCO

Table 3.1. Exemple de structure interne d'un produit

f_1	f_2	f_3	f_4	f_{11}	f_{12}	f_{31}	f_{32}	f_{41}	f_{42}	f_{321}
1	1	1	1	1	0	0	1	1	0	1

Table 3.2. Vue Utilité (UV) de la LdP (SCAM)

UV	f_1	f_2	f_3	f_4	f_{11}	f_{12}	f_{31}	f_{32}	f_{41}	f_{42}	f_{321}
SP_1	1	1	1	1	1	0	0	1	1	0	1
SP_2	1	0	1	1	1	0	1	0	0	1	0
SP_3	1	1	1	1	0	1	0	1	1	1	1

dans le diagramme des activités de notre processus).

Cette tâche est effectuée par la sous-activité d'optimisation. Une correspondance de ces vues optimales de corrélation dans le modèle d'analyse permettra de déterminer les vues Analytiques Optimales selon le Critère ou (VACO dans le diagramme d'activités de notre processus).

En appliquant un deuxième modèle de Mapping , cette fois entre le modèle d'analyse et le modèle des caractéristiques (MA-MCa dans le diagramme d'activités de notre processus de

développement) ,une vue équivalente à la vue d'analyse optimale selon le critère est établie sous forme de diagramme des caractéristiques optimal selon ce même critère ou (MCCO dans le diagramme d'activités de notre processus). En fin de processus, un

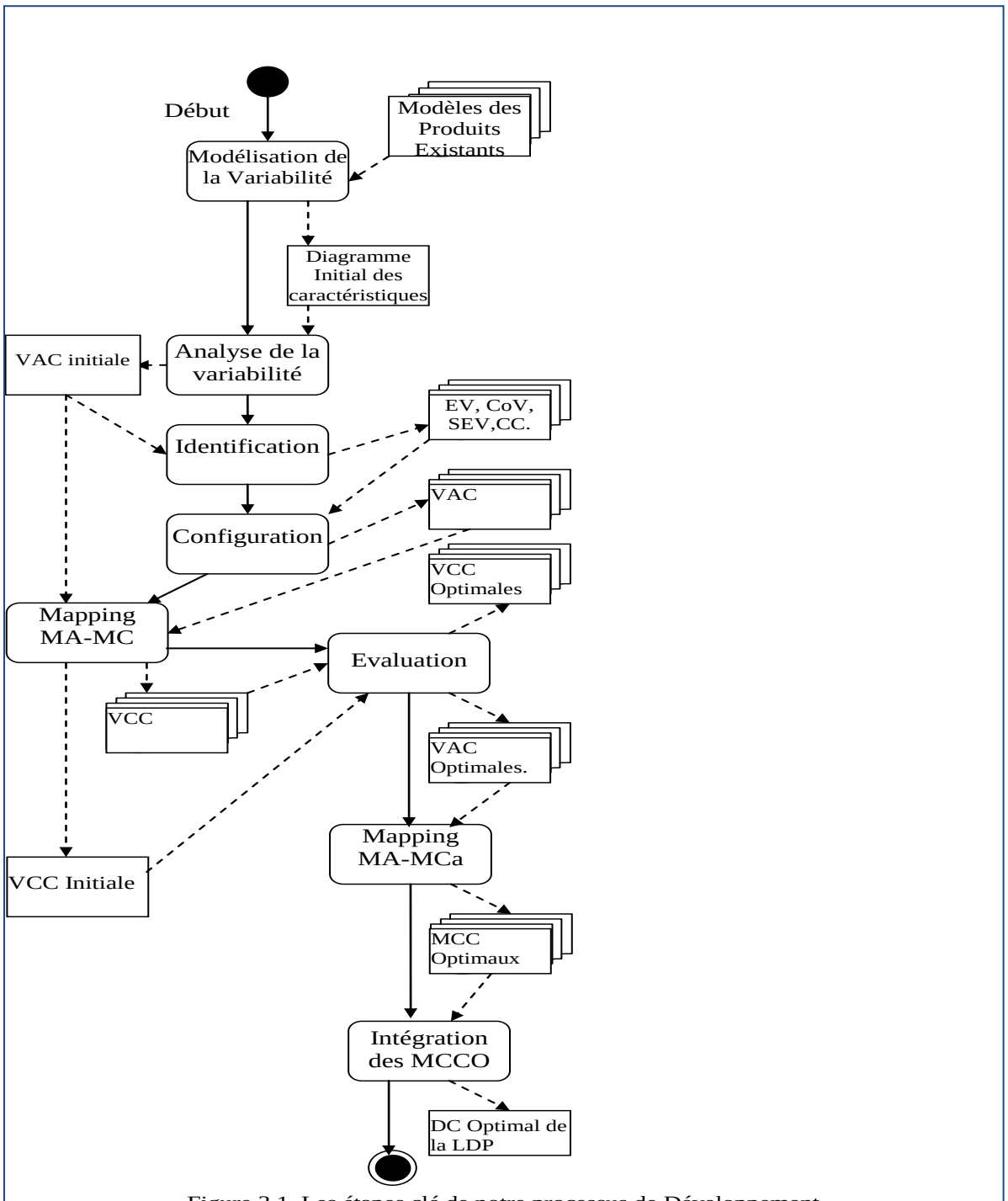


Figure 3.1. Les étapes clé de notre processus de Développement

diagramme optimal des caractéristiques de la LdP peut être obtenu par l'intégration de tous les diagrammes optimaux selon le critère. Pour réaliser cette opération, nous avons opté pour l'usage d'un modèle outillé de composition des diagrammes de caractéristiques, celui conçu par [116].

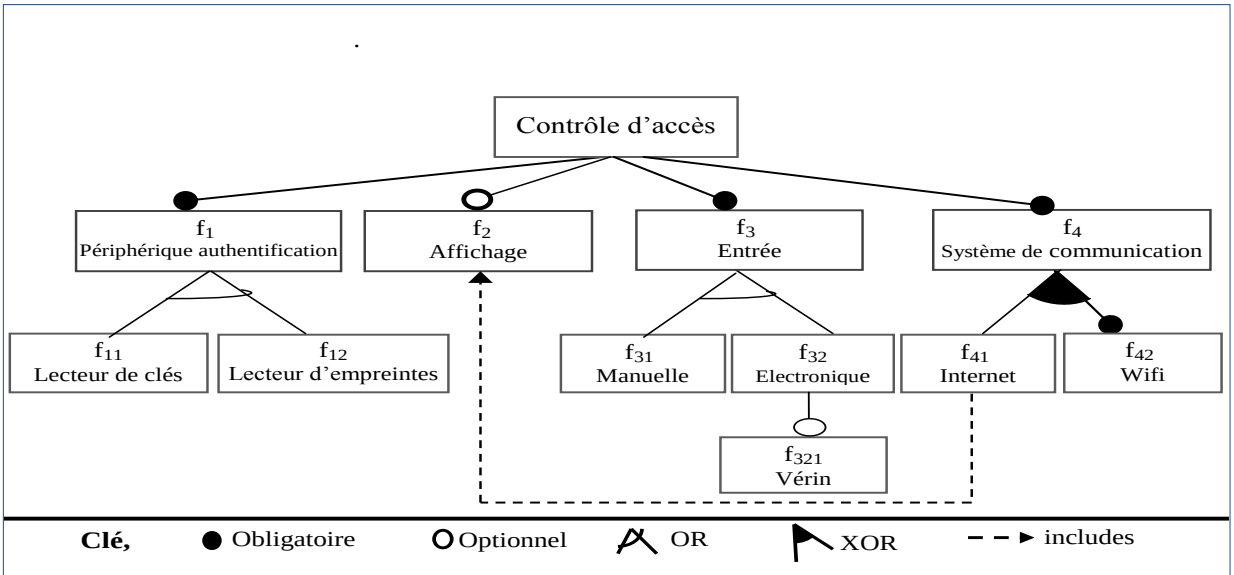


Figure 3.2. Le diagramme des caractéristiques Initial de la LdP (SCAM)

Dans le but de présenter les performances basiques de notre approche, nous avons focalisé nos illustrations sur le domaine des maisons intelligentes abordé par [115] et spécialement le système qui contrôle l'accès à la maison que nous avons noté (SCAM) dont nous avons adapté le diagramme des caractéristiques à notre travail de thèse (figure 3.2).

En vue de décrire la variabilité dans la partie (SCAM) de la ligne de Produits, nous considérons que ce système est conçu pour être utilisé avec différents périphériques d'authentification à savoir : 'le lecteur de clés' et le 'lecteur d'empreintes' et fournit deux alternatives de verrouillage de la porte d'accès : une 'manuelle' et l'autre 'électronique' .Pour

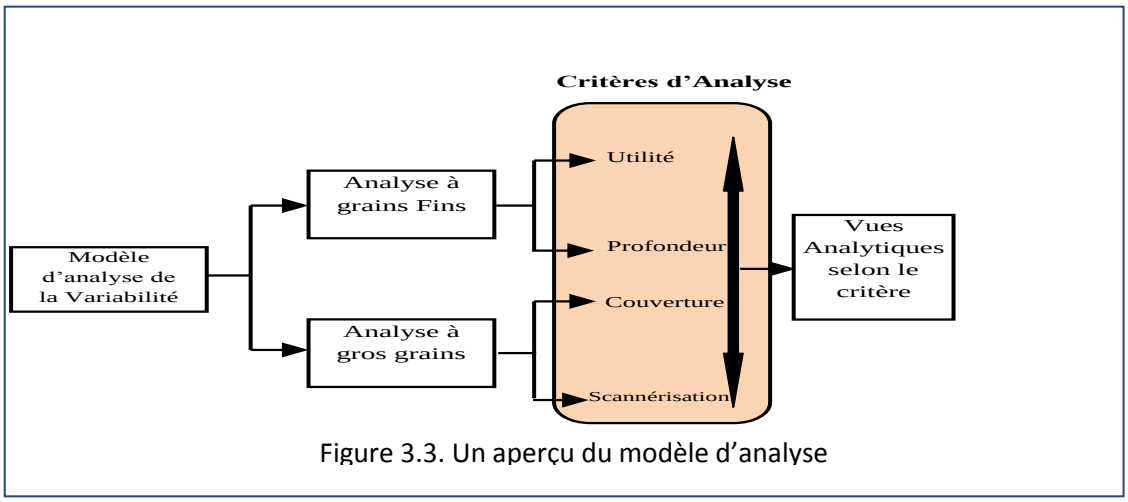


Figure 3.3. Un aperçu du modèle d'analyse

cette dernière, il est possible d'activer un vérin pour déverrouiller la porte après l'authentification. De plus, une option d'affichage de l'état du système d'accès (par l'usage d'un écran) est disponible .Cette option est assurée par la fonction 'Affichage'. Une autre fonctionnalité permet au système de communiquer à distance avec le propriétaire de la maison selon l'un ou les deux modes de communication: 'Wifi' et 'Internet'. Cette dernière

alternative requiert l'activation de la fonction d'affichage 'Affichage'. Le diagramme des caractéristiques (DC) de la ligne (SCAM), qui utilise la notation classique apportée par [9], est illustré à travers la figure 3.2.

Dans ce qui suit, nous allons détailler les étapes clé de notre processus de développement d'une LdP montrées dans le diagramme d'activités de la figure 3.1. Dans la section suivante, nous commençons par décrire le modèle de variabilité.

3.3 Le modèle de variabilité

Dans le modèle de variabilité, une structure interne d'un produit est modélisée par une table (vecteur) booléenne dans laquelle chaque cellule correspond à une unité de construction (entité composant la structure d'un produit telle que la caractéristique).Le concept poursuivi par ce modèle est axé sur l'aspect présentiel de l'unité de construction dans la structure du produit.

Toutes les unités de construction de la ligne de produits doivent être représentées dans cette table. Chaque cellule de la table ne peut prendre que l'une des deux valeurs : '0' ou '1' .La valeur '1' signifie que l'unité de construction du produit est présente dans la configuration du produit et '0' informe que cette unité est absente de cette configuration. Cette modélisation est illustrée dans la table 3.1 pour la ligne de produits (SCAM) avec comme unité de construction : la caractéristique.

3.4 Le modèle d'analyse

Ce modèle est utilisé pour analyser le diagramme des caractéristiques de la LdP selon les différentes formes de construction (unités d'analyse) permises par les structures internes des produits existants selon le critère d'analyse. Notre modèle utilise quatre critères d'analyse.

Le modèle exprime une configuration de produit dans une forme appropriée, requise pour faciliter le calcul des métriques structurelles liées aux critères d'analyse.

Table 3.3. Identification des feuilles dans le graphe des caractéristiques de la LdP (SCAM)

Feature	f_1	f_2	f_3	f_4	f_{11}	f_{12}	f_{31}	f_{32}	f_{41}	f_{42}	f_{321}
$dG^+(fi)$	2	0	2	2	0	0	0	1	1	0	0

Table 3.4 .La vue Profondeur (DV) de la LdP (SCAM)

DV	f_1	f_2	f_3	f_4	f_{11}	f_{12}	f_{31}	f_{32}	f_{41}	f_{42}	f_{321}
SP_1	0	1	0	0	1	0	0	1	0	0	1
SP_2	0	0	0	0	1	0	1	0	0	1	0
SP_3	0	1	0	0	0	1	0	0	0	1	1

L'ensemble considéré des produits logiciels est celui modélisé précédemment dans le cadre du modèle de variabilité .La structure des produits sera toujours considérée dans le modèle d'analyse même si elle a été prédéfinie dans des contextes de projets hétérogènes.

L'analyse est réalisée différemment selon la taille de granularité (à grains fins et à gros grains) des unités d'analyse. Ces dernières seront définies dans la section suivante.

3.4.1 Définition de l'Unité d'Analyse

Dans le modèle d'analyse, chaque produit est perçu et analysé selon sa composition en unités d'analyse. Cette dernière est liée au critère d'analyse .c.à.d. que pour chaque critère, nous aurons à définir sa propre unité d'analyse.

Dans le modèle d'analyse, nous avons proposé des critères d'analyse à granularité fine et à gros grains.

Ces derniers seront détaillés dans la section suivante.

Nous utilisons 4 unités d'analyse dans notre démarche liée chacune à un critère d'analyse .La caractéristique pour l'utilité, la feuille pour la profondeur, le niveau hiérarchique pour la couverture et l'unité de scannérisation pour la scannérisation.

A noter que pour les trois derniers critères, le diagramme des caractéristiques doit être considéré comme un graphe.

3.4.2 Le sous-processus d'analyse

L'analyse de chaque produit est réalisée selon l'unité d'analyse .Ces dernières sont modélisées en fonction des granularités de chaque critère.

La caractéristique et la caractéristique en tant que feuille sont utilisées comme unités d'analyse dans l'analyse à grains fins.

L'analyse à gros grains est basée sur les niveaux hiérarchiques du modèle de caractéristiques et sur une unité spécifique que nous avons appelée 'Unité de Scannérisation'.

Chacun des deux types d'analyse utilise un ensemble de critères. Les critères employés dans l'analyse à grains fins sont l'**Utilité**' et la '**Profondeur**'.

Par contre dans l'analyse à gros grains, nous employons comme critères la '**Couverture par niveau hiérarchique**' et la '**Scannérisation**'.

La figure 3.3 montre la structure de notre modèle d'analyse.

3.4.2.1 L'analyse à grains fins

Nous avons défini deux critères à ce niveau :

L'utilité et

La profondeur.

a. Le critère d'utilité

Nous donnerons la définition du critère puis décrivons les métriques liées et leur application.

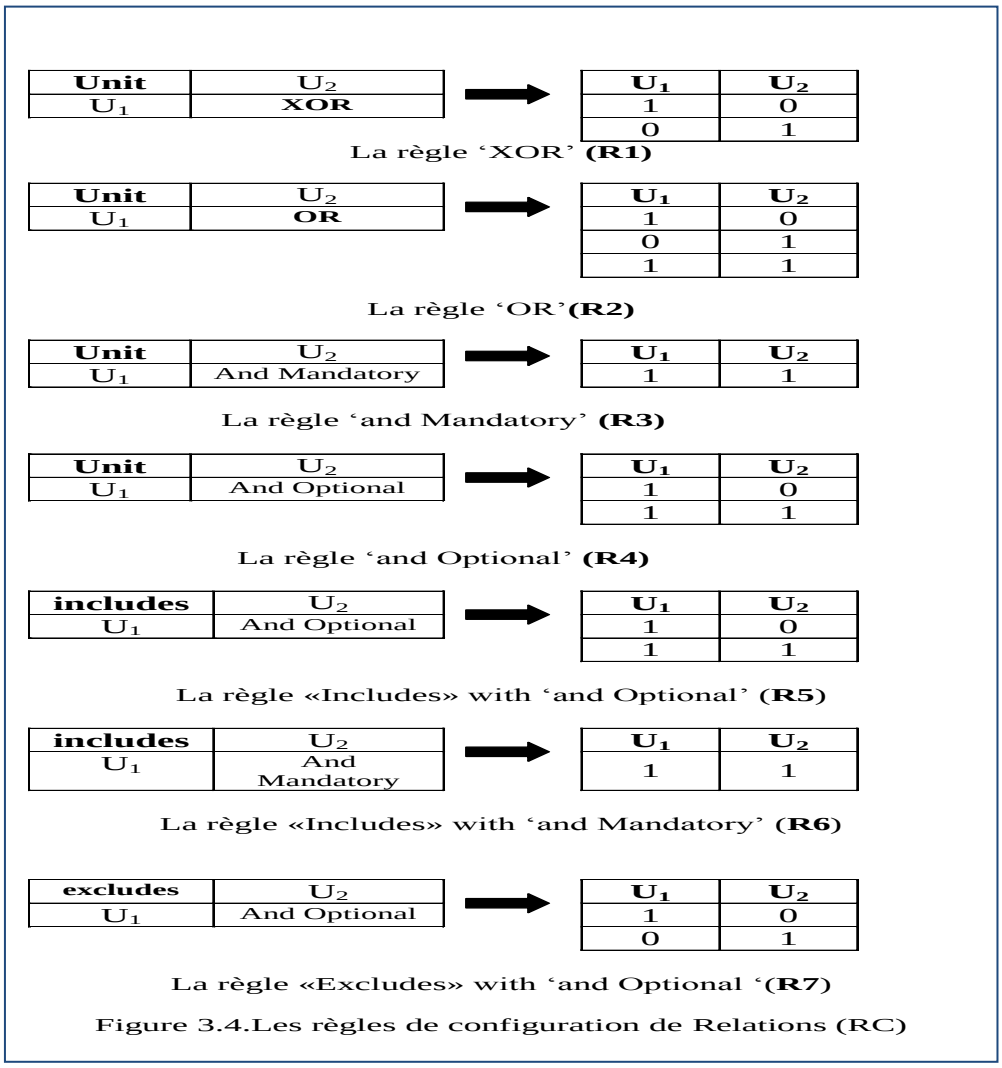


Figure 3.4.Les règles de configuration de Relations (RC)

1. Définition

Le critère d'Utilité permet d'analyser le degré de présence des caractéristiques dans une structure de produit.

Les configurations résultantes sont regroupées en une table appropriée qui représente la « Vue Utilité » des produits notée (UV). Cette dernière portera dans la 1ere colonne l'ensemble des produits et dans la 1ere ligne l'ensemble des caractéristiques composant les structures des produits.

Chaque cellule qui représente le croisement d'une caractéristique et l'appellation du produit dans la vue, portera la valeur '1' si la caractéristique est présente dans la structure de ce produit sinon elle portera la valeur nulle .La Table 3.2 montre une Vue de l'Utilité de la ligne de produits (SCAM).

2. Métriques liées

Avant de définir la métrique essentielle associée à l'utilité (le ratio d'utilité du produit), nous avons besoin de définir au préalable deux autres métriques requises à cet effet, à savoir : le nombre total de caractéristiques modélisées et le cardinal de l'ensemble regroupant les produits en étude dans le modèle de variabilité.

- (NF) est le nombre total de caractéristiques représentées dans le diagramme des caractéristiques à travers le modèle de variabilité.
- (NP) est le cardinal de l'ensemble regroupant les produits en étude préalablement prédéfinis par les parties prenantes que nous noterons (S_p) dans le modèle de variabilité.
- Le « **Ratio d'Utilité »** : le Ratio d'Utilité d'un produit 'i' dans la vue Utilité (UV) est noté (UR_i) ou ($UR(P_i)$) .Il mesure le degré de présence des caractéristiques composant le produit 'i' par rapport à toutes les caractéristiques représentées dans le modèle des caractéristiques. Il est calculé selon la formule suivante :

$$UR(P_j) = (\sum_{J=1}^{NF} UV(i, j)) / NF$$

A noter que les ratios d'utilité doivent être calculés pour l'ensemble des (NP) produits.

3. Application

Dans la ligne de produits (SCAM), Nous aurons: $UR(P_1) = 0.72$. Exprimé sous la forme d'un taux (en pourcentage), $UR(P_1) = 72\%$.

b. Le critère de profondeur

A ce niveau, nous donnons la définition, nous détaillons les métriques liées puis l'application.

1. Définition

La profondeur d'un produit permet d'analyser la structure d'un produit en termes de feuilles dans le diagramme des caractéristiques considéré comme un graphe. Nous noterons ce dernier (GC). Dans ce diagramme, la feuille est définie comme une caractéristique sans successeurs.

Table 3.5 .La vue de Couverture (CV) de la LdP (SCAM)

CV	$HL_{\#1}$	$HL_{\#2}$	$HL_{\#3}$
SP_1	1	1	1
SP_2	1	1	0
SP_3	1	1	1

Table 3.6. La relation entre U_1,U_2 et U_3 dans (C_3)

	U_1	U_2	U_3
U_1		XOR	
U_2	XOR		
U_3		And Opt	

Atteindre plus de feuilles dans la structure d'un produit signifie atteindre plus de profondeur dans le graphe des caractéristiques pour ce même produit. Ce critère s'intéresse à la teneur en feuilles de chaque produit. Les configurations résultantes sont regroupées en une table appropriée qui représente la « Vue profondeur » des produits notée (DV). Cette dernière portera dans la 1^{ere} colonne l'ensemble des produits et dans la 1ere ligne l'ensemble des caractéristiques composant les structures des produits. Chaque cellule qui représente le croisement d'une caractéristique avec l'appellation du produit dans la vue, portera la valeur '1' si la caractéristique en plus d'être une feuille doit être présente dans la structure du produit sinon elle portera la valeur nulle .La Table 3.4 montre une Vue de profondeur de la ligne de produits (SCAM).

2. Métriques associées

Avant de calculer la principale métrique liée à la profondeur que nous avons appelé 'le Ratio de Profondeur' et noté (DR), il y a lieu d'abord d'opérer une analyse de la profondeur du produit pour déterminer le nombre de feuilles et établir la table de profondeur. Ces dernières sont utilisées dans la formule de calcul de (DR).

Pour analyser la profondeur d'un produit, nous avons besoin :

i) De considérer la structure du diagramme des caractéristiques comme une structure de graphe .Nous aurons ainsi un graphe de caractéristiques noté (GC).

ii) D'identifier les feuilles du graphe (GC).

Pour cela, et après avoir noté les caractéristiques par (f_i), nous calculons le demi-degré extérieur des caractéristiques dans le graphe : ($dG^+(f_i)$). Les feuilles correspondent aux caractéristiques (f_i) ayant ($dG^+(fi)$) = 0. La table 3.3 illustre cette identification pour la ligne de produits (SCAM).

iii) De représenter par un tableau Booléen, la structure de chaque produit en termes de feuilles.

iv) De regrouper toutes les configurations qui résultent à partir des étapes précédentes dans une table appropriée que nous avons appelée « la vue profondeur des produits ou (DV) ». Cette vue sous forme de table, représente l'ensemble prédéfini des produits existants en termes de feuilles dans le graphe (GC). La table 3.4 illustre un exemple de cette table pour la ligne de Produits (SCAM).

v) De déterminer le nombre de feuilles noté (NL) du graphe (GC).

vi) De calculer la métrique liée à la profondeur que nous avons appelé 'le Ratio de Profondeur' noté (DR) du produit (Pi) par la formule suivante :

$$DR(Pi) = (\sum_{J=1}^{NF} DV(i, j)) / NL$$

A noter que les ratios de profondeur doivent être calculés pour l'ensemble des (NP) produits.

3. Application

Nous appliquerons la formule de Calcul du ratio de profondeur (DR) pour le produit P_1 dans la ligne de produits (SCAM).

La formule donnera $DR(P_1) = 0.57$. Exprimé en taux, nous aurons $DR(P_1) = 57\ \%$.

3.4.2.2 L'analyse à gros grains

Pour l'analyse à gros grains, nous utilisons deux critères : le critère de « couverture par niveau hiérarchique » et le critère de « scannérisation ».

a. Le critère de Couverture par Niveau Hiérarchique

A ce niveau, nous donnerons la définition, nous décrirons les métriques liées et l'application.

1. Définition

La couverture par niveau Hiérarchique d'un produit permet d'analyser l'utilité d'un produit en termes d'appartenance de ses caractéristiques aux niveaux hiérarchiques du diagramme de caractéristiques considéré comme un graphe (le graphe GC).

Ce critère se base sur une estimation du degré de couverture horizontale de la structure du produit dans le graphe (GC) en termes de caractéristiques.

Pour réaliser cette opération, le processus doit effectuer un balayage plat du graphe par rapport à la structure de chaque produit.

