République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université SAAD DAHLAB de Blida

Faculté des Sciences

Département d'informatique

Mémoire en vue d'obtention du diplôme de Master

Spécialité : ingénierie du logiciel

THEME:

Etude et implémentation d'une ligne de produit pour l'eAPC

Présenté par :

M^{me} : Kibiche Meriem

Sous l'encadrement du :

M^{lle} GUENDOUZ Amina

Sous la direction du promoteur :

M^{r} BENNOUAR Djamel

2012 / 2013

Dédicace

Á ma défunte mère que DIEU la garde dans son vaste paradis

Á mon Père Aucune dédicace ne saurait exprimer l'amour, L'estime, le dévouement et le respect que j'ai toujours eu pour vous.

Á mon mari Tes sacrifices, ton soutien moral et matériel, ta gentillesse sans égal, m'ont permis de réussir mes études.

A mon cher frère Amine, les mots ne suffisent guère pour exprimer l'amour et l'affection que je porte pour vous.

Á ma très chère belle-mère Je vous dédie ce travail avec tous mes vœux de bonheur, de santé et de réussite.

Á Mon cher petit frère et A ma très chère petite sœur vous été présents dans tous mes moments

Je vous souhaite un avenir plein de joie,

de bonheur

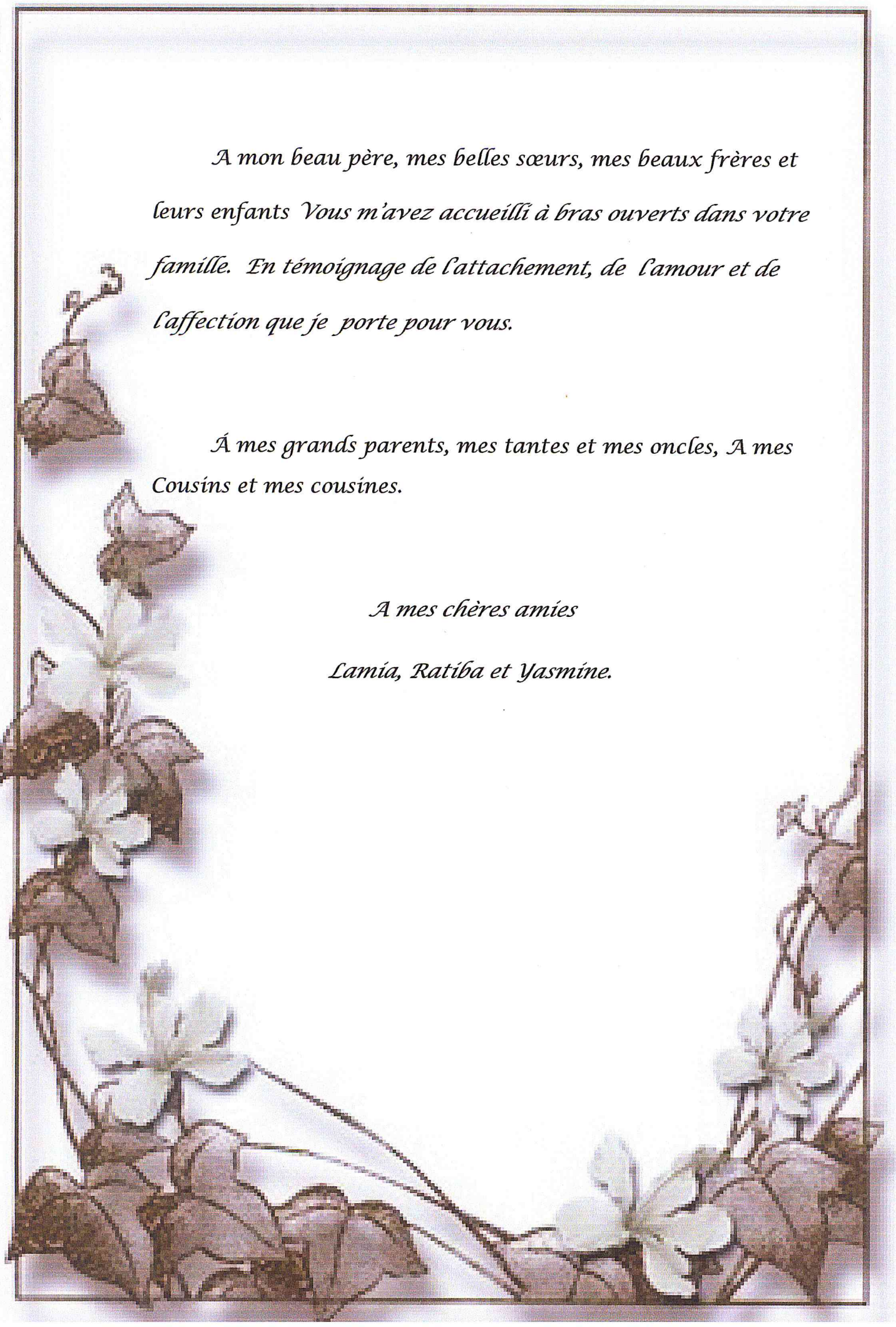
et de réussite

A mon beau père, mes belles sœurs, mes beaux frères et leurs enfants Vous m'avez accueilli à bras ouverts dans votre famille. En témoignage de l'attachement, de l'amour et de l'affection que je porte pour vous.

Á mes grands parents, mes tantes et mes oncles, A mes Cousins et mes cousines.

A mes chères amies

Lamia, Ratiba et Yasmine.



Remerciements

Mes grands remerciements au tout puissant DIEU qui nous a donné la force et la volonté pour pouvoir compléter ce travail.

Mes remerciements à mon mari pour ses sacrifices, son soutien moral et matériel,

Mes remerciements au président du juré ainsi qu'à tous ses membres qui m'ont fait l'honneur d'examiner cette thèse.

Je remercie le promoteur Mr : BENNOUAR Djamel pour ses orientations et son aide précieuse.

Sans oublier de remercie Mlle : GUENDOUZ Amina qui m'a aidé avec beaucoup d'attention et de dévouement et à qui je souhaite une bonne réussite.

Et finalement mes remerciements vont à toute personne qui m'a aidé de près ou de loin.

Résume