Les configurations résultantes sont regroupées en une table appropriée qui représente la « Vue de Couverture en Niveaux Hiérarchiques » des produits notée (CV). Cette dernière portera dans la 1ere colonne l'ensemble des structures de produits et dans la 1ere ligne l'ensemble des niveaux hiérarchiques composant le graphe (GC). Chaque cellule qui représente le croisement d'un niveau hiérarchique avec l'appellation de la structure du produit dans la vue (CV), portera la valeur '1' si la structure de ce produit comporte au moins une caractéristique appartenant à ce niveau, dans les autres cas elle portera la valeur nulle.

La Table 3.5 montre une Vue de la couverture en niveau hiérarchique de la ligne de produits (SCAM).

2. Métriques associées

Avant de calculer la principale métrique liée à la couverture que nous avons appelé 'le Ratio de Couverture' et noté (CR), il y a lieu au préalable de définir un certain nombre de paramètres puis de les formaliser .Ces paramètres permettent à la fois de bien comprendre

le principe du critère de couverture et sont utilisés dans le calcul du ratio de couverture (CR).Tout d'abord, nous avons besoin d'organiser le graphe (GC) en niveaux hiérarchiques.

Le nombre de niveaux hiérarchiques du graphe (GC) obtenu est noté (NHL).

Nous noterons par ($HL_{\#j}$) le niveau hiérarchique de rang 'j' dans le graphe (GC).

$HL_{\#1}$ est le niveau le plus haut et $HL_{\#NHL}$ est le niveau hiérarchique le plus bas dans (GC).

Chaque niveau hiérarchique contient un ensemble de caractéristiques. Cet ensemble est appelé dans notre modèle : Structure du niveau hiérarchique ou (SHL).

La structure du niveau hiérarchique ($HL_{\#j}$) est notée ($SHL_{\#j}$) .

Pour déterminer si un produit est couvert dans un niveau hiérarchique, nous avons besoin de connaitre si au moins l'une de ses caractéristiques appartient à ce niveau Hiérarchique.

Nous donnerons dans ce qui suit les annotations utilisées dans le modèle formel :

f_i : définit une caractéristique du domaine de la ligne de produits. Exemples : f_1, f_{41}.

S_f : définit l'ensemble de toutes les caractéristiques du domaine de la ligne de produits.

SP_j : définit l'ensemble des caractéristiques composant la structure d'un produit j.

$SPHL_{\#k,m}$: est la structure du produit 'k' en termes caractéristiques relevant du niveau hiérarchique $HL_{\#m}$.

Nous allons à présent définir le modèle formel pour la couverture en niveaux hiérarchiques des produits et celui de la vue de couverture (CV).

$$SP_i \subset S_f \; \forall \backslash \, i \in N \text{ et } i \leq NP$$

$$if \;\; f_k \in (Sp_i) \Rightarrow f_k \in (SHL_{\#j})$$

$$(SHL_{\#j}) \subset S_f, \forall \; j \in N \text{ et } j \leq NHL$$

$$\forall \; f_k \in (S_f) \Rightarrow f_k \in (SHL_{\#j})$$

$$if \;\; (f_k \in (SP_i)) \text{ et } (f_k \in (SHL_{\#j})) \Rightarrow f_k \in (SPHL_{\#i,j})$$

$$if \;\; f_k \in (SHL_{\#j}) \Rightarrow f_k \notin (SHL_{\#m}) \forall (m \in N) et (m \leq NHL)$$

$$\forall (HL_{\#j}) \text{ un niveau hierarchique et } (j \in N) \text{ et } (j \leq NHL) \Rightarrow (HL_{\#j}) \in (SHL) \text{ et } (card(HL_{\#j}) \leq card(S_f))$$

A présent, nous allons définir le modèle formel de la vue de couverture (CV).

$$CV(m,n) = \begin{cases} 1, if.at.least. \exists fk \in (SPm).and.fk \in (SHL \neq n) \\ \\ 0, else \end{cases}$$

Maintenant, nous définissons la métrique liée à la couverture en niveaux hiérarchiques ou le Ratio de Couverture.

Le ratio de couverture noté (CR) est calculé à partir de la table (CV) par la formule suivante :

$$CR(Pi) = (\sum_{J=1}^{NHL} CV(i, j)) / NP$$

A noter que les ratios de couverture doivent être calculés pour l'ensemble des structures des (NP) produits.

3. Application

Par application des formalismes du modèle formel pour la couverture en niveaux hiérarchiques des produits et celui de la vue de couverture en niveaux hiérarchiques (CV) , nous pouvons donner dans ce qui suit , quelques ensembles qui définissent la ligne de Produits (SCAM) selon le critère de couverture en niveaux Hiérarchiques :

$S_f = \{ f_1,f_2,f_3,f_4,f_{11},f_{12},f_{31},f_{32},f_{41},f_{42},f_{321}\}$

$SP1=\{f_1,f_2,f_3,f_4,f_{11},f_{32},f_{41},f_{321}\}$.

$SHL_{\#1}=\{f_1,f_2,f_3,f_4\}$; $SHL_{\#3}=\{f_{321}\}$; $SPHL_{\#1,2}=\{f_{11},f_{32},f_{41}\}$.

Par application des définitions des paramètres pour la ligne de produits (SCAM) , le nombre de niveaux hiérarchiques est de 3. Nous aurons ainsi, dans le graphe (GC) les trois niveaux : $HL_{\#1}$, $HL_{\#2}$ et $HL_{\#3}$.

Pour le calcul du ratio de couverture, le produit P1 couvre les trois niveaux hiérarchiques, il a un ratio de couverture = 3/3=1, et en taux $CR(P_1)$ = 100% ; Le produit P2 couvre deux niveaux hiérarchiques et a un ratio de couverture = 0.66 ;

La vue de couverture (CV) de la ligne de Produits (SCAM) est montrée dans la table 3.5.

b. Le critère de Scannérisation

Pour ce critère, nous précisons quelques définitions, nous décrivons les métriques puis nous aborderons son application.

1. Définitions

- **Le critère de scannérisation**

Ce critère permet d'analyser la structure interne d'un produit selon une unité particulière d'analyse que nous avons appelée 'Unité de Scannérisation ou (US)'.

Ce critère s'intéresse à la teneur en branches de chaque produit. A ce titre le diagramme des caractéristiques est perçu tel un graphe composé d'un ensemble de branches. Les configurations résultantes sont regroupées en une table appropriée qui

représente la « Vue de scannérisation » des produits notée (SV). Cette dernière portera dans la 1ere colonne l'ensemble des structures de produits et dans la 1ere ligne l'ensemble des unités de scannérisation (US) composant les structures des produits. Chaque cellule qui représente le croisement d'une unité de scannérisation avec l'appellation du produit dans la vue, portera la valeur '1' si l'unité de scannérisation concernée est présente dans la structure du produit sinon elle portera la valeur nulle.

La table 3.7 montre une Vue de scannérisation de la ligne de produits (SCAM).

- **L'unité de scannérisation**

L'unité de scannérisation (US) est une unité à gros grains de base utilisée pour analyser la structure interne d'un produit. L'unité de scannérisation (US) scanne la surface du graphe (GC) du plus haut niveau hiérarchique vers le plus bas niveau hiérarchique en mémorisant les arcs trouvés sur son chemin jusqu'à la détection de la première feuille.

L'unité de scannérisation qui correspond à une branche dans une structure arborescente, peut être définie comme un simple et élémentaire chemin composé d'un ensemble d'arcs orientés dans le graphe (GC).

- **La méthode d'Identification des unités de scannérisation**

La technique d'identification des (US) est basée sur l'exploration de toutes les alternatives en termes d'arcs orientés offertes à travers le graphe représentant les caractéristiques.

Nous allons dans ce qui suit en présenter les étapes.

Pour identifier les unités d'analyse (US), nous avons besoin:

i) De considérer le diagramme des caractéristiques comme un **g**raphe **s**ans **r**acine. Nous l'appellerons (GSR).

ii) De connaitre les feuilles du graphe (GSR).

iii)De scinder la structure du graphe (GSR) en composantes connexes, que nous avons notées (Ci).

iv)D'établir la relation entre les arcs orientés notés (Uj) dans les (Ci).

v) D'identifier les Unités de scannérisation (US) de chaque (Ci) selon les règles de configuration des relations définies dans la figure 3.4.

A cet effet, la relation entre une paire d'arcs orientés (Uj) peut être de type :

«XOR», «OR» ,«and Optional», «and Mandatory». «Includes with and Optional», «Includes with and Mandatory» et «Excludes with and Optional».

« Excludes with And Mandatory » n'est pas une configuration valide.

2. Métrique associée

Pour analyser la structure d'un produit en termes de (US), nous utilisons le 'Ratio de Scannérisation' ou '(SR)'.

Ce dernier représente le degré de composition en unités de scannérisation (US) d'un produit, relativement aux autres produits.

Avant de donner la formule de calcul de cette métrique, nous considérons (NSU) comme étant le nombre total d'unités de scannérisation de la ligne de produits.

Le ratio de scannérisation (SR) du produit (P_i) ou ($SR(P_i)$), peut être calculé à partir de la vue (SV) par l'emploi de la formule suivante:

$$SR(Pi) = (\sum_{J=1}^{NSU} SV(i, j)) / NSU$$

A noter que les ratios de scannérisation doivent être calculés pour l'ensemble des structures des (NP) produits.

3. Application

Nous allons calculer le ratio de scannérisation de la structure du produit P_1. Pour cela, nous devons d'abord connaitre toutes les (US) de la ligne de produits par l'application des étapes de la méthode d'identification des (US) pour le graphe des caractéristiques (GSR) de la ligne de Produits (SCAM) :

i) La racine du graphe (GSR) de la ligne de produits (SCAM) est 'f_0'.

Elle est la même pour tous les arcs. C'est pour cette raison qu'elle ne sera pas prise en compte dans notre méthode d'identification pour le graphe (GSR).

Pour connaitre les feuilles du graphe (GSR), nous devons déterminer les demi-degrés extérieurs des caractéristiques composant (GSR) c.à.d. calculer ($dG^+(fi)$).

Les Résultats sont montrés dans la table 3.3.

Pour notre modèle, les caractéristiques : f_2, f_{11}, f_{12}, f_{31}, f_{42}, f_{321} ont '0' comme demi-degré extérieur .Elles représentent donc les feuilles du graphe (GSR).

ii) Après avoir scindé le graphe (GSR) de la ligne (SCAM) en composantes connexes (C_k), Nous avons obtenu les résultats illustrés dans la figure 3.5.

Les ensembles produits à travers les (C_k) sont les suivants:

$C_1=\{f_1,f_{11},f_{12}\}$,

$C_2=\{f_2\}$,

$C_3=\{f_3,f_{31},f_{32},f_{321}\}$,

$C_4=\{f_4,f_{42},f_{41},f_2\}$.

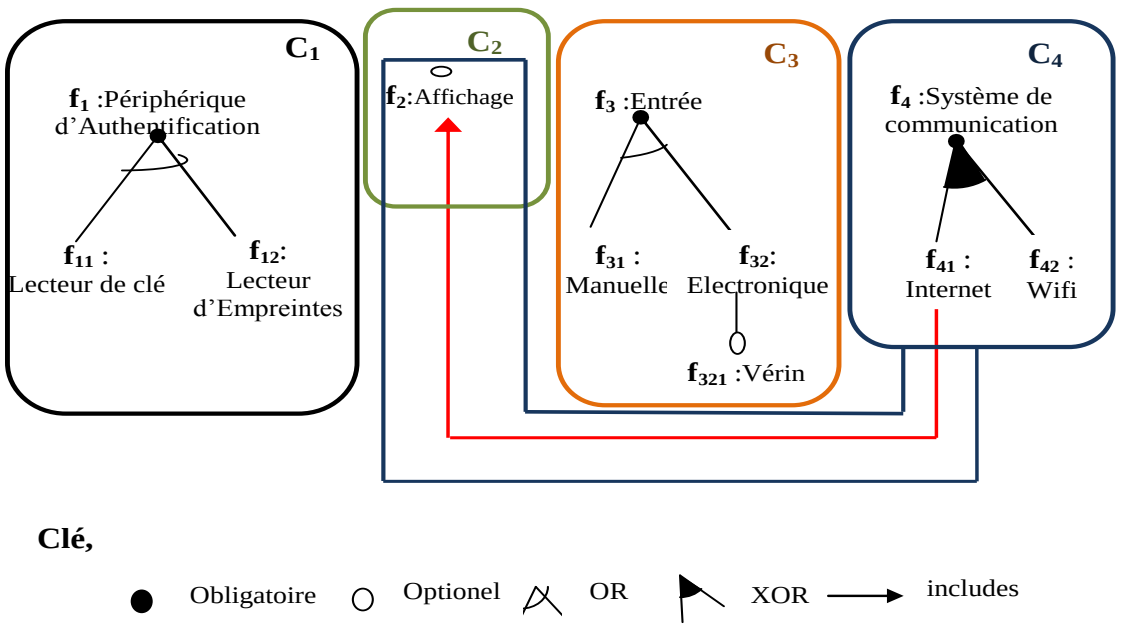


Figure 3.5. Les quatre composantes corrélatives du graphe des caractéristiques de la LdP (SCAM)

iii)Définition des Arcs

Pour des raisons de brièveté et de clarté , nous allons uniquement détailler ci-après la définition des arcs dans (C_3). (U_j) représentent les arcs dans (GSR).

Table 3.7. Les configurations de relation selon les règles (RC)entre U_1,U_2 et U_3 dans (C_3)

Arcs	U_1	U_2	U_3	Configurations	Configurations valides
U_1		1		010	
U_2	1			100	100
U_3			1	011	011
			0	010	010

Chaque (U_j) est une paire de relation 'Une Caractéristique-Un successeur' dans (C_k).

Par l'application des règles (RC) à la table des configurations résultant pour (U_j) dans (C_3), nous aurons trois arcs: U_1, U_2 et U_3 définis comme suit:

$U_1=\{f_3,f_{31}\}$, $U_2=\{f_3,f_{32}\}$,$U_3=\{f_{32},f_{321}\}$.

iv) Etudions maintenant la relation entre les arcs.

La table 3.6 détaille les règles de configuration des relations (RC) de U_1, U_2 et U_3 considérées paire par paire dans (C_3).

v) Identification des (US) pour la composante connexe (C_3).

Par l'application des règles (RC) à la table des configurations résultant après l'étape (iv), nous obtenons les configurations montrées en rouge dans la table 3.7.

En supprimant les configurations en double, nous obtiendrons les configurations valides comme illustrées dans la table 3.7 (la colonne des configurations valides).

Les résultats finaux donneront les configurations valides suivantes : '101', '011' et '010'. Soit pour chaque configuration valide, les (US) suivantes :

'101' donnera une (US) = U_1= (f_3,f_{31})

'011'donnera un second (US) = (U_2,U_3) = (f_3,f_{32},f_{321})

'010'donnera un troisième (US) = U_2= (f_3,f_{32}).

En appliquant le même processus aux autres composantes connexes (C_k) du graphe (GSR), les ensembles finaux résultant de (US) seront les suivants :

A partir de (C_1) , nous aurons : US_1= {f_1,f_{11}} ; US_2={f_1,f_{12}}

A partir de (C_2) , nous aurons : US_3= {f_2}

A partir de (C_3) , nous aurons : US_4={f_3,f_{31}} et US_5={f_3,f_{32}} ;US_6={f_3,f_{32},f_{321}}

A partir de (C_4) , nous aurons : US_7= {f_4,f_{41},f_2}; US_8={f_4,f_{42}}; US_9={f_4,f_{42},f_{41},f_2}

Table 3.8 .La vue Scannérisation (SV) de la LdP (SCAM)

SV	US_1	US_2	US_3	US_4	US_5	US_6	US_7	US_8	US_9
SP_1	0	1	0	0	0	1	0	0	0
SP_2	1	0	0	1	0	0	0	1	0
SP_3	0	1	1	0	0	1	0	0	1

.

Table 3.9 .Les (CoV) Potentielles dans SEV1

DV	f_1	f_2	f_3	f_4	f_{11}	f_{12}	f_{31}	f_{32}	f_{41}	f_{42}	f_{321}
SP_1	0	1	0	0	1	0	0	1	0	0	1
SP_2	0	0	0	0	1	0	1	0	0	1	0
SP_3	0	1	0	0	0	1	0	0	0	1	1

La vue Scannérisation de la ligne de Produits (SCAM) pour l'ensemble initial des produits prédéfinis est illustrée dans la table 3.8.

Calculons à présent le ratio de scannérisation de P1. Nous aurons : SR(P1) = 0.22.

En taux , nous aurons SR(P1) = 22%.

3.4.3 Les activités d'Identification et de Configuration

A travers cette section, nous allons détailler les tâches accomplies par le processus de développement dans le cadre des deux activités d'analyse : l'identification et la configuration.

3.4.3.1 L'activité d'identification

Cette activité permet d'identifier le cadre de variation (EV) à partir de la vue d'analyse initiale selon le critère (VAIC).

L'identification décompose le cadre de variation (EV) en sous-ensembles de configurations (SEVk) selon la similarité des dépendances entre ses éléments c.à.d. ceux qui ont des interdépendances ou des dépendances identiques.

Nous utilisons ces similarités pour établir les contraintes de configuration (CC).

L'activité d'identification permet également d'identifier l'ensemble des configurations de variation (CoV) à partir de chaque cadre de variation (EV) avec considération des contraintes de configuration (CC).

Ces ensembles préparés, lors de cette étape, seront utilisés par l'activité de configuration.

3.4.3.2 L'activité de configuration

Cette activité a pour but de capturer la configurabilité contenue dans l'ensemble prédéfini des produits selon le critère d'analyse. Cette tâche est réalisée en faisant varier les configurations initiales (SP_i) dans la vue initiale d'analyse (VAIC) en accordance avec le cadre de variation du critère (EV).

Pour chaque configuration de variation (CoV), une nouvelle vue est établie dans le modèle d'analyse .Ainsi, la vue d'analyse initiale (VAIC) peut être configurée en plusieurs vues. La configuration variable de (Sp_i) dans le cadre (SEV) en accordance avec les contraintes (CC) est notée (ΔSP_i) (figure 3.7). Le nombre de configurations produites noté (NC) obtenu pour chaque cadre (SEV) dans la (VAC) permet de mesurer la configurabilité dans ce (SEV) pour chaque structure de produit.

Table 3.10a. La (VAC) de profondeur pour la (CoV) = {f_2} dans SEV1

SFV1	f_1	f_2	f_3	f_4	f_{11}	f_{12}	f_{31}	f_{32}	f_{41}	f_{42}	f_{321}
ΔSP_1	0	1	0	0	1	0	0	1	0	0	1
ΔSP_2	0	1	0	0	1	0	1	0	0	1	0
ΔSP_3	0	1	0	0	0	1	0	0	0	1	1

Table 3.10b.La (VCC) de profondeur pour La (CoV) = {f_2} dans SEV1

SVF1	ΔSP_1	ΔSP_2	ΔSP_3
Si	0.57	0.57	0.57
V_i	0.14	0.14	0.14
V_{12}	0.14	0.14	0
V_{13}	0.14	0	0.14
V_{23}	0	0.14	0.14
V_{123}	0.14	0.14	0.14
MC	4	4	4
RS	0.28	0.28	0.28
RR	0.42	0.42	0.42

3.4.3.3 Exemples de fonctionnement des activités d'identification et de configuration

Pour faciliter l'explication du déroulement de notre processus, nous continuons à utiliser le critère de profondeur comme dans les exemples des étapes précédentes.

La vue d'analyse initiale (VAIC) est représentée dans la table 3.4.

Le cadre de variation associé au critère de profondeur = EV = { f_2,f_{11},f_{12},f_{31},f_{32},f_{42},f_{321}}

a. Exemple de sous-cadres de configuration (SEV) sans cadre de variation valide

L'Identification donne un premier SEV: SEV1 composé de f_{11} et f_{12}; SEV1={ f_{11},f_{12}}

Les contraintes (CC) appliquées à SEV1 dans la (VAC) sont les suivantes:

CC1: f_{11} ← – → f_{12} (exclusive OR).
CC2: f_{12} ← – → f_{11}

Après considération des possibles configurations de variation (CoV) pour (SEV1) dans la (VAC) (montrées encadrées dans la table 3.9) et après prise en compte des contraintes (CC) définies précédemment, les configurations de variation représentées séparément par les trois cadres dans la table 3.9 ne sont pas valides. Ainsi, l'ensemble des configurations de variation de (SEV) est vide (= {Ø}) et donc, il ne peut y avoir de (VAC) résultant de cette configuration.

b. Exemple de sous-ensembles de configuration (SEV) avec un (EV) valide: SEV= {f_2}

L'Identification donne un second SEV: SEV2={f_2}

Il n y a pas de contraintes de configuration (CC) qui peuvent être appliquées à (VC) pour (SEV2).Il y a uniquement une seule et possible configuration de variation pour f_2. Elle prend "1" pour (SP_2) montré par un petit cadre dans la table 3.13 et devient (ΔSP2).Cette configuration de variation est valide (pas de contraintes de configuration (CC) pour ce sous-cadre SEV).

La nouvelle vue analytique de profondeur (VAC) produite après considération de cette configuration de variation (CoV) est représentée dans la table 3.13.

Table 3.11a .La (VAC) de profondeur pour la (CoV) = {f_{42}} dans SEV2

SFV2	f_1	f_2	f_3	f_4	f_{11}	f_{12}	f_{31}	f_{32}	f_{41}	f_{42}	f_{321}
ΔSP_1	0	1	0	0	1	0	0	1	0	1	1
ΔSP_2	0	0	0	0	1	0	1	0	0	1	0
ΔSP_3	0	1	0	0	0	1	0	0	0	1	1

Table 3.11b.La (VCC) de profondeur pour La (CoV) = {f_{42}} dans SEV2

SVF2	ΔSP_1	ΔSP_2	ΔSP_3
Si	0.71	0.42	0.57
V_i	0.14	0.14	0.14
V_{12}	0.14	0.14	0
V_{13}	0.28	0	0.28
V_{23}	0	0	0
V_{123}	0.14	0.14	0.14
MC	4	4	4
RS	0.42	0.14	0.28
RR	0.56	0.28	0.42

3.5 Le Modèle de Corrélation

Dans cette partie de la thèse, nous avons employé un modèle similaire au modèle de relations utilisé par [69] que nous avons adapté à notre étude dans l'étape de Mapping ainsi que pour le calcul des métriques. Dans la section suivante, nous allons décrire la structure d'un produit selon notre modèle de corrélation.

3.5.1 La Structure interne du produit dans le Modéle de corrélation

Le modéle de corrélation perçoit la structure interne d'un produit 'p'*:* que nous avons notée 'SP_p' comme un ensemble composé de quatre parties (1,2,3 et 4). Pour des raisons de brièveté et de clarté, nous avons illustré la structure du produit selon le modéle de corrélation à l'aide de seulement un ensemble de trois configurations de produits prédefinies initialement dans la table 3.2 comme produits existants.La description des parties 1,2,3 et 4 est alors la suivante :

① Est l'ensemble des éléments propres au produit 'p' .Ces éléments sont sélectionnés uniquement dans la configuration de 'p' et pas dans les autres configurations de produits.
Cet ensemble représente la '*différentiabilité*' de 'p'. Nous pouvons voir les parties '1' des produits dans la figure 3.6 représentées par la dénomination (D_i)

② Ce sont les ensembles d'éléments partagés par une paire de configurations de produits que nous noterons (p_i and p_j) mais pas par les autres produits. Ces ensembles représentent la '*similarité*' entre cette paire de produits. Les parties '2' sont montrées dans la figure 3.6 par l'abréviation (R_{ij}).

③ Est l'ensemble des éléments communs à tous les produits. Cet ensemble représente la '*commonalité*' .Cette partie est illustrée dans la figure 3.6 par l'abréviation (R_{ijk}).

④ Ce sont tous les éléments qui composent les *structures internes* des produits.
Ces ensembles sont utilisés dans notre étude pour lier le modèle d'analyse au modèle de corrélation .Ils sont illustrés dans la figure 3.6 par la notation (S_p).

3.5.2 Les Métriques du Modéle de Corrélation

Quatre métriques permettent d'estimer en termes d'unité d'analyse (UA) , la taille de chaque partie composant la structure d'un produit en proportion avec les autres parties . Nous citerons ces metriques comme suit :

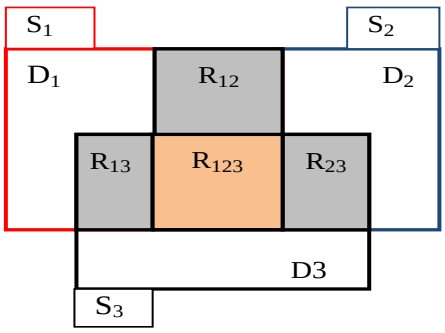
S_i: permet de mesurer *la taille de la structure interne* du produit 'i'.
V_i: permet de mesurer *la différentiabilité* du produit 'i'.
V_{ij}: permet de mesurer *la similarité* entre deux produits 'i' et 'j'.
V_{ijk}: permet de mesurer *la taille de la commonalité.*

Ces métriques sont regroupées dans une vue particulière des produits que nous avons appelée « la vue de corrélation du critère » ou vue corrélative (VCC) en concordance avec le critère d'analyse mappé.

De plus, ce modèle est employé pour calculer la configurabilité de l'ensemble initial des produits prédéfinis (Pi) selon le sous-cadre variable (SEV) et les contraintes de configuration (CC) du critère d'analyse. Ces derniers seront définis dans la section 3.7.

Le résultat de la configurabilité de la structure prédéfinie du produit c.à.d. (SPi) dans le (SEV) est une nouvelle structure du produit que nous avons notée (ΔSPi).

Un exemple a été fourni dans la section 3.4.3.3.



Key,

S1 : la structure interne du produit1 (SP_1);
$S_1=D_1+R_{12}+R_{13}+R_{123}$
S2 : la structure interne du produit2 (SP_2).
$S_2=D_2+R_{12}+R_{23}+R_{123}$
S3 : la structure interne du produit3.(SP_3)
$S_3=D_3+R_{13}+R_{23}+R_{123}$
Di : la differentiabilité de SPi
R_{ij} : la similarité entre (Spi) et (SPj)
R_{ijk} : la commonalité

Figure 3.6.Structure Interne d'un produit selon le Modèle de Correlation

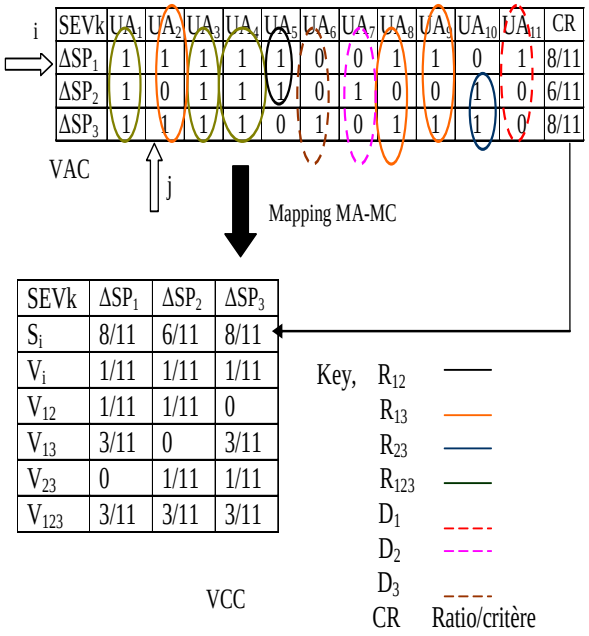


Figure 3.7. Le Mapping "Modèle d'Analyse-Modèle de Corrélation "

La métrique associée à la configurabilité , est le nombre maximal de configurations potentielles (NC) obtenues pour chaque sous-cadre variable (SEV).

Le modèle de corrélation est utilisé également pour estimer la réutilisabilité (similarité plus commonalité) dans les produits logiciels avec la considération de son seuil maximal selon le critère d'analyse et la configurabilité (avec considération du nombre maximal de configurations).

3.6 Le Modèle de Mapping

Le modele de Mapping est basé sur un formalisme qui permet de transfomer une vue d'analyse en une vue de corrélation selon le critére d'analyse .Dans le paragraphe suivant , nous allons définir la représentation formelle du modéle de Mapping .

$\forall i \leq NP$, *(SPi) est une structure du produit 'i' représentée dans la (VAC) initiale selon le critère d'analyse et (ΔSPi) est la structure du produit 'i' représentée dans la (VAC) de SEV après l'avoir soumis au contraintes (CC) du sous-cadre variable SEV.*
La définition de (S_i) dans le modèle de corrélation est la suivante :

$$(Si)=\begin{cases}(UR_i)\ \text{si le critère d'analyse est l'utilité}\\ (DR_i)\ \text{si le critère d'analyse est la profondeur}\\ (CR_i)\ \text{si le critère d'analyse est la couverture en niveaux hiérarchiques}\\ (SR_i)\ \text{si le critère d'analyse est la scannérisation}\end{cases}$$

Les définitions formelles de (D_i) , (R_{ij}) ,(R_{ijk}) et leurs métriques respectives (V_i) , (V_{ij}) et ($V_{1\ldots n}$) sont les suivantes :

$$Di=\begin{bmatrix}\delta i1\\ \delta i2.\\ .\\ \delta in\end{bmatrix} where \delta ik=\begin{cases}1, if\ (i=k)\\ \\ 0, else\end{cases}$$

$$if\ \ f_k \in (Sp_i) \Rightarrow f_k \notin (SP_j),\ j \neq i,$$
$$Vi = card(Di)/NF$$

$$Rij=\begin{bmatrix}\delta j1\\ \delta j2.\\ .\\ \delta jn\end{bmatrix} where \delta ijk=\begin{cases}1, (k=i) or (k=j)\\ \\ 0, (k\neq i) or (k\neq j)\end{cases}$$

$$if\ \ f_k \in (Sp_i),\ f_k \in (SP_j).and.only.to(SPj),$$
$$Vij = card(Rij)/NF$$

$$R1,\ldots,n=\begin{bmatrix}\delta 1\\ \delta 2.\\ .\\ \delta n\end{bmatrix} where \delta i=1, i=1,\ldots,n$$

$$V1\ldots,n = card(R1\ldots,n)/NF$$

La figure 3.7 illustre les mécanismes de fonctionnement du modèle de Mapping qui transforment une vue d'analyse selon un critère (VAC) en une vue correspondante de corrélation (VCC) pour ce même critère.

3.7 Le Modèle d'Evaluation

Ce modèle se base essentiellement sur l'exécution d'une activité : « l'évaluation », elle-même scindée en deux sous-activités, la génération des métriques et rapports d'évaluation et l'optimisation.

Table 3.12a .La (VAC) de profondeur pour la (CoV) = $\{f_2,f_{42}\}$ dans SEV3

SEV3	f_1	f_2	f_3	f_4	f_{11}	f_{12}	f_{31}	f_{32}	f_{41}	f_{42}	f_{321}
ΔSP_1	0	1	0	0	1	0	0	1	0	1	1
ΔSP_2	0	1	0	0	1	0	1	0	0	1	0
ΔSP_3	0	1	0	0	0	1	0	0	0	1	1

Table 3.12b.La (VCC) de profondeur pour la (CoV) = $\{f_2,f_{42}\}$ dans SEV3

SEV3	ΔSP_1	ΔSP_2	ΔSP_3
Si	0.71	0.57	0.57
V_i	0.14	0.14	0.14
V_{12}	0.14	0.14	0
V_{13}	0.14	0	0.14
V_{23}	0	0	0
V_{123}	0.28	0.28	0.28
MC	5	5	5
RS	0.28	0.14	0.14
RR	0.56	0.42	0.42

Dans le but de faciliter la compréhension des différentes métriques utilisées par l'activité d'évaluation (Evaluation dans le diagramme des activités de notre processus de développement) et toujours dans un intérêt de brièveté et de clarté, nous allons décrire seulement cette sous-activité pour le critère de profondeur dans la ligne de Produits (SCAM).

3.7.1 La sous-activité de génération des métriques et rapports d'évaluation

Cette sous-activité liée à l'activité d'évaluation permet de calculer un certain nombre de métriques d'évaluation puis de les faire ressortir groupées sous forme de rapports d'évaluation.

Ces derniers établis pour chaque critére , comportent en particulier les résultats des mesures des differents ratios maximaux (le Ratio maximal de Réutilisabilité et le Ratio maximal de Similarité qui seront décrits plus loin dans cette section) , pour la vue initiale et pour les differents sous-cadres de variation SEV et ce pour toutes les variations de produits ΔSPi. C'est principalement à partir de ces rapports que s'exécutera la sous-activité d'optimisation.

En plus de la taille de configurabilité précédemment définie dans la section abordant l'activité de configuration, le processus de développement de la ligne permet de générer à travers la sous-activité de génération des métriques d'évaluation un certain nombre d'autres métriques à savoir :

La Taille et le Ratio de Similarité entre une paire de produits, la taille de la commonalité pour l'ensemble des produits, le ratio de réutilisabilité et le ratio de différentiabilité du produit.

Nous les définissons dans ce qui suit :

- La *Taille de la similarité*: la taille de la similarité (V_{ij} dans la (VCC) de la figure 3.7) mesure la taille en termes de structure interne similaire d'une paire de produits.

- La *Taille de la commonalité*: la taille de la commonalité est mesurée pour un ensemble de produits considérés dans la vue de corrélation selon le critère (VCC).
  Elle est représentée par (V_{ijk}) dans la vue (VCC) de la figure 3.7.
- Le *Ratio de Similarité*: le ratio de similarité (RS) de la structure d'un produit 'j' (SP_i) ou de sa forme variable (ΔSP_j) dans (SEV) est calculé à partir de la vue de corrélation (VCC) par la formule suivante :

$$RS(SPj) = \sum_{J=1}^{NP} Vij$$

- Le *Ratio de Réutilisabilité* (RR): le ratio de réutilisabilité de la structure d'un produit 'j' (SPj) ou de sa forme variable (ΔSPj) dans (SEV) est le cumul du ratio de similarité de (SPj) (ou ΔSP_j) et de la taille de la commonalité.
  Il est calculé à partir de la vue de corrélation (VCC) à l'aide de la formule suivante :

$$RR(SPj) = (\sum_{J=1}^{NP} Vij) + (Vij..k)$$

  ($V_{ij...k}$) représente la taille de la commonalité que nous avons déjà définie dans la section 3.6.
- Le *Ratio de différentiabilité* (RD): le ratio de différentiabilité de la structure d'un produit 'j' (SPj) ou de sa forme variable (ΔSPj) dans (SEV) mesure le degré de différence en structure interne de ce produit par rapport aux autres produits.
  Il représente l'aspect contraire de la commonalité. Il est représenté par (V_i) dans la table 3.7.
- (MC) représente *le nombre maximal de configurations* dans un SEV. Nous avons également noté *le nombre de configurations permises pour un produit* par (NC).

Dans ce qui suit , nous allons donner un exemple de calcul des métriques d'évaluation toujours pour le critére de profondeur.

Tout d'abord, nous allons déterminer tous les paramètres en entrée du rapport d'évaluation , à savoir l'ensemble des SEV concernés par le cadre de variation EV.

EV= {f_2,f_{11},f_{12},f_{31},f_{32},f_{42},f_{321}}

Les Contraintes de Configuration (CC) sont :

CC1: f_{11} ⟷ f_{12}
CC2: f_{31} ⟷ f_{32} , f_{321} ⟷ f_{32}
CC3: f_{41} ⟶ f_2

La vue d'analyse initiale (VAIC) est montrée dans la table 3.4. Elle inclut 3 configurations.

Après application des contraintes (CC) pour le cadre (EV) dans la vue VAIC de profondeur, l'identification produit 3 différents sous-cadres de variation (SEV), à savoir : SEV1= {f_2}; SEV2= {f_{42}} et SEV3= {f_2,f_{42}}.

Le Mapping de la vue d'analyse initiale (VAIC) dans le modèle de corrélation donne la vue de corrélation (VCIC) initiale montrée dans la table 3.14.

(RR) représente le Ratio de Réutilisabilité et (RS) le Ratio de Similarité .Ils seront décrits plus loin dans cette section.

1. Pour (SEV1):

Après application des contraintes de configuration (CC) dans SEV1, le résultat obtenu est une seule configuration variable valide (CoV) = {f_2}.

En considérant cette (CoV) dans (SEV1), cette dernière permet d'avoir :

La vue d'analyse de profondeur (VAC) ; et

Sa vue de corrélation correspondante, la vue (VCC) dans le modèle de corrélation.

Ces deux vues sont illustrées respectivement dans la table 3.10.a et dans la table 3.10.b.

2. Pour (SEV2):

Par l'application des contraintes (CC) dans SEV2, le résultat est une seule configuration (CoV) valide = {f_{42}}.

En considérant cette (CoV) dans (SEV2), la configuration permet d'avoir :

La vue d'analyse de profondeur (VAC), et sa vue de corrélation correspondante dans le modèle de corrélation (VCC).

Table 3.13 Une possible (CoV) dans SEV2

DV	f_1	f_2	f_3	f_4	f_{11}	f_{12}	f_{31}	f_{32}	f_{41}	f_{42}	f_{321}
ΔSP_1	0	1	0	0	1	0	0	1	0	0	1
ΔSP_2	0	1	0	0	1	0	1	0	0	1	0
ΔSP_3	0	1	0	0	0	1	0	0	0	1	1

Table 3.14.La (VCC) initiale du critère de profondeur

VCCI	ΔSP_1	ΔSP_2	ΔSP_3
Si	0.57	0.42	0.57
V_i	0.14	0.14	0.14
V_{12}	0.14	0.14	0
V_{13}	0.28	0	0.28
V_{23}	0	0.14	0.14
V_{123}	0	0	0
MC	3	3	3
RS	0.42	0.28	0.42
RR	0.42	0.28	0.42

Les deux vues sont représentées respectivement dans la table 3.11.a et dans la table 3.11.b.

3. Pour (SEV3):

Après application des contraintes (CC) dans le sous-cadre SEV3, le résultat est une seule configuration valide (CoV) = {f_{42},f_2}) dans (SEV3) .

En considérant cette (CoV) dans (SEV3), la configuration permet d'avoir :

La vue d'analyse (VAC) de profondeur, et sa vue correspondante dans le modèle de corrélation (VCC).

Les deux vues sont illustrées respectivement dans la table 3.12.a et dans la table 3.12.b.

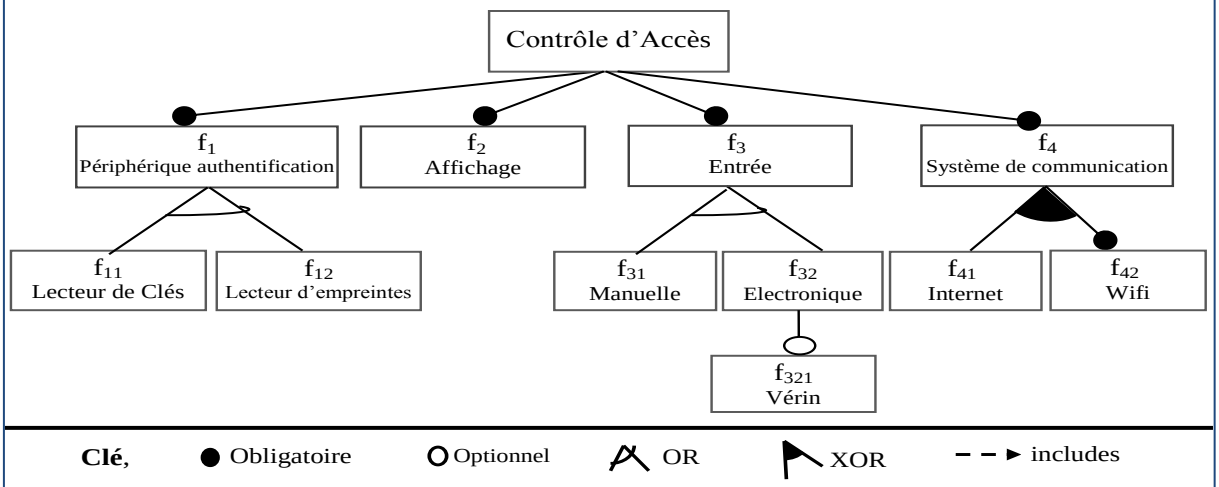


Figure 3.8. La vue optimale du diagramme des caractéristiques de la LdP (SCAM) selon le critère de profondeur

A présent, nous allons entamer le calcul des métriques liées à l'évaluation.

Dans la vue d'analyse initiale (VAIC) , le nombre de configurations (NC) est de trois (table 3.13) . Dans SEV2 , une autre configuration à partir de SP_2 (table 3.13) est ajoutée aux configurations initiales (de la VAIC). Le nombre de configurations devient alors 4 dans SEV2. Dans SEV3 , il est de 5. Le calcul des ratios de similarité (RS) et de réutilisabilité (RR) donneront comme ratios maximaux respectifs pour RS : 0.42 et pour RR : 0.56.

Les métriques d'évaluation sont regroupées dans le rapport d'évaluation selon le critère de profondeur montré dans la table 3.15.

Table 3.15 .Le rapport d'évaluation de la profondeur

profondeur	ΔSP_1	ΔSP_2	ΔSP_3	MC	Max RS	Max RR
Initial	0.42	0.28	0.42	3	0.42	0.56
SEV1	0.42	0.42	0.42	4		
SEV2	0.56	0.28	0.42	4		
SEV3	0.56	0.42	0.42	5		

Table 3.16 .Le rapport d'évaluation de l'utilité

Utilité	ΔSP_1	ΔSP_2	ΔSP_3	MC	Max RS	Max RR
Initial	0.72	0.45	0.72	3	0.45	0.81
SEV1	0.72	0.54	0.72	4		
SEV2	0.72	0.63	0.72	4		
SEV3	0.81	0.45	0.72	4		
SEV4	0.81	0.63	0.72	5		

3.7.2 La sous-activité d'optimisation

L'Optimisation cherche dans une 1ere phase à identifier le sous-cadre de variation (SEV) qui dispose du seuil maximal de réutilisabilité dans le rapport d'évaluation selon le critère

d'analyse et dans une deuxième à déterminer le diagramme des caractéristiques optimal et global de la ligne de Produits à travers l'optimisation globale .

3.7.2.1 Première phase

Nous aborderons cette section par la description de la démarche puis de son application.

a. Démarche

Le résultat crucial est le sous-cadre de variation optimal du critère d'analyse (ou SEVO). Une fois identifié, la vue de corrélation (VCC) correspondante (selon ce même critère) au sous-cadre de variation optimal (SEVO) est considérée comme étant la vue de corrélation (VCC) optimale selon ce critère. Nous l'avons appelée (VCCO) et la vue d'analyse (VAC) correspondante à cette vue optimale de corrélation selon ce même critère (VCCO) comme étant la vue d'analyse optimale. Nous l'avons appelée (VACO).

Le Mapping de la vue d'analyse optimale (VACO) en modèle de caractéristiques selon le critère donnera la vue optimale du diagramme des caractéristiques pour ce critère et selon la vue. Nous l'avons appelé (VODC).

b. Application

Dans notre cas, et toujours pour le critère de profondeur dans la ligne de produits (SCAM), le (SEV3) est le SEV optimal c.à.d. que c'est ce sous-cadre qui représente le (SEVO).

La vue de corrélation (VCC) de (SEV3) est la vue optimale de corrélation de profondeur (VCCO) .Elle permet de déterminer la vue d'analyse (VAC) correspondante c.à.d. la vue d'analyse optimale de profondeur (VACO). Les tables 3.12.a et 3.12.b montrent ces deux vues optimales pour le critère de profondeur.

Le Mapping de la vue d'analyse optimale (VACO) en modèle de caractéristiques donnera la vue optimale du diagramme de caractéristiques. Il correspond au diagramme des caractéristiques de la ligne de produits illustré dans la figure 3.8.

En appliquant le même processus d'évaluation aux 3 autres critères, les résultats seront des ensembles de (VACO, VCCO, VODC) établis pour chaque critère.

2éme Phase : l'Optimisation globale

Nous aborderons cette partie par l'explication de la démarche puis par la description de son application.

a. Démarche

Cette sous-activité basée sur un modèle d'intégration, complète et finalise l'optimisation. Elle a pour but de déterminer le diagramme des caractéristiques optimal de la ligne de produits en considérant les quatre critères.

Elle sera réalisée à travers la composition des quatre diagrammes des caractéristiques optimaux selon le critère suivie par les mécanismes d'intégration proposés par [116].

La vue qui en résulte correspond au diagramme des caractéristiques global et optimal de la ligne de produits.

b. Application

A présent, nous allons appliquer les mécanismes d'optimisation mentionnés précédemment au reste des critères d'analyse dans la ligne de Produits (SCAM).

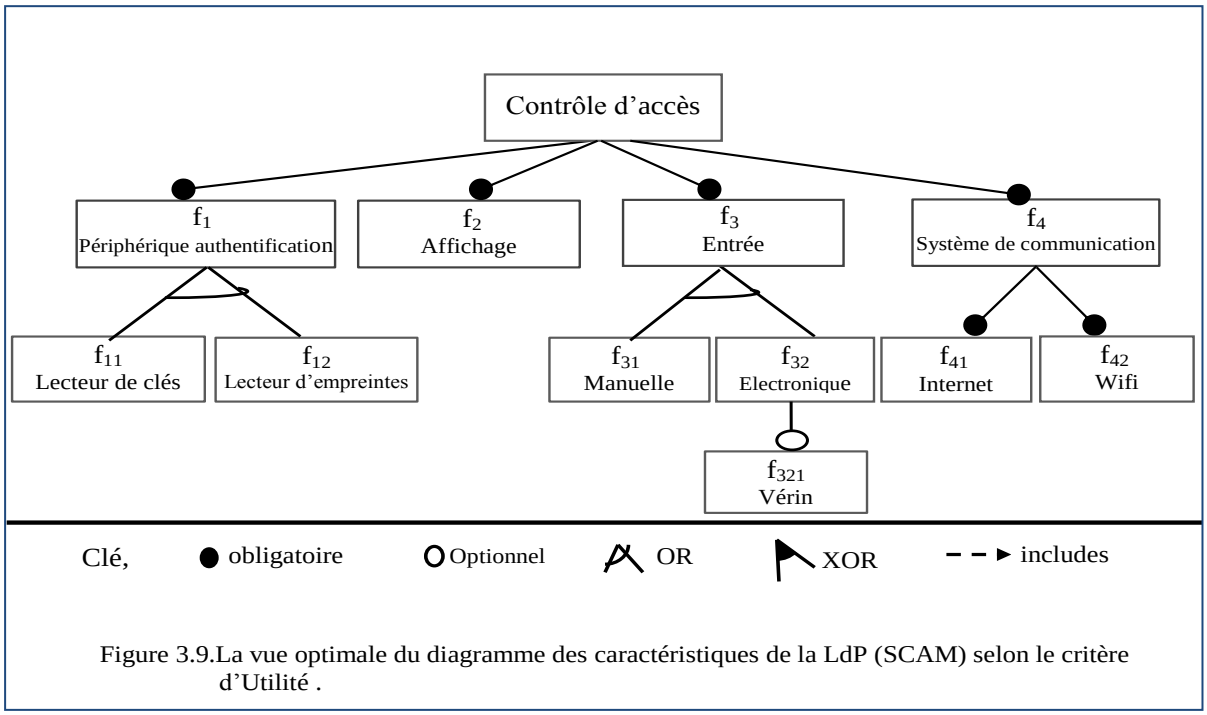


Figure 3.9.La vue optimale du diagramme des caractéristiques de la LdP (SCAM) selon le critère d'Utilité .

Par l'application du même processus d'évaluation (opéré pour le critère de profondeur) aux autres critères d'analyse, les résultats produits seront les suivants :

Critère : Utilité

- EV={f_1,f_2,f_3,f_4,f_{11},f_{12},f_{31},f_{32},f_{41},f_{42},f_{321}}
- Les contraintes (CC) :
  - CC1 :f_{11} ⟷ f_{12}
  - CC2: f_{31} ⟷ f_{32} Λ f_{321}.
  - CC3: f_{321} ⇢ f_{31} (Excludes).
  - Les SEV: SEV1= {f_2} , SEV2={f_{41}}, SEV3={f_{42},f_2} et SEV4={f_{42}}

En appliquant le même processus d'évaluation utilisé pour le critère de profondeur, nous obtenons les résultats mentionnés dans le rapport d'évaluation de la table 3.16 pour ce critère.

Après application des phases de la sous-activité d'optimisation employée pour le critère de profondeur, les résultats donneront (SEV4) comme sous-cadre optimal (SEVO) et la vue correspondant à SEV4 comme étant la vue d'analyse optimale (VACO).

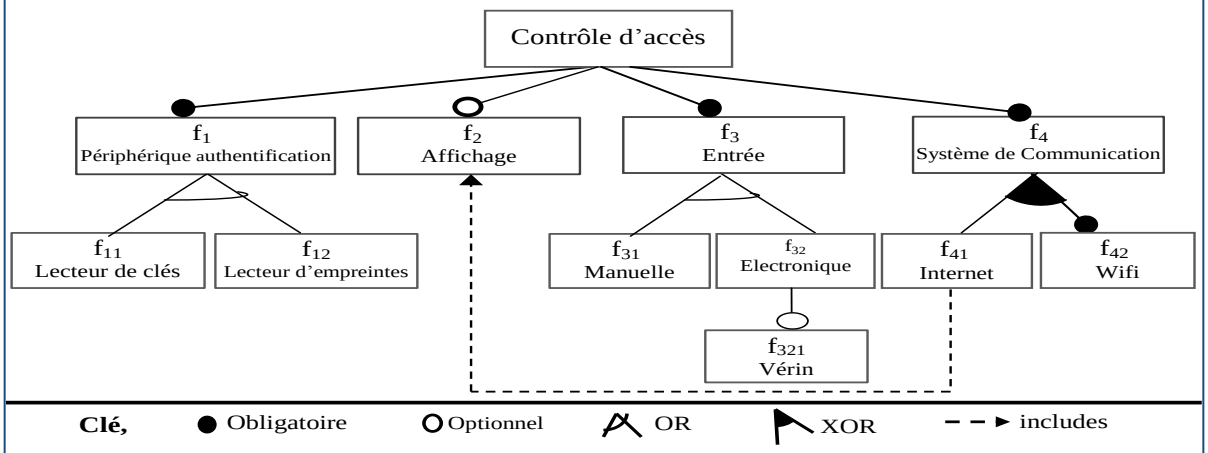


Figure 3.10. La vue Optimale du Diagramme des caractéristiques pour le critère de couverture de la LdP (SCAM)

Le Mapping de la vue (VACO) en modèle de caractéristiques donnera la vue optimale du diagramme de caractéristiques .Elle correspond au diagramme des caractéristiques illustré dans la figure 3.9.

Critère : Couverture en niveaux Hiérarchiques.

- Pour $HL_{\#1}$:
  - EV={f_2}
  - SEV1=EV.
  - Une seule (CoV) valide = {f_2} pour (SEV1) dans la VAC initiale (VAIC).
- Pour $HL_{\#2}$:
  - VE = {f_{11},f_{12},f_{31},f_{32},f_{41},f_{42} } .
  - CC :
    CC1 :f_{11} ← - → f_{12} (exclusive OR).
    CC2: f_{31} ← - → f_{32} (Exclusive OR)

En considérant les contraintes (CC) dans la vue (VAIC), les configurations de (f_{11},f_{12}) et (f_{31},f_{32}) dans la vue (VAIC) n'offrent pas de (VC). Ainsi, Nous pouvons seulement avoir 3 SEV à savoir : SEV1={f_{41}} avec une CoV = {f_{41}}, SEV2={f_{42}} avec une CoV ={f_{42}} et ; SEV3={f_{41},f_{42}} avec une seule (CoV) ={f_{41},f_{42}} .

- Pour $HL_{\#3}$:
  - EV={f_{321}} ; SEV1=EV
    Une (CoV) valide = {f_{321}} de (SEV1) dans la VAIC.

L'évaluation donnera les résultats groupés dans les rapports d'évaluation pour les trois niveaux hiérarchiques de la table 3.17. L'optimisation et le Mapping de la vue (VACO) en modèle de caractéristiques donnera la vue optimale du diagramme des caractéristiques illustré dans la figure 3.10.

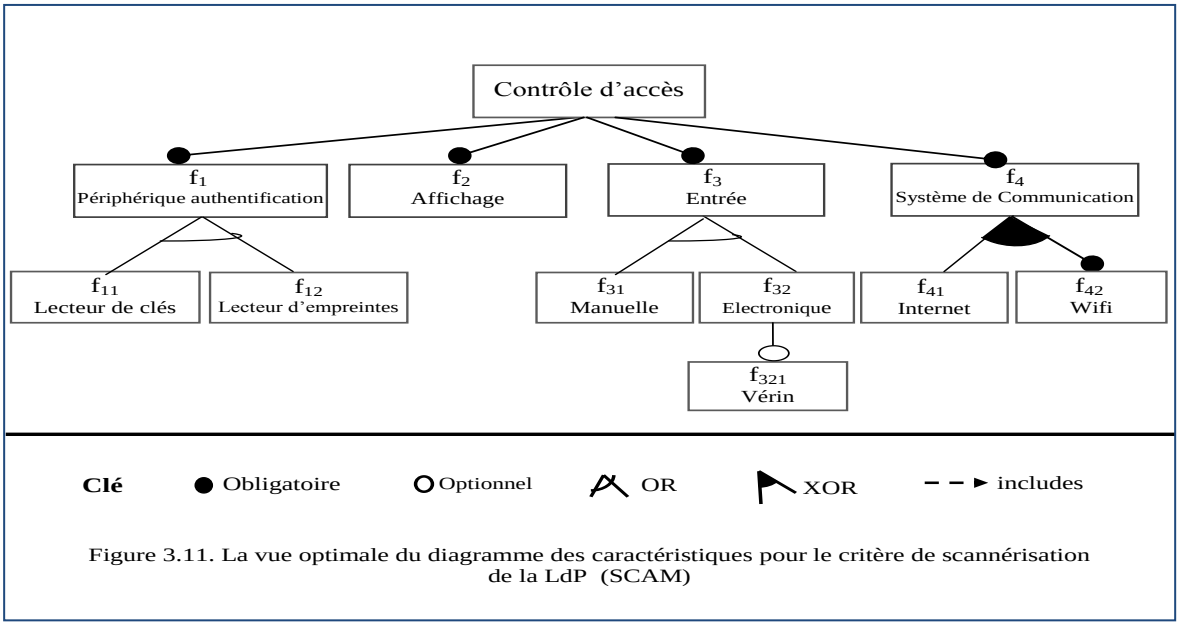


Figure 3.11. La vue optimale du diagramme des caractéristiques pour le critère de scannérisation de la LdP (SCAM)

Critère : Scannérisation .

- EV={US_1,US_2,US_3,US_4,US_5,US_6,US_7,US_8,US_9}
- Les CC:

CC1: US_1 ⟷ US_2 (exclusive OR).
CC3: US_4 ⟷ US_5.
CC4: US_4 ⟷ US_6 (exclusive OR).
CC5: US_7 ⟶ US_3 (excludes)
CC6: US_9 ⟶ US_3

- Les SEV : SEV1={US_3} , SEV2={US_5},SEV3={US_7} SEV4={US_8} , SEV5={US_9,US_3}.

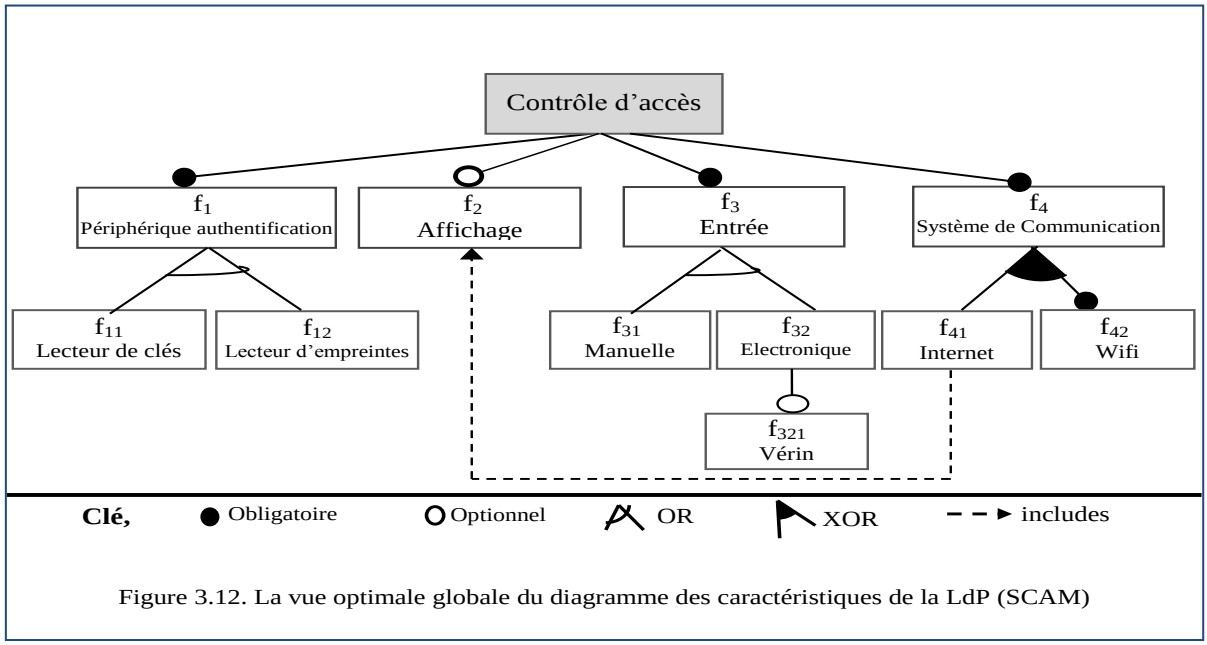


Figure 3.12. La vue optimale globale du diagramme des caractéristiques de la LdP (SCAM)

En appliquant la même sous-activité de génération à ce critère, nous obtiendrons le rapport d'évaluation montré dans la table 3.18. L'optimisation et le Mapping "Modèle d'Analyse vers le Modèle de Caractéristiques" de la vue (VACO) donnera la vue optimale du diagramme des caractéristiques montrée dans la figure 3.11.

Table 3.17 .Le rapport d'évaluation de la couverture

Couverture	ΔSP_1	ΔSP_2	ΔSP_3	MC	Max RS	Max RR
$HL_{\#1}$					1	1
Initial	1	0.75	1	3		
SEV1	1	1	1	4		
$HL_{\#1}$					0.50	0.66
Initial	0.50	0.33	0.50	3		
SEV1	0.50	0.50	0.50	4		
SEV2	0.66	0.50	0.66	4		
SEV3	0.66	0.50	0.50	5		
$HL_{\#1}$					1	1
Initial	1	0	1	3		

Table 3.18 .Le rapport d'évaluation de la Scannérisation

Scannérisation	ΔSP_1	ΔSP_2	ΔSP_3	MC	Max RS	Max RR
Initial	0.33	0	0.22	3	0.44	0.44
SEV1	0.33	0	0.33	5		
SEV2	0.22	0	0.22	4		
SEV3	0.33	0.11	0.22	6		
SEV4	0.33	0.11	0.22	5		
SEV5	0.22	0.22	0.44	4		

La vue du diagramme des caractéristiques qui en résulte correspondra au diagramme optimal global des caractéristiques de la ligne de produits. Dans notre cas, ce dernier est représenté dans la figure 3.12.

3.8 Notre Outil Support

Pour supporter l'essentiel des concepts apportés dans notre processus, nous avons implémenté un outil que nous avons appelé 'FeMoMaS (**Fe**ature **Mo**del **Ma**nagement **S**ystem)' pour Système de Management du Modèle de Caractéristiques (interface principale en figure 3.13).

Cette première version (1.0) a été développée dans un environnement orienté objet MS-Visual Basic composant de la suite MS-Studio 6 de 2003 avec des adaptations de la même version en MS-Studio 2007 supportant essentiellement une base de données (SQL) et le langage MS-VISUAL 6 avec un interfaçage en RAD.

La version FeMoMaS 1.0, disponible en français, couvre les aspects essentiels de notre processus de développement traduits selon les fonctionnalités suivantes (figure 3.13):

- Modèle de caractéristiques : renferme les données liées à la ligne de produits à créer ainsi qu'un Editeur de caractéristiques avec une Edition détaillée ou Simple du diagramme des caractéristiques ;
- Modèle d'analyse (MA) : renferme le calcul des métriques d'analyse et l'élaboration des vues d'analyse selon le critère.

- Mapping MA-MC : renferme le Mapping des Vues d'analyse en des Vues selon le Modèle de Corrélation avec calcul des métriques liées au Mapping.
- Modèle de Corrélation (MC) : renferme le calcul des métriques liées à la corrélation avec Elaboration des vues de corrélation.
- Evaluation : renferme le calcul des métriques d'évaluation et Génération des rapports d'évaluation selon le critère.
- Optimisation : renferme l'optimisation des vues de corrélation selon le critère ainsi que l'identification de la vue optimale globale.
- A propos : renferme informations générales sur la version de FeMoMaS.

FeMoMaS permet de supprimer et de mettre à jour les caractéristiques à travers son Editeur de caractéristiques. Les vues associées à l'analyse sont obtenues après une première transformation des configurations usuelles en configurations binaires.

La construction de produits en termes d'unités à gros grains est aussi permise à travers un configurateur. Ce dernier opère par la sélection et la désélection des caractéristiques en prenant en compte les contraintes de configuration.

Les vues d'analyse peuvent être obtenues également à travers les activités d'identification et de configuration selon les critères d'analyse relatifs aux deux types de granularité.

FeMoMaS permet aussi de calculer les différentes métriques nécessaires selon l'étape du processus. Notre outil est capable de transformer les vues d'analyse en vue de corrélation (VAC) en (VCC) selon le modèle de Mapping MA-MC.

Un rapport des feuilles est également établi juste après leur détection dans le diagramme des caractéristiques. Un rapport d'évaluation global par critère peut également être généré .C'est à partir de ces rapports et des vues générées ultérieurement que FeMoMaS permet de déterminer les vues optimales de la ligne de produits selon le critère et la vue optimale globale.

3.8.1 Agencement des Fonctionnalités

Dans le tableau ci-dessous, nous allons montrer l'agencement par niveau des fonctionnalités de l'outil FeMoMaS.

Le niveau 1 montre les fonctionnalités principales, le niveau 2 présente les fonctionnalités offertes après activation des fonctionnalités du niveau 1. Les fonctionnalités de niveau 3 quant à elles relèvent des fonctionnalités de niveau 2 dont elles sont directement accessibles.

Tableau 3.19. FeMoMaS : Agencement par niveau des principales fonctionnalités

Fonctionnalité principale (Niveau 1)	Niveau 2 de fonctionnalités	Niveau 3 de fonctionnalités
Modèle de caractéristiques		
	Ligne de Produits	
	Editeur de caractéristiques	
	Navigation dans diagramme des caractéristiques	
		Caractéristique fille
		Caractéristique Parent
		Diagramme des caractéristiques initial
	Configurateur du Modèle	
		Simple Configuration
		Configuration Binaire
Modèle d'analyse		
	Métriques	
	Générateur de Vues d'analyse	
	Editeur vues d'analyse	
	Identificateur du modèle	
		Cadre de Variabilité
		Configuration Variable
		Sous-cadre de Variabilité
		Contraintes de Configuration
	Configurateur du Modèle	
Mapping MA-MC		
	Métriques	
	Générateur de vues de Corrélation	
Modèle de Corrélation		
	Editeur de vues de corrélation	
	Métriques	
Evaluation		
	Préparation des Rapports d'Evaluation	
		Métriques d'Evaluation selon le critère
		Métriques d'évaluation globale
	Rapports d'évaluation selon le critère	
Optimisation		
	Métriques	
	Vues Optimales	
	Mapping VCCO-VACO	
	Editeur de vues Optimales	
Tables		
	Critères	
	Types de Variabilité	
	Types de Relation	
A propos		

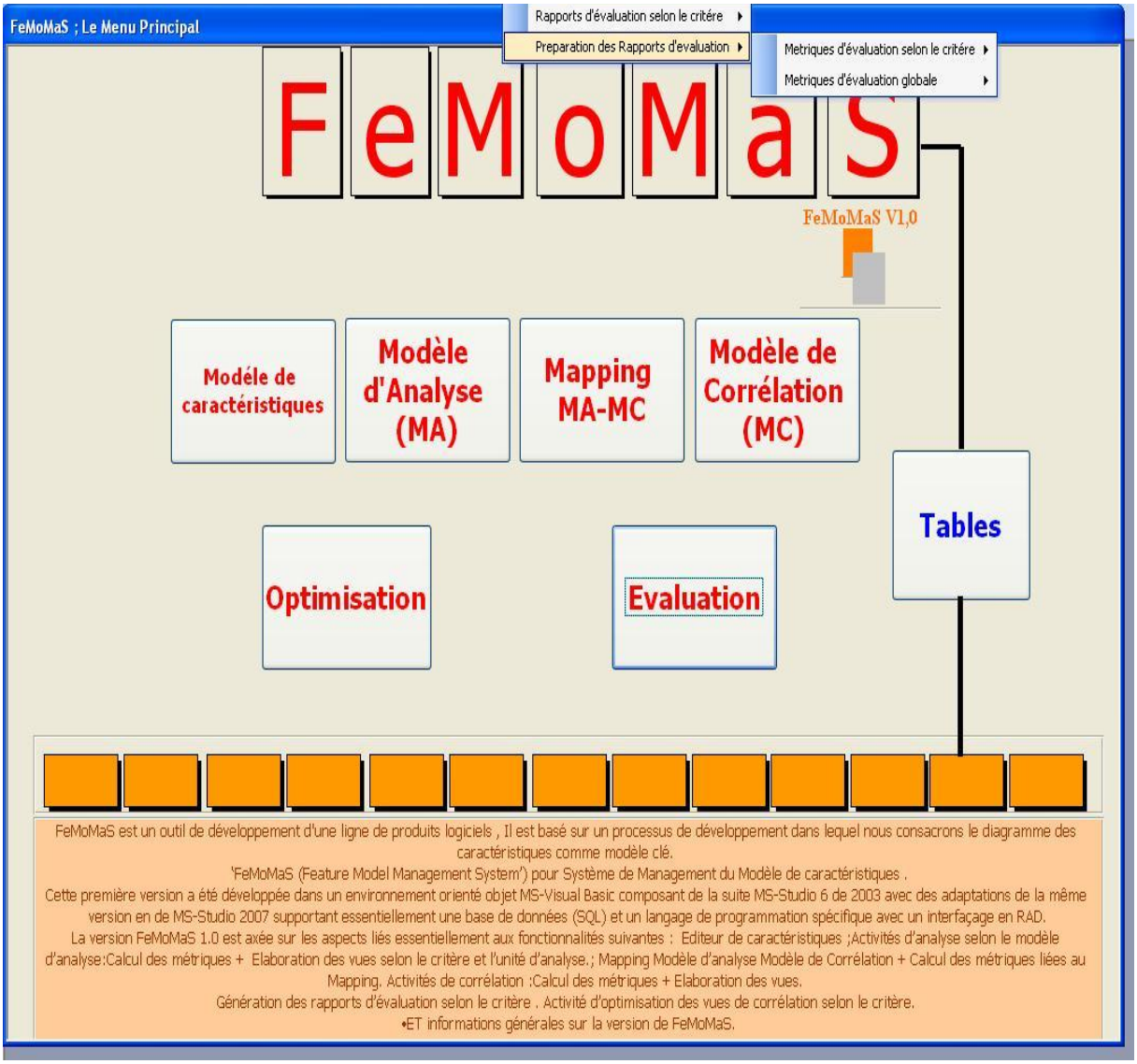
Cependant, les limitations de FeMoMaS concernent l'inaptitude de composer et d'intégrer les diagrammes de caractéristiques. Ceci est dû au fait que notre outil n'a pas été conçu pour supporter les formalismes graphiques.

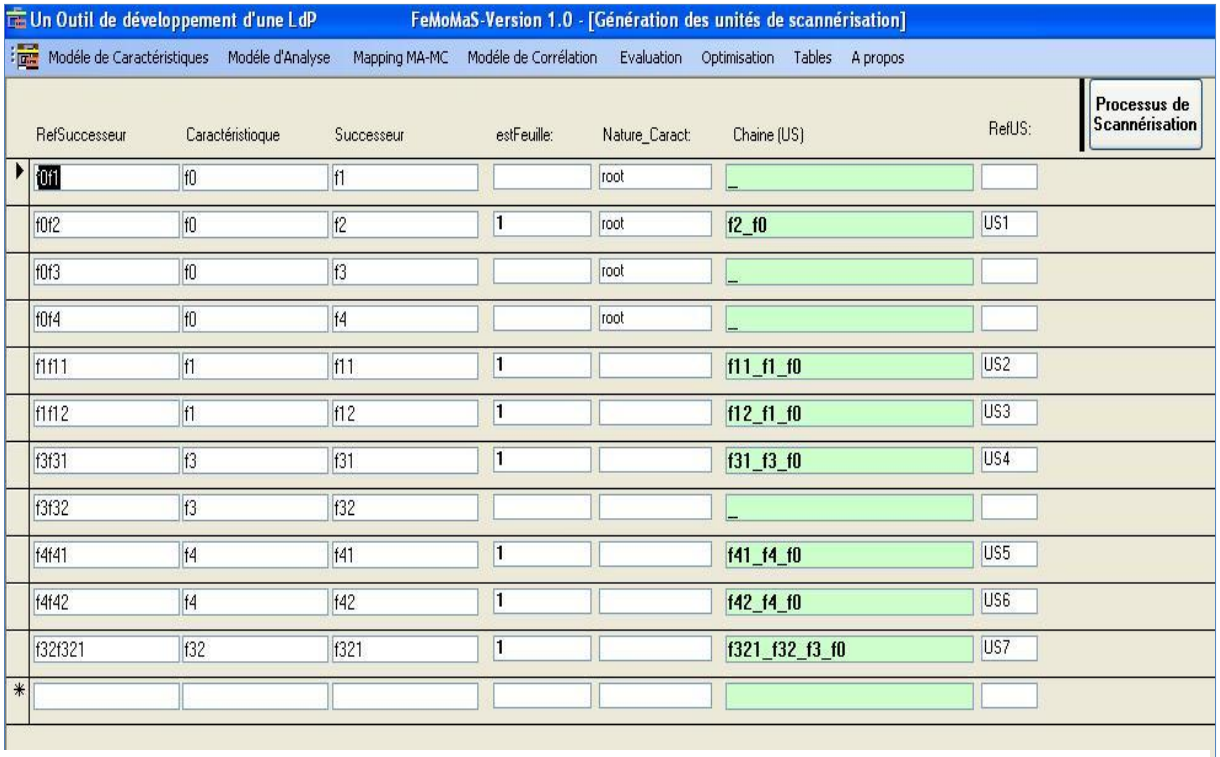
3.8.2 Les Interfaces

Nous allons montrer dans cette section à travers des captures d'écran de FeMoMaS deux interfaces utilisateurs de notre outil à savoir :

- *L'interface Principale de l'outil FeMoMaS*
- *Un Aperçu sur le mécanisme de scannérisation et de génération des (US).*

Figure 3.13. Interface Principale de notre outil FeMoMaS





RefSuccesseur	Caractéristioque	Successeur	estFeuille:	Nature_Caract:	Chaine (US)	RefUS:
f0f1	f0	f1		root	_	
f0f2	f0	f2	1	root	f2_f0	US1
f0f3	f0	f3		root	_	
f0f4	f0	f4		root	_	
f1f11	f1	f11	1		f11_f1_f0	US2
f1f12	f1	f12	1		f12_f1_f0	US3
f3f31	f3	f31	1		f31_f3_f0	US4
f3f32	f3	f32			_	
f4f41	f4	f41	1		f41_f4_f0	US5
f4f42	f4	f42	1		f42_f4_f0	US6
f32f321	f32	f321	1		f321_f32_f3_f0	US7

Figure 3.14. Aperçu du mécanisme de scannérisation et de génération des (US)

3.9 Conclusion

Dans ce chapitre de la thèse, nous avons ciblé une problématique spécifique au domaine des lignes de produits logiciels liée à la conception de modèles de développement performants. Parvenir à réaliser de tels modèles, représente un enjeu majeur pour renforcer la qualité des logiciels produits, notamment dans le secteur de l'embarqué qui est amené à être utilisé à large échelle surtout dans les systèmes critiques. Cet état de fait est dû à l'aspect générique des modèles de la ligne de produits et à la structure de leur ingénierie.

Pour parfaire cette insuffisance, nous avons proposé une démarche basée sur un processus de développement qui cherche à concevoir un diagramme de caractéristiques optimal pour la ligne de produits à partir d'un ensemble restreint de produits similaires dans un domaine. Notre processus emploie un ensemble de six modèles (variabilité, analyse, corrélation, Mapping, évaluation et intégration) ainsi que des métriques générées graduellement lors de ses différentes étapes.

Le modèle de variabilité formule la structure d'un produit en termes de présence de la caractéristique dans la composition du produit. Le modèle d'analyse quant à lui, sert à examiner la structure interne d'un produit en termes d'unités d'analyse selon le critère. A cet effet, ce modèle emploie quatre critères de différentes granularités (gros grains et grains fins) et quatre unités d'analyse pour calculer les métriques qui lui sont appropriées à savoir : la

caractéristique, la feuille (par rapport au graphe des caractéristiques), le niveau hiérarchique et l'unité de scannérisation. Une autre activité est effectuée dans le cadre de ce modèle, il s'agit de la configurabilité des produits. Cette dernière est scindée en deux étapes :

- L'étape d'identification, permet de préparer les outils qui seront directement utilisés lors de l'étape de configuration. Il s'agit des cadres de variation avec l'ensemble de leurs sous-cadres de variation valides en plus des cadres et des contraintes de configuration.
- L'étape de configuration, permet de générer les configurations valides des produits en s'appuyant sur les outils précédemment crées lors de l'étape d'identification.

En vue de doter notre démarche de plus de robustesse, nous ne nous sommes pas contenté d'évaluer les configurations générées en termes de métriques selon un seul modèle (le modèle d'analyse) incluant pourtant quatre critères, mais nous employons trois autres modèles : le modèle de corrélation , le modèle de Mapping et le modèle d'évaluation.

Le modele de corrélation perçoit la structure interne d'un produit comme un ensemble de quatre compartiments .Ce modèle engage les métriques d'analyse selon le critère et non la configuration proprement dite du produit ainsi que le positionnement de ces métriques (les ratios selon le critère) par rapport à l'unité d'analyse du produit dans le modèle de corrélation.

Cet apport permet de renforcer l'usage des métriques d'analyse déjà obtenues et vient consolider le modèle d'analyse dans l'étape d'évaluation par un deuxième jeu de métriques liées à une autre vision de la configuration du produit.Toutefois, compte tenu de la complexité et de l'importance de cette dernière activité, la définition d'un Mapping qui transforme les données résultant du modèle d'analyse en structures du modèle de corrélation est devenue nécessaire. L'étape d'évaluation compte à elle, s'effectue à partir des métriques selon le modèle de corrélation des configurations du produit organisées sous forme de vues de corrélation. Elles concernent la similarité, la réutilisabilité, la commonalité et la différentiabilité. L'évaluation est sanctionnée par un rapport d'évaluation sous forme de table selon le critère. L'objectif principal de l'évaluation dans notre démarche, en plus de l'estimation des configurations selon le critère des produits, est de faciliter l'étape qui la suit , à savoir l'optimisation des vues obtenues , en lui fournissant les rapports des configurations de corrélation en termes de métriques Notre démarche fournit une vue optimale sous-forme d'un modèle de caractéristiques selon le critère ainsi qu'une vue optimale globale sous la même forme mais pour toute la ligne de produits.

De plus, il est à signaler que durant le déroulement du processus, des ensembles de vues et de modèles de caractéristiques selon le critère d'analyse sont fournis systématiquement à chaque étape. Cette possibilité offre aux parties prenantes un maximum de perspectives, sous forme

de vues de la ligne de produits à toute étape. Ceci confère à notre approche un aspect multi-perspectives typique aux conceptions performantes de lignes de produits.

Nous avons appuyé notre processus par un outil-support propre (FeMoMaS) qui dans sa première version, permet de prendre en charge les principales activités de la démarche.

Enfin, et pour montrer les points forts de notre processus de développement, nous l'avons testé dans le domaine des maisons intelligentes et plus précisément le système de contrôle d'accès.

Deuxième Partie

Contributions

Chapitre IV

Modélisation de l'Évolutivité de la variabilité à partir d'une adaptation du rôle UML aux Lignes de Produits

4.1 Introduction

Pour une stratégie ligne de produits, le concepteur doit parfois prendre en compte des exigences spécifiques émises par des utilisateurs particuliers qui n'ont pas été prévues lors de la conception du domaine. Dans la pratique et selon [45],[61], envisager des modèles évolutifs pour la ligne de produits demeure la solution la plus préconisée.

Dans la littérature ,l'évolutivité d'une ligne se faisant primordialement à partir des caractéristiques [13],[39],[42], les solutions les plus significatives pour prendre en charge ce type de préoccupations se sont orientées vers deux directions : dans la première , il s'agit de concevoir puis cogérer un modèle d'extensibilité des caractéristiques en même-temps que le modèle originel des caractéristiques de la ligne [6],[81],[68]; ou bien prévoir une modélisation évolutive des caractéristiques dans un seul modèle [10],[30].

Dans notre étude, nous avons proposé une modélisation qui enlace la seconde voie à travers un modèle d'évolution de la variabilité basé sur des rôles évolutifs. Ces derniers sont une représentation à base d'une adaptation du rôle UML (dédié initialement aux systèmes uniques) au contexte multi-systèmes des lignes de produits.

Pour parvenir à une telle conformité, au lieu du rôle UML tel que défini, nous proposons l'intronisation à UML de deux nouvelles associations désignées « **role** » et « **evolution** » à travers lesquelles nous représentons respectivement la variabilité liée aux lignes de produits et l'évolution des éléments variables.

L'association « **role** » permet de représenter les caractéristiques variables à travers ce que nous avons appelés les '**f-roles**' (pour **f**eature-**role** ou caractéristique-role) ou ' **rôles évolutifs'** et l'association « **evolution** » permet de représenter leurs évolutions par des mécanismes introduits conjointement dans le diagramme de classes et la machine à états. Chaque élément variable (point de variation et ses variants) peut évoluer en jouant ou en cessant de jouer un ou plusieurs f-roles et chaque f-role peut aussi évoluer en jouant ou en cessant de jouer d'autres f-roles.

Les éléments variables et les f-roles sont représentés sous forme de classes dans le diagramme de classes. Ces derniers peuvent sous certaines conditions (**conditions d'évolution** ou contraintes) évoluer à leur tour pour jouer d'autres f-roles.

L'association « role » permet également aux classes f-roles d'hériter leur classe-mère (la classe représentant l'élément variable).

Toutefois et pour des problèmes de visibilité, toutes les conditions d'évolution ne peuvent être représentées dans le diagramme de classes . Pour pallier cette difficulté , nous avons d'abord

scindé le diagramme de classes en partitions que nous avons appelées « **partitions variables évolutives** » .Chaque partition porte le nom de l'élément variable qu'elle modélise. Elle représente une composante connexe du modèle de caractéristiques considéré comme un graphe avec comme racine l'élément variable dont elle porte le nom. La partition est à son tour divisée en deux compartiments. Le premier représente une partie du diagramme de classes composée d'un élément variable et de tous ses f-roles actuels et potentiels. Pour mieux exprimer les **contraintes d'évolution,** nous avons associé dans un second compartiment « une machine à états » spécifique, équivalente aux éléments du 1er compartiment de chacune des partitions variables évolutives qui montre les évolutions possibles avec toutes les contraintes d'évolution.

Pour compléter le niveau d'information des conditions d'évolution, un listing des contraintes d'acquisition et de cession des f-roles par les instances de chaque classe évolutive peut être joint à ce modèle. Ces contraintes peuvent être exprimées sous la forme de prédicats dans le langage formel OCL [89].

Pour faciliter la lecture et l'exploitation des partitions variables, les deux compartiments représentant la variabilité et son évolution décrites précédemment sont regroupés dans un nouveau modèle que nous avons appelé « **modèle d'évolution** ».

Outre la représentation de la variabilité à travers les variables et de son évolution à travers les f-roles, un autre avantage de cette modélisation est la séparation entre les caractéristiques variables et leurs évolutions dans un seul modèle. Cette représentation permet également une meilleure visibilité des conditions d'évolution 'élément variable-f-role' et 'f-role-f-role'. De plus, nous avons ajouté deux autres scénarios d'évolution plus complexes par rapport aux scénarios classiques d'acquisition et de cession de rôles définis pour le rôle UML actuel, il s'agit de la **Mutation** du f-role vers un autre f-role qui définit une acquisition d'un nouveau f-role avec cession d'un ancien f-role ou de plusieurs f-roles , et le **Prolongement** défini comme une acquisition d'un nouveau f-role avec conservation d'un ancien f-role particulier ou de plusieurs f-roles. Cet apport permet d'améliorer les mécanismes de flexibilité de la ligne de produits.

Notre modèle d'évolution peut être perçu à travers deux vues, une vue globale du modèle et une autre partielle à travers les deux compartiments des partitions variables évolutives. De plus, ces deux vues sont consultées à travers un instantané de l'état courant du modèle d'évolution selon la vue. Ce dernier montre les deux compartiments des partitions variables en termes de classes variables et leurs f-roles évolutifs dans le premier compartiment et les états, leurs sens d'évolution avec les contraintes d'évolution en plus des f-roles évolutifs dans

le compartiment équivalent. Cette étude est une extension du sous-modèle d'expression de la variabilité par les rôles que nous avons réalisé préalablement dans [117]. Pour rappel , ce dernier est une vision multi-système des objets migrants approchés par [49].

Dans ce chapitre, nous nous attachons à décrire tous les mécanismes et concepts de notre modèle précédemment abordés. Les avantages de notre modèle seront montrés à travers une illustration dans le domaine des moteurs automobiles et plus particulièrement « le système de calcul de la consommation en fuel ». Un résumé de nos apports en contributions est relaté dans la section qui conclut le chapitre.

4.2 Modèlisation de l'évolutivité

Nous allons décrire dans cette section, les concepts et les mécanismes de notre modèle, liés à l'évolutivité de la variabilité à partir des rôles UML conformés aux lignes de produits.

4.2.1 Aperçu de notre Modèle

Dans ce qui suit , nous allons examiner l'expression de l'évolution de la variabilité à travers notre modèle [46],[47] qui utilise le concept de rôle UML revu et adapté au problème des lignes de produits logiciels. A travers notre modèle, la ligne de produits est perçue non plus comme un ensemble de caractéristiques variables mais tel un ensemble d'éléments variables qui peuvent évoluer en jouant ou en perdant des rôles tout en gardant ou en cédant ceux déjà procurés , que nous avons appelé "f-roles" ou rôles évolutifs (figure 4.1). La variabilité est

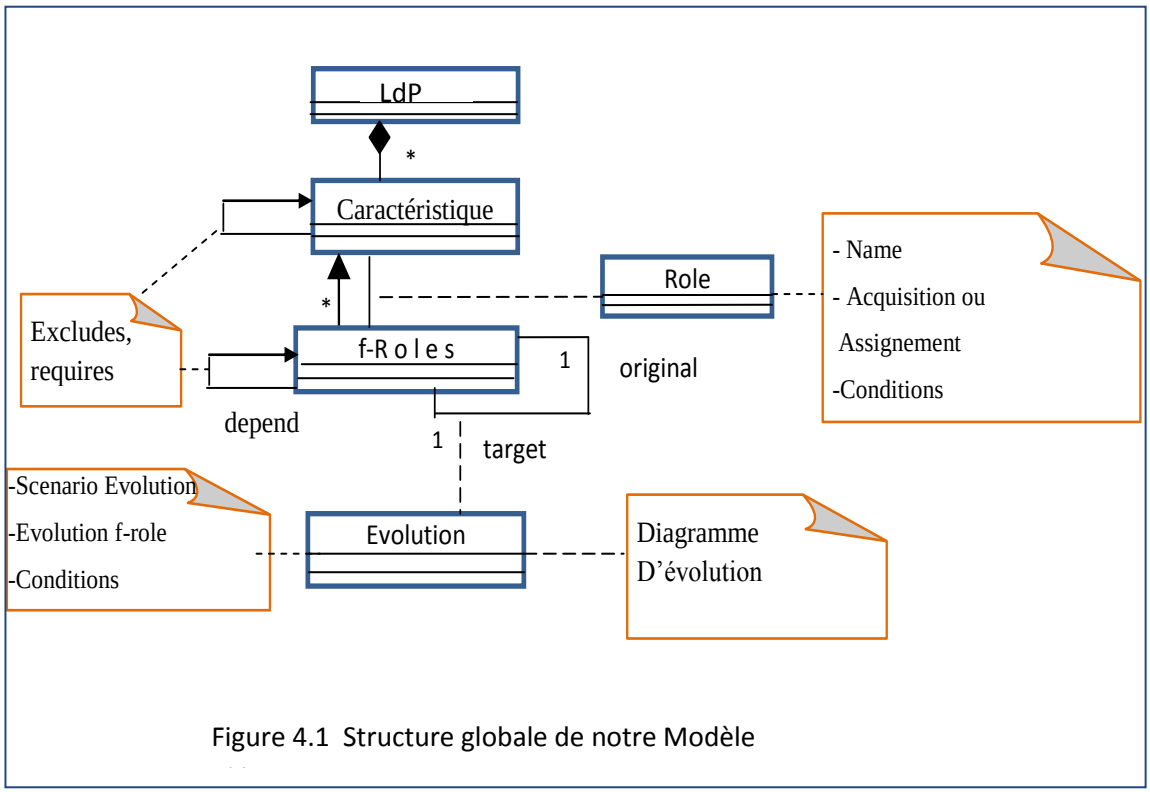


Figure 4.1 Structure globale de notre Modèle

exprimée à travers les caractéristiques variables qui correspondent soient à des points de variation obligatoires optionnels ou alternatifs ou dans certains cas (optionalité) des variants.

La relation entre chaque caractéristique variable et ses formes évolutives les f-roles, est réalisée à l'aide d'une nouvelle association que nous avons appelé association « role ». L'évolution des f-roles est obtenue par une autre association que nous avons appelée association « evolution » .

La figure 4.2 compare une représentation de la perception d'une ligne de produits entre le modèle classique utilisé dans la majorité des techniques liées aux lignes de produits et notre modèle.

Dans notre modèle, les caractéristiques variables sont représentées comme des super-classes et leurs variants ainsi que leurs évolutions (f-roles) comme des sous-classes.

L'association "role" permet aux sous-classes évolutives 'f-roles' d'hériter leurs classes mères.

Le mécanisme de dépendance entre les f-roles est le même utilisé dans UML.

Notre modèle permet à un « **f-role source** » ou « **original f-role** » sous certaines conditions

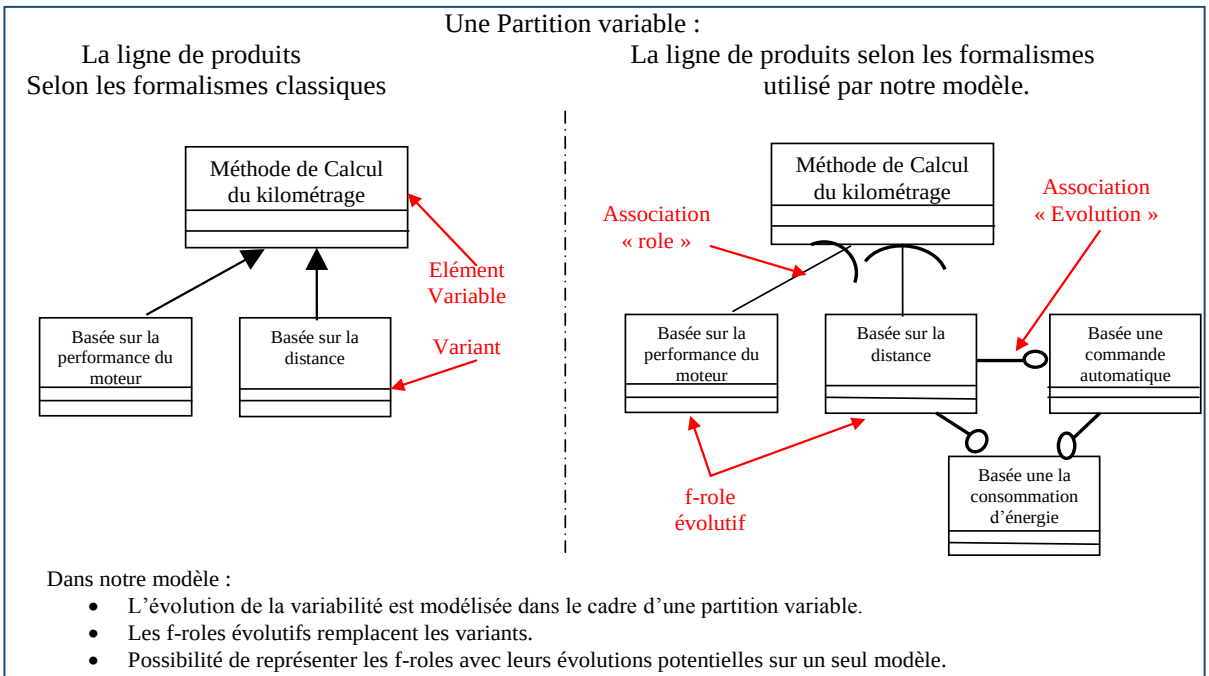


Figure. 4.2. Comparaison de la représentation de la ligne de produits entre notre modèle et le modèle classique

d'évoluer (acquérir ou perdre des f-roles en gardant ou en cédant les f-roles obtenus précédemment) vers un autre f-role: le « f-**role cible** » ou « **target f-role** ».

Cette évolution peut être modélisée dans le cadre d'une partition variable évolutive selon deux représentations:

- Une représentation à l'aide de formalismes utilisés dans le diagramme de classes pour les deux concepts : caractéristique et f-role.

- Une représentation qui fait ressortir les évolutions, les dépendances entre tous les éléments variables (et leurs variants) évolutifs ainsi que les contraintes d'évolution. Cette représentation est réalisée à base d'une machine à état-transitions.

4.2.2 Les Eléments de notre Modèle

Dans cette section, nous allons définir les formalismes des concepts employés par notre modèle et les règles qui permettent de les administrer.

Le premier cadre de représentation repose sur l'ensemble des formalismes utilisés dans un modèle statique : le diagramme de classes. Nous avons opté pour cette représentation pour

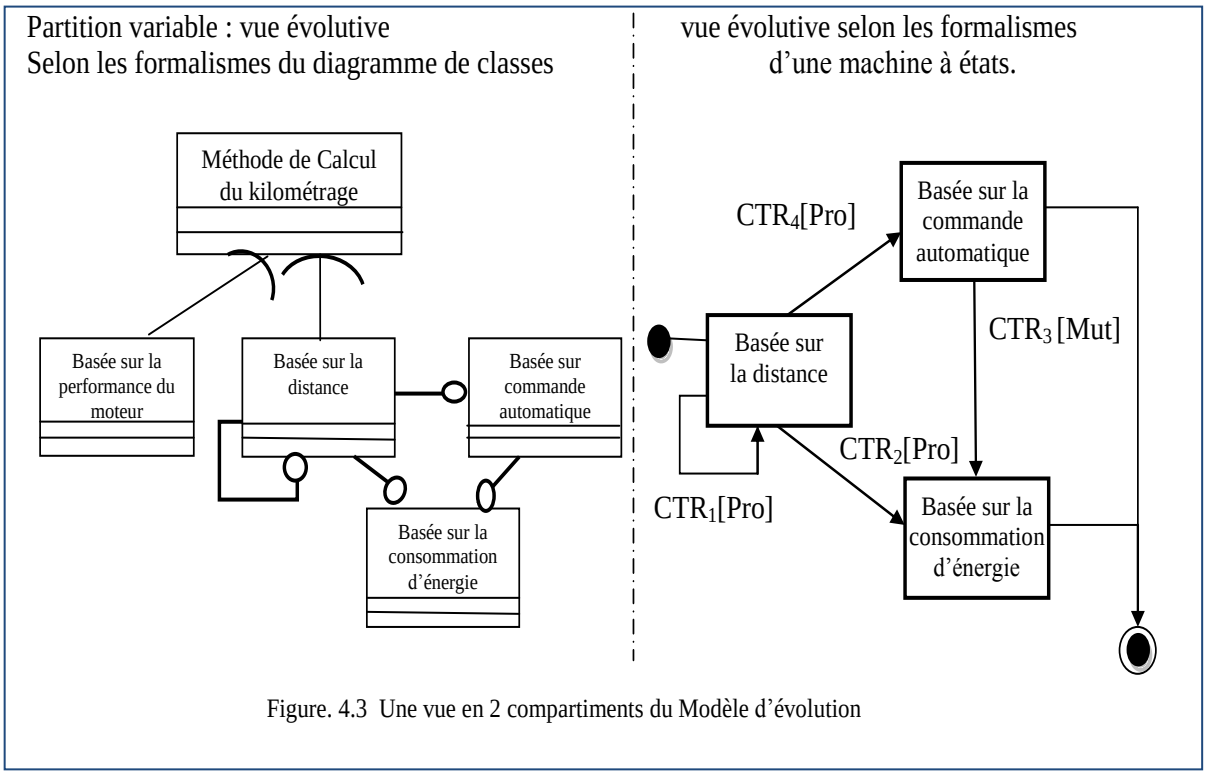


Figure. 4.3 Une vue en 2 compartiments du Modèle d'évolution

deux raisons: la première est liée à la grande similarité entre la modélisation de la caracteristique dans le modéle de caracteristiques et celle de la classe UML dans les principaux travaux basés sur les modéles de caracteristiques pour introduire la variabilité dans le domaine d'analyse [51],[65] ; la deuxieme est beaucoup plus liée au fait que le diagramme de classes est la vue principale pour le langage UML .C'est un diagramme incontournable dans toute étude à base d'UML. De plus , Il montre la structure interne du système et peut offrir une représentation résumée des objets du système qui interagissent ensemble. Ses éléments principaux sont les classes et leurs relations (les associations, notamment la généralisation, ainsi que de nombreux types de dépendance).

Le deuxième cadre de représentation se base sur un ensemble de formalismes et de règles de notre modèle utilisés cette fois dans un modèle dynamique: la machine à états.

Le troisième cadre offre des formalismes et des règles qui permettent d'associer les deux premiers cadres.

4.2.2.1 Les formalismes du diagramme de classes et les régles associées appliqués aux concepts introduits dans le 1er compartiment

a. Les formalismes

Les formalismes proposés qui permettent une adaptation du rôle UML aux lignes de produits dans notre modèle statique sont à base de deux nouvelles associations: l'association "role" et l'association « evolution ». Ces deux associations en plus d'autres concepts introduits et un certain nombre de règles, permettent de représenter la variabilité évolutive .Ce sont ces formalismes que nous allons décrire dans cette section.

- Les points de variation sont représentés sous forme de classes dites classes variables.
- Les variants des classes variables sont représentés également sous forme de classes.

Ce sont les classes filles de la classe mère variable.

- Les f-roles représentent la variabilité des classes variables (variants , dans un modèle classique) ainsi que la variabilité évolutive des classes f-roles qui lui sont associées (les f-roles évolutifs).
- Les f-roles sont représentés sous forme de sous-classes (dans la figure 4.2) de la classe variable associée ou de la classe mère f-role.
- Les classes "f-role" héritent les propriétés et les opérations de leurs classe mère.
- Tous les f-roles et évolutifs doivent être représentés par les formalismes utilisés dans le diagramme de classes.

b. Les Régles

- Dans notre modèle, nous disons qu'une classe joue un f-role si elle évolue vers un f-role de même nom que ce f-role et nous disons qu'elle cesse de jouer ce f-role si elle cède ce f-role. Dans ce dernier cas, la classe f-role sera conservée dans le modèle mais ne sera plus liée à sa classe mère.
- Un f-role est soit optionnel, soit obligatoire. Les alternatives sont considérées comme des f-roles.
- Un f-role est dit optionnel lorsqu'il est joué (résultat d'une évolution) c.à.d. si une condition d'évolution au moins est satisfaite et il cesse d'être joué si la même condition et pas une autre n'est plus satisfaite. Le « f-role optionnel » est représenté par une classe dotée en plus de la dimension « 0,1 » [40].

- Le f-role obligatoire est un f-role qui doit toujours être joué. Il ne dépend d'aucune condition imposée par la classe mère. Il ne dépend d'aucun autre f-role. La classe qui correspond au f-role obligatoire est identifiée par les formalismes du diagramme de classes avec la dimension [1.1] [40].
- Les instances des classes variables évoluent en jouant ou en cessant de jouer d'autres f-roles selon certaines conditions.
- Les instances des classes f-roles évoluent en jouant ou en cessant de jouer des f-roles selon certaines conditions.
- Les conditions d'évolution sont appelées contraintes .Elles sont notées (CTRi). Elles peuvent être formalisées séparément dans un état associé au modèle d'évolution à l'aide du langage des contraintes objet OCL [89].
- Une instance d'une classe f-role peut évoluer selon deux f-roles différents de la même classe f-role .Ce principe s'appelle la multi-instanciation .Exemple : la "*méthode basée sur la distance*" peut être utilisée avec deux différentes énergies : « *l'essence* » et « *le gasoil* » (figure 4.3).
- La prise en compte des conditions d'évolution , dans la partition variable , selon les formalismes du diagramme de classes se traduit à travers les formalismes illustrés dans la figure 4.3 (1er compartiment). Dans ce formalisme, (Ctri) sont les conditions selon lesquelles,
  - Les classes variables évoluent vers des f-roles
  - Les classes f-roles évoluent vers d'autres f-roles.

4.2.2.2 Les formalismes de la machine à etats et les régles associées appliqués aux concepts utilisés dans le 2eme compartiment

a. Les Formalismes

- Le deuxième compartiment , utilise une machine à états-transitions pour représenter l'évolution des éléments variables du 1er compartiment de la partition sous forme de f-roles correspondant aux classes variables et qui montre également l'interaction inter-f-roles d'une même classe variable.
- Les états représentent les f-roles ;
- Les transitions libellées (CTRi) indiquent les contraintes auxquelles sont soumises les instances de la classe variable ou de la classe f-role pour pouvoir évoluer ;
- Un état initial et un état final sont ajoutés pour parachever le formalisme de la machine ;

- Les conditions d'évolution (CTRi) proprement dites sont formalisées par l'emploi du langage UML de déclaration des contraintes objet OCL [89].
- Les contraintes d'intégrité (de dépendance) :

Ce type de contrainte est obtenu par une évolution unidirectionnelle de l'élément variable vers un f-role ou d'un f-role vers un autre f-role. Les flèches unidirectionnelles est le formalisme employé pour montrer les dépendances.

- Les transitions :

Une transition est définie par :

– Un événement déclencheur.

– Une condition (contrainte d'évolution) garde de la transition.

– Une action exécutée (un scénario d'évolution) lorsque la transition est tirée.

b. Les Règles

- L'évolution est réalisée soit à partir d'un f-role source vers un f-role cible, soit d'un élément variable vers un f-role.
- Un f-role concerné par ce type de contraintes d'évolution sera supprimé (aura cessé d'exister) si le f-role source ou l'élément variable est supprimé.
- Le f-role cible doit cesser d'exister si le f-role source est supprimé.
- Le f-role cible ne peut être créé sans que le f-role source l'ait été auparavant. Cette situation est modélisée par le formalisme de la figure 4.4.

4.2.2.3 Formalismes et Mécanismes des Concepts utilisés dans notre Modèle d'évolution

Les formalismes du diagramme de classes seuls et ceux de la machine à états seuls ne sont pas suffisants pour représenter convenablement plusieurs cas d'évolution à la fois et notamment pour la même classe (comme le montre la figure 4.3).

Essayer de représenter cet aspect de l'évolution séparément dans chaque compartiment selon les formalismes proposés donnera une partition variable surchargée et difficile de lecture et une machine à états qui ne distingue pas entre les éléments variables et les f-roles.

Associer les deux formalismes à travers un modèle permettra sans aucun doute d'y pallier. C'est ce que nous avons proposé. Nous l'avons appelé le « **Modèle d'évolution** ».

a. Les Formalismes

Le modèle d'évolution est un ensemble de partitions variables évolutives représentant chacune une composante connexe du modèle de caractéristiques (une branche d'un graphe) formalisé en graphe.

Elle regroupe chacune deux représentations parallèles organisées en deux compartiments.

Dans le premier compartiment, nous trouvons le sommet de la composante connexe sous forme d'une classe, ses f-roles initiaux c.à.d. à l'étape de la modélisation du domaine (les variants initiaux) ainsi que l'évolution de ces f-roles mais sans les contraintes d'évolution comme illustré dans la figure 4.8.

Dans le deuxième compartiment, nous trouvons les éléments variables et les f-roles concernés par l'évolution qui ont été définis dans le 1er compartiment sous forme d'états-transitions en plus des conditions d'évolution et de dépendance mais avec les formalismes d'une machine à états. Ainsi, les formalismes introduits dans le diagramme de classes auront leurs équivalents dans le compartiment de la machine à états. De ce fait, les formalismes proposés du modèle d'évolution sont basés sur la transformation des formalismes du 1er compartiment vers ceux du deuxième et ce dans une seule et unique partition variable évolutive .Le tableau 4.1 regroupe la représentation des principaux cas de dépendance dans le modèle d'évolution.

Les formalismes de l'évolution considérés dans le premier compartiment se présentent comme suit :

- L'association « role » liée aux classes f-roles pour modéliser les variants sous forme de f-roles.
- L'association « evolution » concerne l'évolution des f-roles précédemment définis.

Les formalismes de l'évolution pris en charge dans le deuxième compartiment sont les suivants : une machine à états-transitions pour représenter l'évolution des classes variables

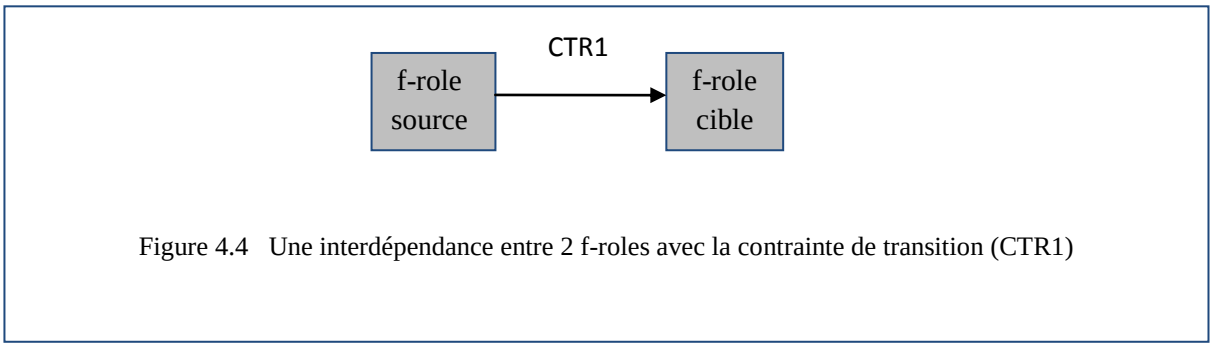


Figure 4.4 Une interdépendance entre 2 f-roles avec la contrainte de transition (CTR1)

et des classes f-roles ainsi que les f-roles qui montrent l'interaction inter-f-roles d'une même classe variable.

Nous nous sommes focalisés dans notre modèle sur les machines à états sans hiérarchie et sans orthogonalité.

b. Les mécanismes

En plus des formalismes déjà cités, le modèle d'évolution s'appuie sur deux nouvelles associations que nous avons proposé à savoir, l'association « role » et l'association « evolution ».

Nous allons définir dans cette section, les mécanismes proposés en plus des deux associations « role » et « evolution » .

1. Les types de classes f-role

Les types de classes représentant les f-roles sont les suivants :

Obligatoire, Optionnel ou à Alternatif.

- Pour l'optionalité (type optionnel).

Les superclasses concernées par l'optionalité ont une dimension égale à (0,1). Ceci représente dans un diagramme de classes classique une association entre la classe mère et la classe optionnelle de cardinalités : 1 ⟶ 0,1. Le 1 du côté de la classe mère et les (0,1) du côté de la classe optionnelle.

Ces classes ont un seul f-role ou pas de f-role du tout et ne peuvent évoluer vers d'autres f-roles. Elles sont représentées à travers notre modèle d'évolution dans la figure 4.5.

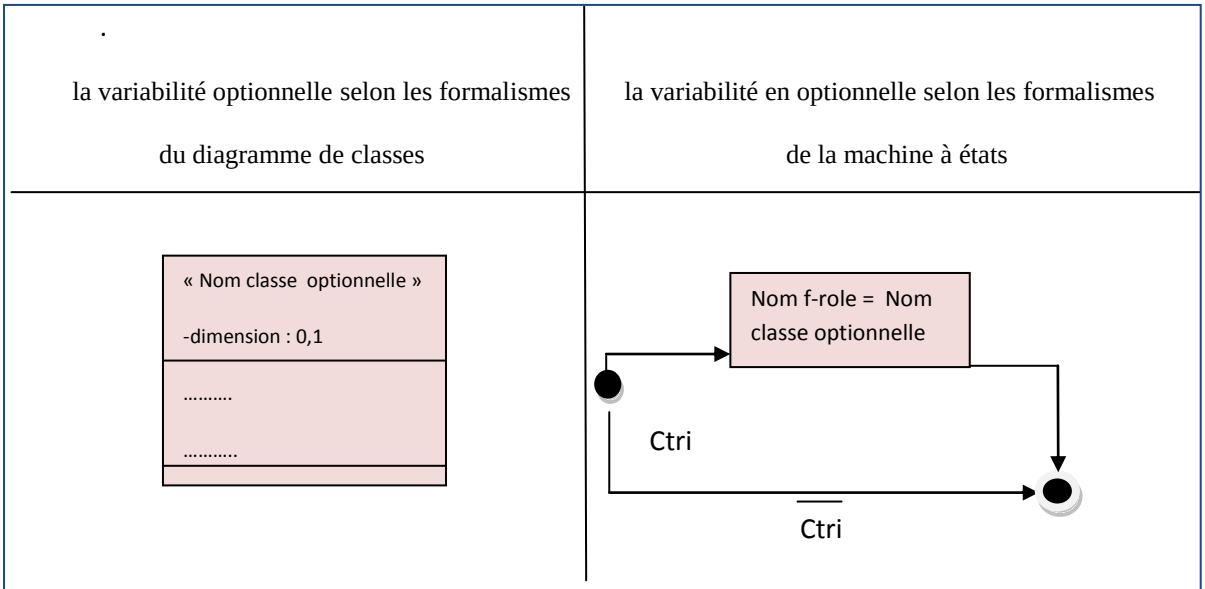


Figure 4.5. Transformation des formalismes « Diagramme de classes – Machine à états » pour une classe optionnelle.

- Pour le type alternatif

Les f-roles sont représentés sous forme de classes associées à la classe variable .Cette dernière peut avoir des sous-classes f-roles obligatoires ou alternatives. Ces deux types sont représentés sous forme de classes f-role liées à la classe variable.

2. Les transformations d'évolution

Nous avons regroupé les principales transformations des formalismes 'Diagramme de Classes – Machine à Etats' modélisant l'évolution des f-roles dans le tableau 4.2.

A noter que l'emboitement des formalismes utilisés à travers des regroupements est possible dans le compartiment des états transitions du modèle d'évolution.

3. Identification des partitions variables

Nous allons d'abord, définir la partition variable puis nous décrirons la méthode utilisée pour l'identifier dans le modèle des caractéristiques de la ligne de produits.

- **Définition**

La partition variable est déterminée à partir du modèle des caractéristiques de la ligne de produits puis transformée selon les formalismes du diagramme de classes dans le 1er compartiment du modèle d'évolution.

Une partition variable est une composante connexe du modèle des caractéristiques si l'on considère ce dernier comme un graphe.

La partition variable peut être définie comme un ensemble de sommets qui ont deux à deux la relation de connexité.

De plus tout sommet en dehors de la composante connexe n'a pas de relation de connexité avec les sommets de cette composante.

Elle peut être définie également comme un chemin simple et élémentaire composé d'un ensemble d'arcs orientés dans le graphe du modèle des caractéristiques.

Ce chemin parcourt la surface du modèle des caractéristiques du niveau hiérarchique le plus haut vers le niveau hiérarchique le plus bas avec la mémorisation des arcs orientés rencontrés pendant ce parcours et ce jusqu'à la rencontre de la première feuille .

- **Méthode d'Identification**

Le processus d'identification des partitions variables explore toutes les alternatives offertes en termes d'arcs orientés dans le graphe des caractéristiques.

Pour identifier une partition variable que nous noterons (Ci) dans le modèle des caractéristiques, nous avons besoin :

- De considérer le modèle des caractéristiques comme un graphe sans racine.
- De connaitre les feuilles (caractéristiques sans successeurs) contenues dans le graphe des caractéristiques sans racine.
- De définir les ensembles de sommets qui ont deux à deux la relation de connexité.
- De Vérifier que tout sommet en dehors de la composante connexe n'a pas de relation de connexité avec les sommets de cette composante.

- **Application.** Exemple de définition des composantes connexes dans le modèle des caractéristiques de la ligne de produits «*Contrôle d'Accès à une Maison Intelligente*». Après application de la méthode d'identification de ces composantes (Ci) , nous obtenons les résultats illustrés dans la figure 4.6. Les ensembles résultats sont les suivants :

$C_1=\{f_1,f_{11},f_{12}\}$,
$C_2=\{f_2\}$,
$C_3=\{f_3,f_{31},f_{32},f_{321}\}$ et
$C_4=\{f_4,f_{42},f_{41},f_2\}$.

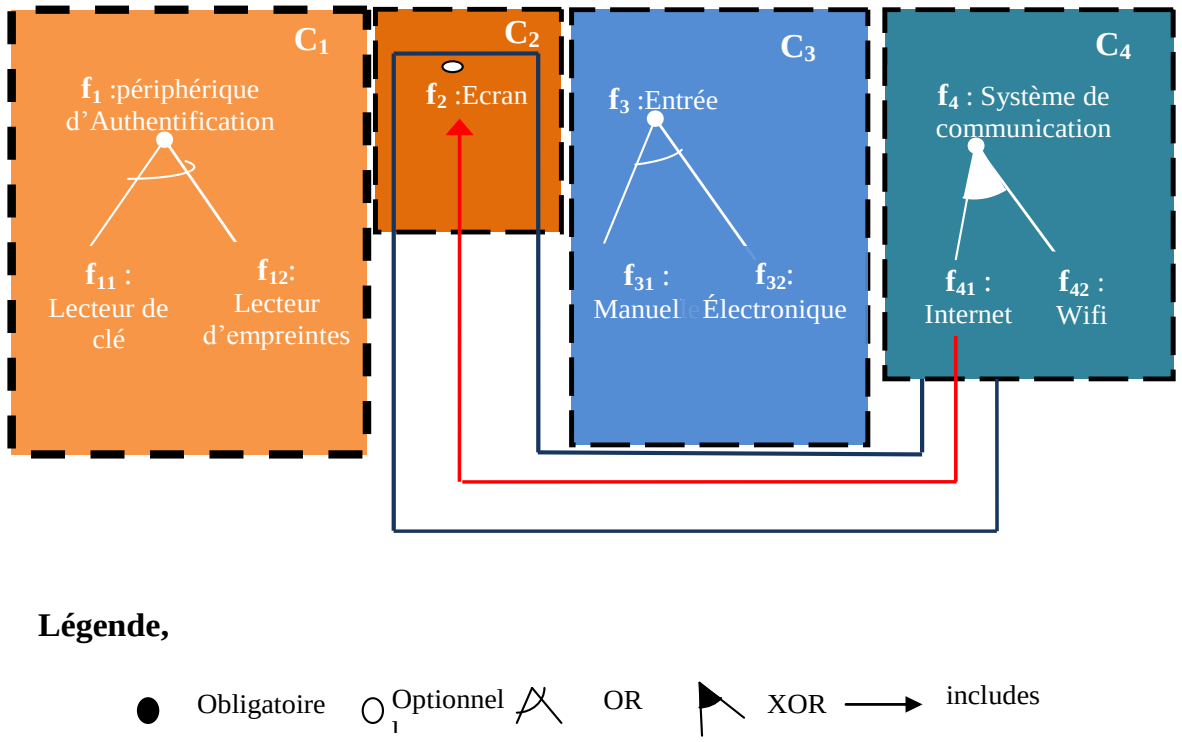


Figure 4.6 Les quatre composantes connexes dans le graphe des caractéristiques de la ligne de produits « Contrôle d'accès à une maison intelligente »

Plus de détails sur cette illustration sont mentionnés également dans le chapitre 3.

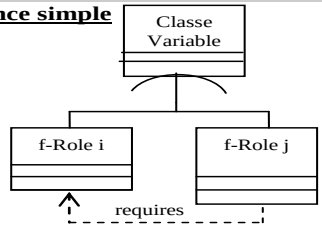
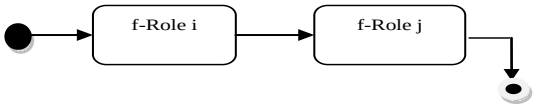
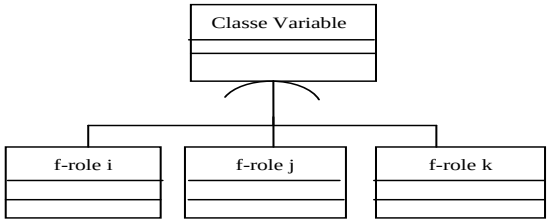
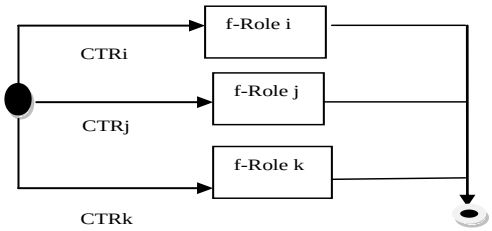
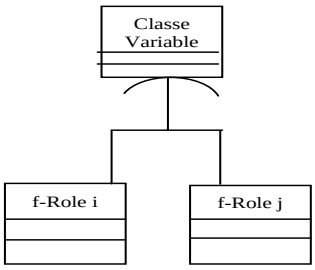
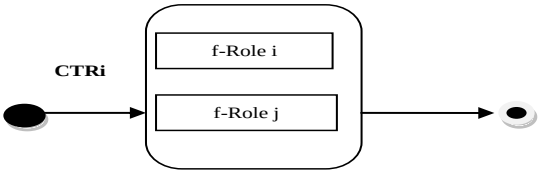
4.2.2.4. Les associations.

Dans cette section nous allons exposer tous les aspects descriptifs des deux associations introduites dans notre modèle d'évolution à savoir l'association « role »et « evolution »

a. L'association « role »

Cette association est la première association introduite en vue d'adapter le rôle UML aux lignes de produits.

Tableau 4.1. Modélisations des cas de dépendance dans le Modèle d'évolution .

Représentation dans le diagramme de classes	*Représentation dans la machine à etats*
Dependance simple 	 f-rolej depend du f-rolei
Exclusion Systematique 	
Inclusion Systematique 	

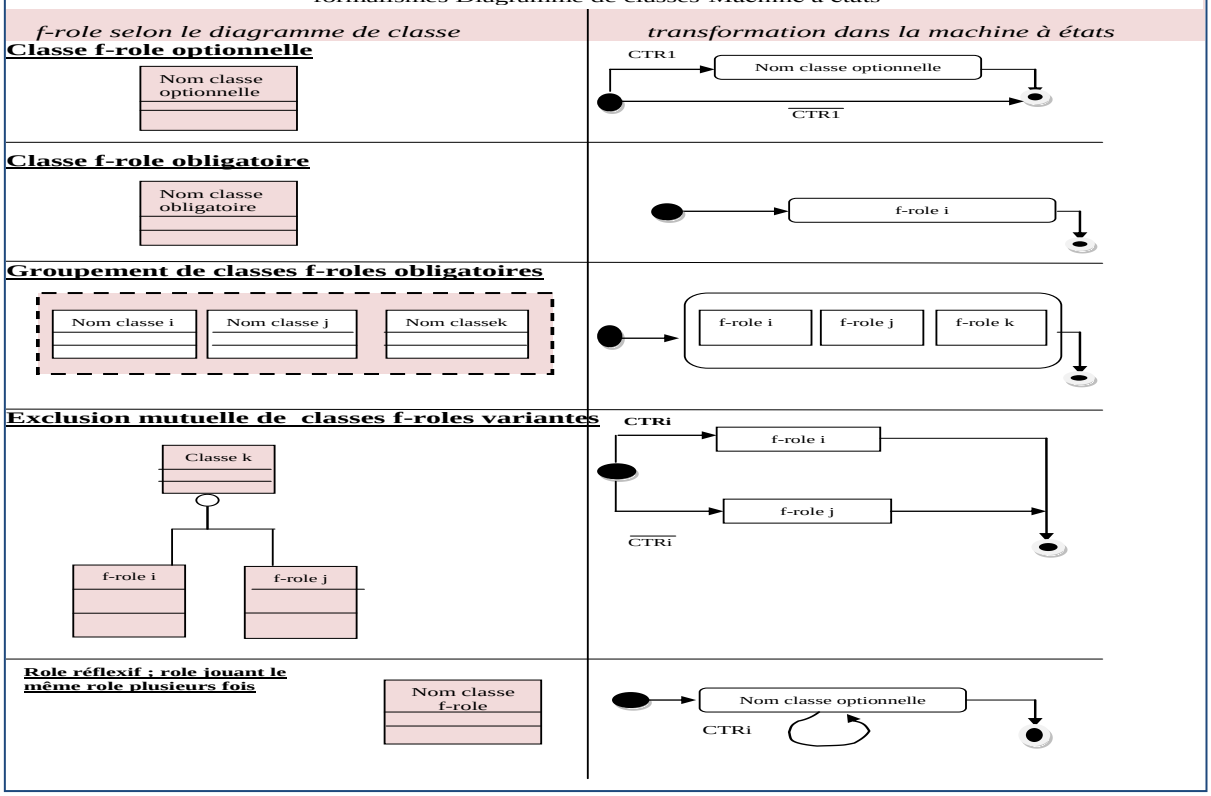
Un ensemble de mécanismes d'adaptation et d'intégration de cette association qui permet la relation d'éléments de type « Élément variable/f-roles » et « f-role/f-roles » seront décrits également dans cette section.

1. Définition

L'association « role » est une généralisation dynamique associant une classe à sa classe mère.

Pour la généralisation classique, les instances d'une classe sont statiques, contrairement à notre association « role » où à travers les classes f-roles, le modèle prend en compte les changements dynamiques des instances des éléments variables (classes variables).

Tableau 4.2 Modèle d'évolution. Principales transformations formalismes Diagramme de classes-Machine à états '



Nous avons choisi le concept d'association pour représenter la variabilité dans la partition variable –compartiment des formalismes selon le diagramme de classes- à travers une adaptation du concept de role UML.

Nous décrirons dans la section suivante comment intégrer cette association dans les Méta-modèles UML.

2. Sémantique

Notre association « role » est une adaptation de l'association « role-of » de [49] (conçue pour les systèmes uniques) à l'ingénierie des lignes de produits logiciels .Les formalismes accommodés et introduits sont les suivants :

- L'Association « role » est représentée par le symbole << ⌒ >> dans le premier compartiment du modèle d'évolution.
- L'association « role » (schématisée par V : v ← F) relie une classe dite variable ou classe de variants (V) et une autre classe, appelée classe de f-roles (F), qui décrit les f-roles exprimant l'évolution de la classe variable (V).
- Les instances de la classe variable (V) évoluent en jouant des f-roles , lesquels sont des instances de la classe de f-roles (F).
- Les instances de la classe variable s'appellent des variants, tandis que les instances de la classe des f-roles s'appellent des f-roles.

- Les classes de f-roles sont utilisées pour modéliser les aspects évolutifs des classes variables.
- Une classe variable non évolutive (sans f-roles) est représentée comme une instance permanente et spécifique d'une classe.
- Une classe évolutive f-role,en plus d'être une entité permanente et particulière d'une classe variable, est représentée par un ensemble d'instances de la classe des f-roles.
- Quand une classe variable évolue par l'acquisition d'un nouveau f-role, une nouvelle instance de la classe des f-roles appropriée est créée.
- Si une classe variable cède un f-role, l'instance de la classe des f-roles correspondante n'est plus prise en compte.

3. Illustration

La figure 4.3 montre une association « role » qui lie la classe variable '*méthode de calcul du kilométrage* ' à la classe de f-roles '*basée sur la distance'* .Cette dernière est une forme d'évolution d'une classe variable.

La classe '*méthode de calcul du kilométrage'* définit les propriétés permanentes du f-role '*méthode basée sur la distance'* dont les attributs *sont: type-fuel, quantité maximale du réservoir ; initialiseur du compteur de kilométrage cumulé ; compteur de kilométrage cumulé ; date-début-utilisation-plaquettes-freins ; date-changement-plaquettes-freins...*

La classe f-role '*méthode basée sur la distance'* définit quelques propriétés transitoires issues de l'évolution de la « *méthode de calcul du kilométrage'* en tant que méthode basée sur la distance du véhicule dont les attributs *sont : indice-fuel ; quantité consommée ; quantité disponible : distance parcourue ; nombre arrêts du véhicule.*

b. L'association « evolution »

Cette association est la deuxième association introduite , en vue d'adapter le role UML , aux lignes de produits.

1. Définition et Sémantique

- L'association « evolution » est définie comme une association qui permet de lier une classe f-role initiale évolutive ou f-role source à une autre classe de f-roles (dits f-roles cibles) eux aussi évolutifs mais pour la classe f-role source.

  La sémantique de cette association peut être décrite à travers les spécifications suivantes :

  -L'Association « evolution » est représentée par le symbole << ○ >> dans le 1er compartiment.

-L'association « evolution » (schématisée par E : f ← F) lie une classe f-role (f) à une autre classe de f-roles (F) qui décrit les f-roles évolutifs de la classe (f).

-Les instances de la classe f-role (f-roles évolutifs) évoluent en jouant d'autres f-roles, lesquels sont des instances de la classe de f-roles (F).

- Les instances de la classe f-role initiale s'appellent : des f-roles sources , tandis que les instances de la classe de f-roles résultant de l'évolution s'appellent des f-roles cibles.

-Les classes f-roles cibles héritent les classes f-roles sources.

-Une classe f-role non évolutive est représentée comme une instance permanente et spécifique d'une classe f-role.

-Une classe f-role évolutive , en plus d'être une entité permanente , et spécifique d'une classe f-role, est représentée par un ensemble d'instances de la classe de f-roles évolutifs.

-Quand une classe f-role évolue par l'acquisition d'un nouveau f-role, une nouvelle instance de la classe des f-roles cibles est créée.

-Si une classe f-role cède un f-role, l'instance de la classe des f-roles correspondante n'est pas prise en compte mais n'est pas supprimée.

2. Illustration

La figure 4.3 montre une association « evolution » qui lie (permet l'évolution de) la classe f-role source '*méthode de calcul du kilométrage'* à (vers) la classe de f-roles '*basée sur la performance du moteur'* .Cette dernière est une forme d'évolution d'un f-role.

La classe '*méthode de calcul du kilométrage* 'définit les propriétés permanentes non concernées par l'évolution du f-role '*méthode de calcul du kilométrage'* dont les attributs *sont : indice-fuel ; quantité consommée ; quantité disponible ; distance parcourue entre deux arrêts successifs ; nombre arrêts du véhicule*.

La classe f-role cible ou f-role évoluée '*méthode basée sur la performance du moteur'* définit des propriétés transitoires exprimant l'évolution de '*méthode de calcul du kilométrage* ' qui sont les attributs *: type-moteur ; puissance ; marque ; énergie.*

4.2.2.5. Les scénarios d'Evolution

Nous allons décrire dans cette section, l'ensemble des scénarios d'évolution basiques et complexes, de type Elément variable vers des f-roles ou « Élément variable/f-roles » et f-role vers des f-roles ou « f-role/f-roles ».

a. Les évolutions basiques

Ce sont les Evolutions de type « Elément variable/f-roles ». Nous les formalisons comme suit

Soit une classe de variants (V) représentant un élément variable, les évolutions possibles de cet élément sont des classes de f-roles notées : **f-roles$_i$.** Cette évolution peut s'effectuer selon deux scénarios différents :

- **L'acquisition de f-roles**, notée →F, indique qu'un élément variable peut acquérir un f-role de F sous certaines conditions ou automatiquement.
  Par exemple, '→basée sur la performance du moteur' signifie que la classe '*méthode de calcul du kilométrage'* peut acquérir le f-role '*méthode basée sur la performance du moteur* '.
- **La cession de f-roles,** notée F→, indique qu'un élément variable peut cesser de jouer un f-role de F automatiquement ou sous certaines conditions.
  Par exemple, '*méthode basée sur la distance →*' signifie que '*la méthode de calcul du kilométrage'* cesse d'être basée sur la distance.
  Ces deux scénarios ne tiennent pas compte du patrimoine en termes de f-roles dont dispose la classe source avant et après l'évolution.

b. Les évolutions complexes

Ce sont les évolutions de type « f-role/f-roles ». Ils sont dits complexes car ils sont réalisés à partir des deux scénarios élémentaires précédents avec une considération du patrimoine en f-roles de la classe source avant et après l'évolution.

Nous les formalisons comme suit :

Soit une classe de f-role (F) représentant un f-role. Les évolutions possibles de ce f-role sont des classes de f-roles notées : **f-role$_j$**. Cette évolution peut s'effectuer selon quatre scénarios différents.

En plus des deux premiers scénarios élémentaires, à savoir l'acquisition et la cession, deux autres scénarios plus complexes sont possibles, la mutation et le prolongement.

- **La Mutation.** ou acquisition d'un nouveau f-role avec cession de l'ancien .Elle est notée $F_1 \rightarrow^{-} F_2$ pour les classes de f-roles F_1 et F_2, avec F_1 distincte de F_2 .Cette notation indique qu'un f-role jouant le f-role f_1 (instance de F_1) peut cesser de jouer le f-role f_1 et acquérir le f-role f_2 (instance de F_2).
  En plus, les f-roles de F_2 ne peuvent pas être créés indépendamment des f-roles de F_1. Ce type d'acquisition peut être dans un seul sens ou dans les deux sens selon les contraintes imposées (« ou exclusif » ou « mutex » ou « dépendances »).Nous avons proposé

l'annotation [Mut] pour désigner la mutation dans le compartiment des formalismes de la machine à états du modèle d'évolution.

- **Le Prolongement .**ou acquisition d'un nouveau f-role sans cession de l'ancien .Elle est notée $F_1 \rightarrow + F_2$, avec F_1 non nécessairement distincte de F_2. Ceci indique que le f-role qui joue le f-role f_1 de F_1 peut gagner un nouveau f-role f_2 de F_2 en retenant f_1 (« ou inclusif »).

  Nous avons proposé l'annotation [Pro] pour désigner le prolongement dans le compartiment des formalismes de la machine à états du modèle d'évolution.

  Nous noterons que les 4 situations d'évolution relatives aux différents états liés aux associations « role » et « evolution » couvrent sémantiquement la totalité des cas d'expression de la variabilité évolutive.

4.2.2.6 L'Expression des contraintes de transition

L'expression des contraintes de transition dans le compartiment « machine à états » spécifique du modèle d'évolution, se fait à travers les prédicats**.** Soit une classe de variants V en relation avec les classes de f-roles F_1 et F_2**.**

Un prédicat de transition est associé à F_1 pour décrire les conditions nécessaires et/ou suffisantes sur la façon dont les f-roles joués par F_1 peuvent explicitement ou automatiquement acquérir des f-roles dans F_2. La classe source F_1 peut être une classe de f-roles évolués dans une mutation ou un prolongement ou bien une classe de f-role évolutifs dans une acquisition.

Les prédicats qui décrivent les transitions peuvent être déclarés par l'utilisation du langage OCL d'UML [89].

Dans ce qui suit, nous donnons quelques exemples de prédicats de transition (cas issus de la figure 4.3) exprimant les 4 scenarios d'évolution par des f-roles dans le modèle d'évolution.

- Pour la contrainte (CTR1). *« Une méthode de calcul du kilométrage jouant le f-role d'une méthode de calcul basée sur la consommation d'énergie peut l'être pour le fuel 'essence normale' et pour le fuel 'essence sans plomb' en même temps » (la multi-instanciation). le prédicat* « nombre-type-fuel ≤ 2 Λ énergie = 'essence'» déclare 2 comme étant le maximum de types de fuel que cette méthode peut prendre en charge en énergie de type essence en même temps. La notation pour (CTR1) est la suivante :

  B*asée sur la consommation d'énergie* $\rightarrow +$ B*asée sur la consommation d'énergie*

- Pour (CTR2). *« Une méthode jouant le f-role d'une méthode de calcul basée sur la distance peut être en même temps basée sur la consommation d'énergie si la puissance du*

moteur est supérieure ou égale à 9 CV » car nécessitant une économie dans la consommation du fuel. La notation pour (CTR2) est la suivante :

B*asée sur la distance* ⟶ + B*asée sur la consommation d'énergie*

- Pour (CTR3). *« Une méthode jouant le f-role de méthode basée sur une commande automatique peut basculer vers une méthode basée sur la consommation d'énergie si le statut de la commande = 0 »* c.à.d. si la commande tombe en panne. Le prédicat statut-commande = 0 est associé aux méthodes qui basculent vers un calcul basé sur la consommation d'énergie. La notation pour (CTR3) est la suivante :

B*asée sur la commande automatique* ⟶ - B*asée sur consommation d'énergie*

- Pour (CTR4). *« Une méthode jouant le f-role de méthode basée sur la distance parcourue peut basculer vers une méthode basée sur une commande automatique si quantité-fuel-restante inférieure à 30 % quantité-globale »*. Le prédicat « *quantité-fuel restante < 30 % quantité-globale* » est associé aux méthodes basées sur la distance qui basculent vers un calcul basé sur une commande automatique.

La notation pour (CTR4) est alors la suivante :

B*asée sur la distance* ⟶ - B*asée sur la commande automatique*

4.2.2.7 Intégration des éléments de notre modèle dans les Méta-Modèles UML

Nous présentons dans cette section les mécanismes d'intégration liés aux associations « role » et « evolution » ainsi que les f-roles dans les Méta-modèles d'UML en respectant les formalismes de spécification présentés dans les sections précédentes.

a. Intégration de l'association « role »

Nous définissons dans ce qui suit le Méta-modèle de l'association « role » décrite dans la section précédente.

1. **Définition de l'association « role » dans le Méta-modèle UML**
   - Nous définissions l'association « *role* » comme sous-classe directe de « *Relationship* » appartenant au paquetage « Core ».
   - Nous définissons deux Méta-classes « V*ariableElement* » et « *f-RoleElement* » comme des sous-classes directes de « ModelElement » .

La Méta-classe « *VariableElement* » représente les classes de variants.

La Méta-classe « *f-RoleElement* » représente les classes de f-roles impliquées dans une association « role » et qui sont évolutifs.

Notre Méta-modèle est reporté sur la figure 4.7. Nous allons détailler dans la suite chacun des éléments de ce Méta-modèle.

- Les classes V sont des instances de la Méta-classe « *VariableElement* ».
- Les classes F sont des instances de la Méta-classe « *f-RoleElement* ».
- Nous pouvons ainsi considérer que V et F représentent les qualificateurs 'V*ariable'* pour (V) et f-*role* pour (F) dans la figure 4.7.
- L'association « role » entre V et F, peut être décomposée en 2 sous-associations : '*VariableL*ink*'* (reliant un f-role à un élément variable) et 'f*roleLink'* (reliant un élément variable à un f-role) comme le montre la figure 4.7.

2. Sémantique.

Ainsi, dans le Méta-modèle, « role » est une sous-classe de « *Relationship* » qui sert à modéliser les associations .Elle représente le lien entre une classe de variants et une classe de f-roles. Elle comporte les attributs suivants :

- **discriminateur :** indique le nom de la partition variable évolutive à laquelle le lien de f-role appartient. Il désigne tous les liens de f-roles qui partagent la même classe variable (classe de variants). Chaque partition représente une dimension orthogonale qui regroupe un ensemble de classes de f-roles associées à une classe de variants.
  Les classes de f-roles de la même partition variable ont toutes le même discriminateur.
- **cessionPredicate :** est une assertion décrivant les conditions nécessaires et/ou suffisantes à vérifier par les variants de la classe variable pour que ces derniers cèdent explicitement ou automatiquement des f-roles.
- **acquisPredicate :** est une assertion décrivant les conditions nécessaires et/ou suffisantes pour que les variants de la classe variable puissent acquérir explicitement ou automatiquement des f-roles.

  La Méta-classe « *role* » est liée par deux associations « Variable » et « f-Role » aux Méta-classes '*VariableElement*' et 'f-*RoleElement*'.

b. Les autres Associations introduites au Méta-modèle

Les autres associations du Méta-modèle en relation avec « role » sont : « Variable », « f-Role » , « froleLink » et « VariableLink ».

1. Pour les associations « Variable » et « f-Role »

- **« Variable » :** désigne un *VariableElement* qui peut évoluer (évolutif) vers un f-role.

- **« f-Role » :** désigne un f-*RoleElement* qui représente une éventuelle évolution d'un variant.

Les sous-classes VariantElement et f-RoleElement sont des sous-classes de « ModelElement » qui peuvent participer dans une association « role ». VariableElement et f-RoleElement sont également des Méta-classes abstraites.

- **Définition des cardinalités**
  - **1** du côté du qualificateur *Variable* : signifie qu'un seul VariableElement de type '*classe de variants'* participe à l'association « role ».
  - **1** du côté du qualificateur f-*role* : signifie qu'un seul f-RoleElement de type « *classe de f-roles* »(c.à.d. un f-RoleElement) participe dans l'association « role ».
  - **1** du côté du qualificateur *VariableLink* : signifie qu'un f-RoleElement ne peut avoir qu'une seule relation avec la classe de variants.
  - ***** du côté du qualificateur f*roleLink* : siginifie qu'un VariableElement peut avoir plusieurs relations avec les classes de f-roles.

2. **Pour les associations « froleLink » et « VariableLink »**

- **froleLink :** représente la liaison , qui part de la classe de variants , vers la classe de f-roles.
- **VariableLink :** représente la liaison qui part de la classe de f-roles vers la classe de variants.

c. Mécanismes de prise en compte des scénarios d'évolution basiques

Les évolutions basiques sont au nombre de deux : la cession d'un f-role préalablement acquis ou l'acquisition d'un nouveau f-role.

- La cession d'un f-role est contrôlée par l'attribut *cessionPredicate* du modèle f-role. Quand l'assertion « *cessionPredicate* » est satisfaite , le f-role de cette liaison liaison sera détruit.
- L'acquisition (assignation) d'un f-role est contrôlée par l'attribut « *acquisPredicate* » du modèle f-role. Quand l'assertion *acquisPredicate* est satisfaite, un nouveau f-role pour cette liaison est construit.

Les évolutions complexes à base des deux scénarios élémentaires précédents sont au nombre de deux. Elles sont prises en compte à travers l'association « evolution » et seront décrites dans la section suivante.

d. Intégration de l'association « evolution »

Nous définissons dans ce qui suit le Méta-modèle de l'association « evolution ».

1. **Définition et Syntaxe de la classe-association « evolution » dans le Méta-Modèle UML**

L'association « evolution » est une classe-association récursive qui relie une classe de f-RoleElement appelée *classe source* et une classe de f-RoleElement appelée *classe cible*.

Elle permet de spécifier les types d'évolution complexes à base d'acquisition et de cession c.à.d. les prolongements et les mutations décrites dans la section 4.2.2.5 ainsi que

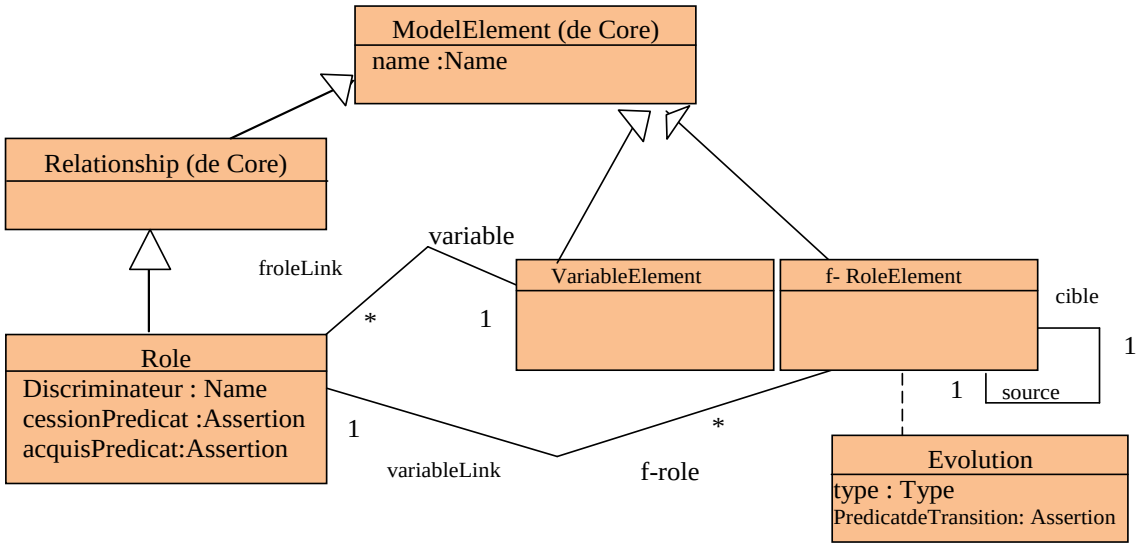


Figure 4.7 Le Méta-modèle intégrant les associations « *role* » *et* « *evolution* » à UML.

Les prédicats de transition des f-roles décrits dans la section 4.2.2.7.

Les Attributs de l'association « evolution » sont :

- **Type.** est un attribut à 2 valeurs qui définit les scénarios complexes d'évolution des f-roles. Il peut prendre l'une des 2 valeurs:« prolongement » ou « mutation ».
- **PrédicatdeTransition.** est un attribut de type Assertion. Il décrit les conditions nécessaires à une instance de la classe f-role source pour pouvoir évoluer vers la classe f-role destination.

2. **Les associations liées à « evolution »**

Les deux associations liées à « evolution » dans le Méta-modèle sont :

- **Source**. désigne une classe de f-roles : une classe f-RoleElement qui va subir soit les règles du prolongement ou de la mutation.
- **Cible**. désigne la classe des f-roles destination après que les conditions de transition sur la classe source aient été vérifiées.

Les autres liens, associations et prédicats en relation avec « evolution » ont déjà été décrits dans la section définissant l'association « role » (figure 4.7).

e. Intégration de la Machine à Etats

Nous allons décrire dans cette section la technique utilisée pour adapter la Méta-modélisation de la machine à états à notre modèle en vue de pouvoir l'intégrer dans les Méta-modèles d'UML. Nous en donnerons une illustration à travers une partition variable évolutive du modèle d'évolution.

1. Technique d'intégration

La technique utilisée pour intégrer à UML la machine à état comme spécifiée dans les formalismes de l'infrastructure d'UML repose sur l'usage du concept de 'virtualité d'une machine à état-transition'. Dans notre modèle d'évolution de la variabilité, nous utilisons le même principe avec des modifications dans certains paramètres en vue de la considération des concepts proposés.

Pour réaliser cette intégration, il y a lieu d'adapter le comportement de ce mécanisme UML en le redéfinissant ou en le reparamétrant . Une tentative a déjà été opérée par [90] pour l'étape de dérivation dans le contexte des lignes de produits. Dans ce travail, la machine virtuelle a pu être redéfinie par une machine de raffinement associée puis paramétrée non pour décrire des états et leurs transitions mais pour spécifier une application particulière de la ligne de produits.

C'est la technique que nous avons retenue pour paramétrer notre machine à états spécifique à notre modèle d'évolution .Nous avons appliqué cette adaptation de la machine pas pour un état généraliste mais pour un état spécifique. Pour notre modèle, cet état est une classe f-role qui évolue dans le cadre de la partition virtuelle évolutive.

Nous introduisons la virtualité de la machine par le stéréotype <<virtual>> avec le tagged value « virtualPart » qui indique l'occurrence de la machine virtuelle.

« VirtualPart » pour notre modèle, représente la partition variable évolutive notée (C_k) qui comporte bon nombre de classes f-roles du 1^{er} compartiment du modèle d'évolution.

2. Illustration

Chaque partition variable renferme une classe variable avec ses sous classes de f-roles évolutifs.

Les partitions variables sont numérotées (Ci). La figure 4.6 montre un exemple de ces partitions.

Nous présentons à travers l'illustration de la figure 4.8, un exemple d'une machine à états qui référencie une machine virtuelle (PVME) pour la partition C_k du système de surveillance d'un moteur.

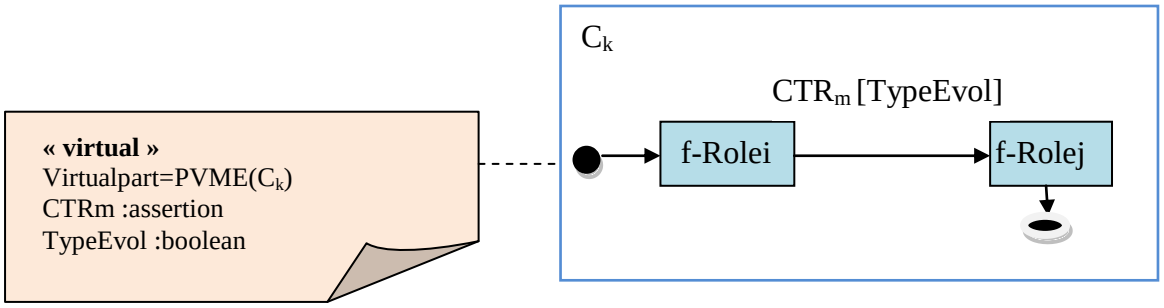


Figure.4.8 Exemple d'une machine à états
référençant une machine virtuelle PVME pour la partition C_k

Les **f-rolei** et **f-rolej** sont les états représentés dans le 2eme compartiment du modèle d'évolution pour la partition variable évolutive C_k.

CTR_m sont des assertions qui représentent les contraintes d'évolution.

TypeEvol représente le scénario mutation pour la valeur nulle sinon il s'agit d'un prolongement.

4.3 Conclusion

Dans ce chapitre, nous avons proposé un nouveau modèle de spécification de l'évolutivité de la variabilité contenue dans les lignes de produits en s'appuyant sur le modèle clé des caractéristiques et sur le standard de la notation orientée objet UML.

En se concentrant sur l'aspect extensibilité des caractéristiques et sur l'importance de sa prise en compte pendant l'étape de modélisation par une nouvelle représentation des variants, nous avons mis au point des mécanismes d'évolution de la variabilité à travers l'utilisation du concept de role UML revu et adapté au contexte des lignes de produits.

Dans ce modèle, une caractéristique variable peut évoluer simplement en procurant ou en cédant des roles (f-roles) tout en préservant ou en perdant d'autres à travers son cycle de vie sans toutefois changer.

Face à l'inadaptation du modèle des caractéristiques à représenter l'aspect évolutif de la variabilité préconisé dans notre étude , nous avons intronisé un nouveau modèle de représentation , le modèle d'évolution , qui transforme le modèle de caractéristiques en diagramme de classes dans une première étape puis dans une seconde , scinde le diagramme de classes obtenu en partitions variables selon un processus bien défini et enfin fait correspondre à

chaque partition son volet évolutif avec une prise en compte des conditions d'évolution à travers une machine à états spécifique.

Devant l'inadaptation du rôle UML à représenter l'aspect évolutif de la variabilité, nous avons proposé l'expression de la variabilité dans chaque partition variable au niveau des 'éléments variables' à l'aide d'une nouvelle association désignée « *role* ».

Dans notre modèle, les éléments variables contenus dans les Partitions sont représentés sous forme de classes et leurs variants sous forme de sous-classes *f-roles*. Tout variant est considéré comme une première forme d'évolution (f-role) de son élément variable. De plus, l'association « *role* »permet également aux classes f-roles d'hériter leurs classes variables .A chaque Partition variable correspond une machine à état-transitions qui représente l'évolution de la variabilité à travers les f-roles issus des f-roles évolutifs des variants avec leurs contraintes d'évolution.

Pour compléter le modèle, nous pouvons détailler les conditions d'évolution par un listing en OCL.

Pour terminer, nous avons intégré tous les concepts introduits dans les Méta-modèles d'UML.

Conclusion Générale
&
Perspectives

Conclusion Générale

Nous avons adressé dans ce travail de thèse des problématiques spécifiques à deux aspects importants des lignes de produits logiciels à savoir le développement de modèles et l'évolutivité de la variabilité. Nous avons ciblé le paradigme ligne de produits compte tenu du fait qu'il représente le contexte de développement le plus utilisé pour les systèmes embarqués. Dans la littérature, les démarches existantes pour le 1er aspect, n'ont pas proposé de méthodes effectives dédiées au développement propre d'une ligne de produits logiciels. Elles se sont concentrées principalement sur la mesure soit de la similarité renfermée dans les modèles de la ligne de produits, soit de la qualité des produits et de l'effort de développement. L'objectif recherché demeure évaluatif et repose sur l'existence préalable des modèles de la ligne de produits. L'enjeu pour notre travail a été de proposer une démarche efficiente dédiée au développement de modèles pour la ligne de produits logiciels.

Pour l'aspect évolutivité, la problématique a consisté à trouver des mécanismes capables de représenter la variabilité et son aspect évolutif à travers un seul dispositif compte tenu que la majorité des approches existantes dans ce domaine traitent ces deux dimensions séparément à l'aide de mécanismes complexes.

Notre proposition liée au développement d'une ligne de produits est basée sur le déroulement d'un processus qui permet aux parties prenantes d'explorer au maximum les possibilités de créer un modèle performant de caractéristiques pour un domaine donné. Le processus proposé démarre à partir de configurations d'un ensemble restreint de produits similaires formulés selon notre modèle de variabilité. Ces dernières soumises aux analyses, aux transformations et aux évaluations d'une série de modèles (Analyse, Mapping , Corrélation et Evaluation) génèrent en sortie et dans toutes les étapes du processus un ensemble de métriques notamment liées à l'analyse , la corrélation et l'évaluation . Notre démarche offre également aux parties prenantes à travers des jeux de vues liées aux modèles, un ensemble de perspectives. Ces dernières sont obtenues selon un ensemble de quatre critères de différentes granularité (l'utilité, la profondeur, la couverture en niveaux hiérarchiques et la scannérisation).

Dans le but de montrer l'efficacité de notre démarche, nous avons utilisé dans toutes les étapes du processus des illustrations basées sur une étude de cas du domaine des maisons intelligentes. Un outil-support a été développé en vue d'implémenter les activités du processus sous forme de fonctionnalités. Cette version couvre toutes les étapes de notre

démarche à l'exception de la dernière étape relative à l'intégration des diagrammes optimaux de caractéristiques car basé sur un mécanisme graphique non supporté par notre outil.
Dans le deuxième aspect abordé, nous avons cherché à représenter l'évolutivité de la variabilité à un stade précoce, celui de la modélisation du domaine c.à.d. au stade des caractéristiques et non à l'étape de dérivation lorsqu'il s'agit de composants. Pour cela, nous avons utilisé une expression de la variabilité et de son évolution à travers les rôles évolutifs (les f-roles) issus d'une adaptation du rôle UML -dédié initialement aux systèmes uniques-, au domaine multi-systèmes des lignes de produits. Pour parfaire cette modélisation, nous étions contraints de changer la perception de la ligne de produits, pour la considérer comme un ensemble de points de variation jouant des f-roles pouvant évoluer. Cette évolution concerne tout aussi les points de variation que les f-roles eux-mêmes à travers d'autres f-roles .Pour cela nous avons introduit un nouveau modèle « le modèle d'évolution » construit sur la base de partitions variables évolutives .Chaque partition est scindée en deux compartiments , le premier permet de représenter un point de variation avec les f-roles qui lui sont liés selon les formalismes du diagramme de classes et le second représente à travers une machine à états associée , les évolutions possibles des points de variation et leurs f-roles sous forme d'états transition avec les contraintes d'évolution.

En vue de faciliter l'intégration aux méta-modèles d' UML des concepts proposés , nous avons ajouté deux nouvelles associations , la première désignée « *role* » lie les f-roles aux classes représentant les points de variation dans la partition variable selon les formalismes du diagramme de classes et la deuxième désignée « evolution » exprime le lien entre les f-roles et leurs formes d'évolution potentielles, les f-roles évolutifs .De nouveaux f-roles peuvent être joués ou cesser de l'être selon que le dernier f-role acquis soit préservé dans le patrimoine des f-roles de la classe représentant le f-role ou le point de variation ou qu'il en soit supprimé . Pour cela, quatre scénarios d'évolution ont été proposés : deux basiques : l'acquisition et la cession de f-roles et deux complexes : le prolongement du patrimoine en f-roles et la mutation.

.

Perspectives

Dans notre démarche liée à l'aspect développement, tous les types de dépendance n'ont pas été pris en charge. Dans notre prochain travail de recherche, nous allons étendre notre approche pour supporter non seulement tous les types de dépendance entre les unités de construction mais aussi ceux relatifs aux unités d'analyse d'un produit à savoir :

1) Entre les caractéristiques ("Caractéristique-Groupe de caractéristiques", "Groupe de caractéristiques-Caractéristique") au lieu du type classique "Caractéristique-Caractéristique" ; et
2) Entre les Unités de Scannérisation ("Unité de scannérisation - Groupe d'unités de scannérisation") et "Groupe d'Unités de scannérisation- Unité de scannérisation " au lieu du type classique "Unité de scannérisation- Unité de scannérisation "

Une extension de l'outil FeMoMaS par l'ajout de deux fonctionnalités sera examinée dans notre prochaine version. Ces dernières offriront davantage d'automatisation au processus en permettant une exécution :

1) Sans aucun choix préalable de paramètres (paramètres par défaut et déroulement entièrement automatisé);
2) Selon le choix de l'utilisateur, d'abord selon une sélection partielle par étape puis totale (de tout le processus de développement).

Pour l'aspect évolutivité , la recherche d'une simplification du modèle d'évolution proposé pour exprimer la variabilité et son évolution par les f-roles ainsi qu'une amélioration des vues instantanées de ce modèle par la prise en compte de l'aspect historique pourront faire l'objet de notre prochain travail.

Une autre perspective concerne la génération du code d'application des produits à partir des modèles UML. Cette perspective très importante pour les lignes de produits logiciels est envisageable. Elle concerne la génération du code à partir des modèles UML existants. Nous ciblerons les vues UML non encore abordées, à savoir les vues « Composants » et « Machines à Etats ».

Références

[1]. P, Kadionik ; “Les systèmes embarqués : une introduction”, Linux Magazine, Série 24, pp. 4-8, Février-Mars 2006.

[2]. Burns; “The development of embedded and mobile systems, Java and Ada” [Lea 1997] et C#”,1997. [Lea 1997] D. Lea. Concurrent Programming in Java . 340 pages. Addison Wesley 1997.

[3]. P, Kadionik; “Ressources sur les systèmes embarqués ”, http://kadionik.vvv.enseirb-matmeca.fr/systémesembarqués , 2016.

[4]. R, Zurawski; *“*Embedded Systems Handbook*”,* Editions CRC Press, New York, 2006. Embedded Systems Handbook**,** Second Edition -Volume Set.

[5]. B, Geoff; “Embedded Systems: A Primer.” Staff, IBM. Rational. Novembre 2003.

[6]. L, Hotz ; K, Wolter ; T. Krebs ; S. Deelstra ; M. Sinnema ; J. Nijhuis ; J. MacGregor ; “ Configuration in Industrial Product Families, The ConIPF Methodology”, IOS Press, 2006.

[7]. L-M, Northrop ; “A Framework for Software Product Line Practice”,Version 3.0. Software Engineering Institute , 2002. Web : http://www.sei.cmu.edu/plp/framework.html,2002.

[8]. W,El Elkaim; “System Family Architecture Glossary”. Technical report, ESAPS Project, 2001. OMG. UML specification. 2001.

[9]. M, Clauß ; “Modeling Variability With UML” in GCSE’2001 –CiteSeerX-Young Researcher Workshop , Erfurt , 2001 .

[10]. T, Ziadi ; J-M, Jézéquel ; “ Product Line Engineering With the UML : Deriving Products ”, Chapter in Software Product Lines – Research Book, Springer, 2006.

[11]. K, Pohl; G, Böckle; F-J-v-d,Linden” Software Product Line Engineering: Foundations - Principles and Techniques (OVM) ” , Heidelberg , 2005.Book - Software Product Line Engineering: Foundations, Principles and Techniques - Springer-Verlag New York, Inc. Secaucus, NJ, USA ;2005.

[12]. P, Clements; L. Northrop ; “ Software Product Line : Practices & Patterns ”, 3d .- Addison-Wesley , Reading, 2001.

[13]. C-W.Krueger BigLever Software Methodology - published in OOPSLA’07 companion to the ACM SIGPLAN conference on Object-oriented programming systems and applications companion- Montreal – Canada , Octobre 2007.disponible sur le site : http://www.biglever.com ,.

[14]. N, Loughran ;P Sanchez , N,Gamez , J ,Kovacevic “. Survey on State-of-the-Art in Product Line Architecture Design www.researchgate.net/publication/255907099. 2007

[15]. J-M. Jézéquel ;” Reifying Variants in Configuration Management ” . ACM Transaction on Software Engineering and Methodology, 8(3) :284–295, July 1999.

[16]. C, Atkinson ; J, Bayer ; C, Bunse ; E, Kamsties ; R, Laqua ; D, Muthing ; B, Paech ; J, Wust and J,Zettel ; “Component-based Product Line Engineering with UML,” *Component Software Series* ,2001.

[17]. E, Christoph ; Siemens, Erlangen, Germany ; " Light-Weight Tool for Staged Product Derivation " in 16thSPLC'12-Vol 1–pp:146-155 – 2012.

[18]. A, Gacemi ; D, Seriai ; "La réutilisation : Concepts et Techniques" , Technical Report ,2003-4-2, Ecole des Mines de Douai, France. 2003.

[19]. F, Barbier; C, Cauvet ;M, Ousselah, D, Rieu; C, Souveyet; "Composants dans l'ingénierie des systèmes d'information-Concepts clés et Techniques de réutilisation"; Assises nationales du GDR-13 ,2002.

[20]. F, Le Mouel ; " Environnement Adaptatif d'Exécution Distribuée d'Applications dans un Contexte Mobile", Thèse de doctorat – Université de Rennes - IRISA-France; 12/2003.

[21]. Y-Y, Xu ; B, Traverson ; "Procédés de Réutilisation pour les Lignes de Produits Logiciels" ; Source: DBLP-conference: Actes du XXVIème Congrès INFORSID, Fontainebleau, France, 27 au 30 mai 2008 Université de Versailles S-Quentin , France - 2008.

[22]. D-L, Hurtz ; « Quatre étapes pour la réutilisation du logiciel » ; Conférence AOSD'2010, Université de Rennes , France ,2010.

[23]. Y,Djebar; N,Guersi; MT,Kimour ;and R.Rouabhi Rachid , "SysPL:A generic Model-based approach for Software product Line Enhanced Reusability " in World Journal of Managing & Engineering " , Vol:1(4):pp:124-137, December, 2013.

[24]. Y,Djebar ; MT,Kimour; and N,Guersi; " Optimization of Asset engineering and product engineering for and by reuse" in IEEE- International Conference on Information Technology and e-Services- ICITES' 2012 du 24 au 26 Mars 2012 – Université de Sousse – Tunisie.

[25]. M, Svahnberg ;M-J, BoschJ ; " Issues Concerning Variability in SPL ", Workshop IWSAPF- LNCS –Vol :1951, pp.146-157,Springer , 2000.

[26]. D, Benavides ; P, Trinidad ; R, Cortes;"Automated Reasoning on Feature Models",17th CAISE'05,Vol. 3520.Springer,pp 491–503 ; 2005.

[27]. D, Batory ; " Feature Models, Grammars , and Propositional Formulas " , Lecture Notes in Computer Science Journal , Vol. 3714, p 7, 2005.

[28]. D, Zubrow ; G, Chastek ; "Measures for Software Product Line" in Software Engineering Measurement and Analysis Initiative issues, Octobre 2010 , SEI- http://www.sei.cmu.edu/measurement/.

[29]. Y, Djebar ; N, Guersi ; MT, Kimour ; " Un processus en 4 phases pour une meilleure testabilité des produits dans les lignes de produits logiciels " in Conférence internationale sur l'intelligence artificielle et les technologies de l'information-ICA2IT'14-du 10-12 Mars 2014 à l'Université de Ouargla-Algérie.

[30]. M, Voelter ; I, Groher ; " Product Line Implementation Using Aspect-Oriented and Model-Driven Software Development", SPLC'07, Washington, DC, USA, pp. 233-242, 2007.

[31]. M, Janota ; J, Kiniry;"Reasoning about Feature Models in Higher-Order Logic" in 11thSoftware product line Conference, Kyoto , Japan - pp 13-22, 2007.

[32]. M, Boskovic ; E, Bagheri ; D, Gasevic ; B, Mohabbati ; N, Kaviani ; M, Hatala ; " Automated Staged Configuration with Semantic web Technologies" in IJSEKE journal. World Scientific, 2010.

[33]. T, Forster; D, Muthig, and D, Pech; " Understanding Decision Models.Visualization and Complexity reduction of software variability " In Proceedings of the 2nd Int. Workshop on Variability Modeling of Software-intensive Systems,Essen, Germany, January 2008.

[34]. H, Arboleda ; R, Casallas and J-C, Royer; " Dealing with fine grained configurations in Model-Driven SPL ". In Proceedings of the 13th International Software Product Line Conference (SPLC'09) , San Francisco, USA, August 2009..

[35]. M, Volter and I, Groher. “Product line implementation using aspect-oriented and model driven software development” In Proceedings of the 11th International Software Product Line Conference, pages 233-242, 2007.

[36]. F,Bachmann; “A Meta-Model for Representing Variability in Product Line Development” , Heidelberg , 2003. Volume 3014 of the book series Lecture Notes in Computer Science (LNCS) - Springer

[37]. K, Czarnecki and W, Ulrich Eisenecker ; “ Generative Programing :Methods ,Tools, and Applications ” Addison-Wesley, 2000.

[38]. Y,Djebar; MT,Kimour and N,Guersi ; “ A Generic Variability Model-Based Approach for Enhanced Reusability” in 2nd International Conference of Information Systems and Technologies-ICIST’12 du 24-26 Mars, 2012 – Université de Sousse-Tunisie.

[39]. K, Czarnecki ; M, Antkiewicz ; “ Mapping Features to Models : A Template Approach based on Superimposed Variants ” ; 4th International Conference Generative Programming and Component engineering, vol. 3676 of LNCS, Springer-Verlag, pp. 422-437, 2005a , 2005.

[40]. Y, Djebar et MT, Kimour ; “ Un modéle pour la variabilité dans les Lignes de Produits logiciels” International Conference of applied informatics ICAI’09, BBA, Algeria ,15-17 11,2009.

[41]. BigLever; GEARS engénierie ; Website http://www.biglever.com/software product line enginnering solutions for systems and software, 2012.

[42]. F, Fleurey ; B, Baudry ; R, France ; S, Ghosh ; “A Generic Approach For Automatic Model Composition ”, Workshop on Aspect-Oriented Modeling, AOM at Models’07, 2007.

[43]. H, Bouaricha; H, Berrebah; Y, Djebar; R, Rouabhi; N, Guersi; MT, Kimour; MR, Djebar , “ Approach Based on a generic-predictive Model in search of the least toxic insecticides in toxicological application of a SysPL tool” in Journal of Computational Methods in Molecular design , Vol: 3:issue 4; Décembre 2013; pp:1 à 5.

[44]. R, France ; F, Fleurey ; R, Reddy ; B, Baudry ; and S, Ghosh ; “ Providing Support for Model Composition in Metamodels”, 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007) , Annapolis, MD, USA, 2007.

[45]. G, Perrouin ; J. Klein ; N. Guelfi ; J-M Jézéquel ; “ Reconciling Automation and Flexibility in Product Derivation”, 12th Software Product Line Conference, Limerick, Ireland, 2008.

[46]. Y, Djebar; N.Guersi; MT, Kimour; “ A Model-Based on role for Software Product Line Evolving Variability ” in Springer - Advanced Studies in Computational Intelligence (SSCI) , Vol : 488 – 13, pp :259-272.2013

[47]. Y, Djebar; N, Guersi; MT, Kimour ; “ A Model Based on Role for Software Product Line Evolving variability” in 4th International Workshop on Computer Science and its applications” – CIIA’13 – du 4-6 Mai 2013 à l’université Tahar Mouley de Saida – Algeria .

[48]. Y, Djebar; MT, Kimour; “ A software product line extensibility Model” in international Workshop on systems communication and engineering in computer science – CECS’10 , organisée conjointement par l’université de Batna et l’université de technologie de Berlin-TU-Berlin-Germany du 3-5 octobre 2010.

[49]. M, Dahchour ; H, Rayad ; Y, Lakhrissi ; A, Krioule ; “ Extension UML par les Rôles ” . e-TI, Numéro 4 / Issue 4. Spécial MCSEAI , 29 juin 2007, http://www.revue-eti.net/document.php-id=1399.

[50]. C, Seidl ; F, Heidenreich ; U, Aßmann ; “ Co-Evolution of Models and Feature Mapping in Software Product Lines”,Proceeding of the 16th International Conference of the Sofware Poduct Line ,Salvador-Brazil, 2012.

[51]. A, Jodłowski; J, Płodzień; E, Stemposz and K, Subieta ; Poland-Varsovia University ; " Introducing Dynamic Object Roles into the UML Class Diagram", IASTED International Conference on Software Engineering and Applications (SEA), ACTA , CoopIS/DOA/ODBASE, volume 2888 of LNCS , 2003.

[52]. R, Depke ; G, Engels ; J-M, Küster ; " On the Integration of Roles in the UML ". Technical Report No. 214, University of Paderborn, Germany, 2000.

[53]. B-A Guido ; L, Van der Torre, and H, Verhagen ; " Roles, an Interdisciplinary Perspective" , University of Torino, Italy **, "** Roles, An Interdisciplinary Perspective " : Papers from the AAAI Fall Symposium Guido Boella, James Odell, Leendert van der Torre, and Harko Verhagen, *Cochairs*,November 4-6, 2005, Arlington, Virginia Technical Report FS-05-08 , 164 pp., 2005

[54]. S, Kazimierz ; A-J, Lowskiz; P, HabelAx ; J, PLodzie ; " Conceptual modeling of business applications with dynamic object roles" , Book technology supporting business solutions pp 49-71 , 2003

[55]. D, Chernuchin ; G, Dittrich ; "Dependencies of Roles" , Computer science Dept. Dortmund University , Germany ,In 2005 AAAI Fall Symposium.

[56]. H, Zhu ; R, Alkins ; "Towards Role-Based Programming", Department of Computer Science, Nipissing University, 2008. Workshop on Role-Based Collaboration , CSCW 2006, Banff, Alberta, Canada, Nov. 4, 2006.

[57]. J, Paulo; A, Almeida; G, Guizzardi; " Semantic Foundation for Role-Related Concepts in Enterprise Modeling" , University of Espírito Santo, Brazil ,12th international IEEE enterprise distributed object computing conference , 2008

[58]. K, Kozaki; E, Sunagawa; Y, Kitamura; R, Mizoguchi ; " Role Representation Model Using OWL and SWRL"- ISIR'07 conference , Osaka University , Japan ,2007.

[59]. F, Steinmann; " A radical Revision of UML's Role Concept ", Proceeding of the 3rd International Conference. on the Unified Modeling Language UML, Springer ,2000.

[60]. D, Wagelaar ; " Composition techniques for rule-based model transformation languages ".In ICMT '08: Proceedings of the 1st international conference on Theory and Practice of Model transformations , pages 152-167,Berlin,Heidelberg ,Springer-Verlag , 2008.

[61]. J-M, Jézéquel ; G, Perrouin ; " Vers des Lignes de Produits Flexibles - Apports de l'Ingénierie Dirigée par les Modèles à la Dérivation de Produits"; Équipe Projet Triskell IRISA/INRIA Rennes - Campus de Beaulieu 35042 Rennes, France, 2007.

[62]. P, Borba ; L, Teixeira; R, Gheyi.; "A Theory of Software Product Line Refinement " In A, Cavalcanti, D, Deharbe, M.-C, Gaudel, and J, Woodcock, Editors, Theoretical Aspects of Computing (ICTAC 2010 Conference).Springer Berlin/Heidelberg, 2010.

[63]. N, Anquetil; U, Kulesza; R, Mitschke; A, Moreira; J-C,Royer; A, Rummler; A, Sousa; "A Model-driven Traceability Framework for Software Product Lines"in Software and Systems Modeling Journal, Janvier 2010.

[64]. K, Schittkowski ; PCOMP: " A Modeling Language for Nonlinear Programs in Modeling Languages in Mathematical Optimization", Kluwer Academic Publishers, 2004.

[65]. C, Parra; X, Blanc; and L, Duchien ; "Context Awareness for Dynamic Service-Oriented Product Lines" , Software Product Line Conference '2009 –SPLC'09-San Francisco – California , 2009.

[66]. G-A, Parada; E, Siegert; L-B, de Brisolara; " Generating Java Code from UML Class and Sequence Diagrams",Brazilian symposium of computing system engineering , Brazil.2011.

[67]. E, Dincel ; N, Medvidovic ; and A, Van der Hoek ; " Measuring product line architectures," in PFE '01: Revised Papers from the 4th International Workshop on Software Product-Family Engineering. London, UK:Springer-Verlag, vol.2290, pp. 346–352. , 2002.

[68]. C-W, Krueger ; " New Methods in Software Product Line Development ", Software Product Line Conference SPLC'06, IEEE, pp. 95- 102, 2006.

[69]. C, Berger ; H, Rendel ; and B, Rumpe; " Measuring the Ability to Form a Product Line for a Set of Existing Products," in Proceedings of 4th International Workshop on variability modeling of software - intensive systems-2012.

[70]. C, Berger ; H, Rendel ; B, Rumpe ; C, Busse ; T, Jablonski ; and F, Wolf ; " Product Line Metrics for Legacy Software in Practice ", Software Engineering , RWTH Aachen University , Germany , Electronics Development , Volkswagen AG Business Unit Braunschweig , 2011.

[71]. D, Fischbein ; S, Uchitel ; and V, Braberman; "A Foundation for Behavioural Conformance in Software Product Line Architectures" in Proceedings of ISSTA on Role of Software Architecture for Testing and analysis. ROSATEA'06.New York, USA:ACM, pp.39–48, 2006.

[72]. T, Zhang ; L, Deng ;J, Wu ;Q, Zhou ; and C, Ma ;"Some Metrics for Accessing Quality of Product Line architecture" in Proceedings of Computer Science & Software Engineering International Workshop,Vol. 2, pp. 500–503, 2008.

[73]. Y, Djebar ; MT, Kimour; and N, Guersi ; " A feature Model Metrics-Based Approach to Develop a Software Product Line " in International Arab Journal of Information Technology – IAJIT, Vol 3-2017 First Online-May 2016.

[74]. A, Van der Hoek ; E, Dincel ; and N, Medvidovic ; " Using Service Utilization Metrics to Assess the Structure of Product Line Architectures " in Proceedings of 9th International Symposium on Software Metrics. METRICS '03: Washington, DC,USA:IEEE Computer Society,pp.298., 2003.

[75]. A, Macheto; and A, Trentini ; " A Framework to Build Quality Models for Web Applications " in International Arab Journal of Information Technology, vol.4, no.2, pp.168-176, 2007.

[76]. Vranic V. and Marco V., "Developing a Product-Line Based Architecture in a Domain under research," Tools for Acquisition, Organization and Presenting of Information and Knowledge, Research Project Workshop (NAZOU), in conjunction with ITAT 2006, Slovak University of Technology At Bystrá dolina, Nízke Tatry, Slovakia, 2012.

[77]. J.S, Her ; J.H, Kim ; S.H, Oh ;S.Y, Rhew ; and S.D, Kim ; "A Framework for Evaluating Reusability of core asset in product line Engineering" Informatics Software Technology Elsevier., vol. 49, issue. 7, pp.740–760, 2007.

[78]. K, Khalfaoui K; A, Chaoui; C, Foudil; and E,Kerkouche ; " Automatic Generation of SPL Structurally Valid Products Using Graph Transformations Approach" in Proceedings of International Conference of Computer Science and its Applications CIIA'2013 Saida University ,Springer-MAAACA, vol.488,2013 pp. 333-342, 2013.

[79]. S, Mann ; and Rock G; "Dealing with Variability in Architecture descriptions to support automotive PL: Specification and analysis methods " in Proceedings of Embedded World Conference ., Nurnberg , Deutschland, WEKA Fachmedien, Mar.3-5, 2009.

[80]. A, Hubaux ; P, Heymans ; P-Y, Schobbens; D, Derrider ; and E-K, Abbasi ; " Supporting Multiple perspectives in feature-based configuration Software System Model "-Springer-Verilag Vol 270,pp. 91-102, 2011.

[81]. G, Perrouin ; " Architecting Software Systems Using Model Transformation and Architectural Frameworks ", PhD thesis, FSTC, Université du Luxembourg, et Institut d'Informatique, Université de Namur , Sept, 2007.

[82]. P, Sanchez ; " A Meta-Model for designing Software architectures of aspect oriented SPL " in Proceedings of 8th International Workshop on Aspect-Oriented Modeling (AOM) at 5th AOSD Conference , 2007.

[83]. Y, Djebar; MT, Kimour ; “Managing Software Product Line Architectural Variability with VMS Tool” dans le Séminaire National de l’Informatique de Biskra - SNIB’2010, du 02 au 04 Novembre 2010- Université Mohamed Khider – Biskra- Algérie.

[84]. PureSystems; Pure : :Variants ; site Web; http://www.pure-systems.com/, Pure : :Variants , 2009.

[85]. Y, Djebar and MT, Kimour ; “Un Modéle Conceptuel basé sur les Vues pour la gestion de la variabilité architecturale dans les Lignes de Produits logiciels ” , ICAI09, BBA, Algeria , 2009.

[86]. T. Ziadi and Jean-Marc Jezequel, Manipulation de Lignes de Produits Logiciels : Une approche dirigée par les modèles, IDM 05, 1ère Journées sur l'Ingénierie Dirigée par les Modèles, Paris 2005

[87]. T, Ziadi ; “Manipulation des Lignes de Produits en UML” - *Thèse de Doctorat*, Université de Rennes1- France, 2004.

[88]. Object Management Group OMG ; “ OMG Unified Modeling Language Specification , Version 1.5”, Technical Report, March 2003.

[89]. OMG , “Unified Modeling Language Specifications version 2.1.2”, Object Management Group - formal 2007-11-02 , 2007.

[90]. Object Management Group OMG ; “ Unified Modeling Language Specification Version 2.0 ” Superstructure” ; Technical Report PCT/03-08-02, OMG, 2003.

[91]. ESAPS Styles ; “ Structures and Views for Handling Commonalities and Variabilities ” , available at : http://www.esi.es/esaps , last visited 2001.

[92]. ESAPS;“Engineering Software Architectures, Processes & Platforms for System-Families- Introduction to domain analysis” - ESAPS Deliverables,2001.

[93]. B, Douglass-Editeur(s) : Newnes; Auteur(s) :; Collection : Embedded Technology ; Profil : Niveau : “ illustrating the various aspects of UML and its application to real-time and embedded systems”. Groupe Eyrolles , 1999.

[94]. D, Doose ; Z, Mammeri ; “Acquisition de données du capteur à l’ordinateur”, Dunod, Février - 2003. in Incremental validation of real-time systems - Schedulability analysis and design of real-time embedded systems with partition , 2003.

[95]. B, Rieder ; P, Puschner ; “Cross-Platform Verification Framework for Embedded Systems”. Real-Time Systems 18(2/3): IFIP International Workshop on Software Technolgies for Embedded and Ubiquitous Systems SEUS 2007: Software Technologies for Embedded and Ubiquitous Systems ; pp 137-148

[96]. A, Benatitallah ; P,Kadionik ; F, Ghozzi ; P. Nouel ; N, Masmoudi ; H, Levii “An FPGA Implementation of HW/SW Codesign Architecture for H.263 Video Coding ”, International Journal of Electronics and Communications” , December 2006.

[97]. J, Turley ; “ The two Percent Solution , Embedded Systems Programming” , ESC’2002 conference San Francisco – USA- December 2002.

[98]. Noyau GNU /LINUX, http://www.kernel.org , Last visited 2016.

[99]. H, Falk ; P, Marwedel ; “ Utilitaire GNU”, Proceedings of the 10th International Workshop on Software and Compilers Embedded Systems”, Nice, France, April 2000.

[100]. A, Selic (1992); Octopus (Awad, 1996); RT UML (Douglass, 1998) ; Real-Time Object - Fernandez , Wu & Hancock 173 Oriented Modeling (ROOM) , 1999.

[101]. S, Gérard ; E. Pelachi, P. Petterson; and B, Josko ;" Methodology for Developing Real Time Embedded systems",. Conference: Advances in Object-Oriented Information Systems, OOIS 2002 Workshops, Montpellier, France, September 2, 2002, Proceedings-p.158 , 2002.

[102]. D-W, Franke ; M.K, Purvis ; “ Hardware-Software Codesign : a Perspective Workshop of Software Engineering" , California, USA; pp 344-352, 1991.

[103]. W-H, Wolf ; "Hardware/Software Codesign", IEEE Codesign conference, Vol. 82, no 7, p. 967-989, Juillet1994.

[104]. A, Kalavade; J.L, Pino ; E.A , Lee ; “ Managing Complexity in Heterogeneous System Specification, Simulation, and Synthesis ”,in Proceedings de la conference Internationale sur l’Acoustique, Speech, et le traitement du Signal (ICASSP), Detroit, Michigan, Mai, 1995

[105]. J, Bayer; O, Flege; P, Knauber; R, Laqua; D, Muthig; K, Schmid; T, Widen; and J.-M, DeBaud. PuLSE: “ A Methodology to Develop Software Product Lines ” . In Proceedings of the Fifth ACM SIGSOFT Symposium on Software Reusability (SSR’99), pp 122–131, Los Angeles,CA, USA, May 1999. ACM.

[106]. C, Atkinson; J, Bayer; C,Bunse; E, Kamsties; O, Laitenberger; R, Laqua ;D, Muthig; B, Paech; J, Wüst; and Jörg Zettel. “Component-based Product Line Engineering with UML” .Component Software Series. Addison-Wesley, 2001.

[107]. D, Weiss; “Software Synthesis: The FAST Process”. In Proceedings of the International Conference on Computing in High Energy Physics (CHEP),September 1995.

[108]. “ Domain Analysis (FODA) - Feasibility Study ”. Technical Report CMU / SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University,November 1990.

[109]. I, Jacobson; M, Griss; and P, Jonsson. “ Software Reuse Architecture , Process and Organization for Business Success”. Addison-Wesley, 1997.

[110]. M, Griss; J, Favaro; and M, d’Alessandro ; “ Integrating Feature Modeling with the RSEB ”. In Proceedings of the Fifth Intern.Conference on Software Reuse, pp 76–85, Vancouver, BC, Canada, June 1998.

[111]. Kyo C,Kang; J, Lee; and P, Donohoe; “Feature-Oriented Product Line Engineering . IEEE conference–Software ,19(4):58–65, July/August 2002.

[112]. L, Hotz ; T, Krebs ; and A, Gunter; “ Knowledge -Based Configuration for Configuring Combined Hardware / Software Systems ” . In Proceedings of Workshop Planen, Scheduling und Konfigurieren, Entwerfen Puk 2002, pages 61–70, october,2002.

[113]. SPLIT-Daisy ; W, El Kaim ; “ Managing Variability in the LCAT SPLIT / Daisy ”. In Proceedings of Product Line Architecture Workshop. The First Software Product Line Conference (SPLC1), 2000.

[114]. Bergey J., Cohen S., Donohoe P., and Little R., *Software Engineering Institute (SEI) of Carnegie Mellon Repository, Software Product Lines: Report of the 2010 U.S. Army Software Product Line Workshop*, 2010.

[115]. N, Loughran ; “ Language Support for Managing Variability in Architectural Models (VML) ” ; Lancaster University , LA1 4WA, UK ,2008. International Conference on Software Composition , ICSC 2008: Budapest-,Hungary - pp 36-51

[116]. M, Acher; P, Collet; P, Lahire; and R, France , “ Composing Feature Models ” , International Conference on Software Language Engineering , ICSLE 2009: Denver, CO, USA - pp 62-81

[117]. Y.Djebar,“ Un Modèle Logique pour les Architectures Logicielles dans l’Embarqué ”,Mémoire présenté en vue de l’obtention du diplôme de Magister en Informatique - Option : Informatique Industrielle , Université Badji Mokhtar de Annaba , 2010.