La réutilisation logicielle est une préoccupation fondamentale et récurrente depuis l'origine du génie logiciel. Son objectif principal est de réduire les coûts de développement logiciel en favorisant la réutilisation d'éléments logiciels préexistants. Actuellement l'approche des lignes de produite logicielle et d'architecture logicielle sont répondre conjointement à ces objectifs.

Notre travail consiste à mettre sur pied une ligne de produits logiciel pour la gestion des APCs. dans le but de produire des services plus efficaces aux citoyens toute en gagnant en termes de temps et de coût. Cette ligne permet par la suite de générer des applications différentes selon les besoin des utilisateurs.

Mots clés : ligne de produit logiciel, Composant, Architecture logicielle, eAPC, Réutilisation.

Abstract

Software reuse is a fundamental and recurring concern since the beginning of software engineering. Its main objective is to reduce the cost of software development by promoting the reuse of existing software components. Currently the software product line approach and software architecture can jointly meet these objectives.

Our aim is to develop a software product line for APCs management. in order to produce more efficient services to citizens all by winning in terms of time and cost. This line eventually generates different applications according to the needs of users.

Keyword : Software Product Line, Component, software architecture, IASA, eAPC.

ملخص

إعادة استخدام البرمجيات هو الشاغل الأساسي و متكرر منذ بداية هندسة البرمجيات. هدفها الرئيسي هو خفض تكلفة تطوير البرمجيات من خلال تشجيع إعادة استخدام مكونات البرامج الموجودة .حاليا تسمح خطوط انتاج البرمجيات وهندسة البرمجيات من تلبية هذه الأهداف.

مهمتنا هي تطوير خط انتج لإدارة البلديات . من أجل تحقيق خدمات أكثر كفاءة للمواطنين تحقق لهم اربحا في الاوقات والتكاليف. هذا الخط يسمح في نهاية المطاف من توليد تطبيقات مختلفة وفقا لاحتياجات المستخدمين

الكلمات المصنفة : خط إنتاج البرمجيات, مكونات ,هندسة البرمجيات, eAPC , IASA.

Liste des figures

Sommaire

Introduction générale

I. Généralité

La réutilisation logicielle est une préoccupation fondamentale et récurrente depuis l'origine du génie logiciel. Son objectif principal est de réduire les coûts de développement logiciel en favorisant la réutilisation d'éléments logiciels préexistants. Actuellement deux grandes approches semblent être très prometteuses pour répondre conjointement à ces objectifs: l'approche dite de LIGNE DE PRODUITS et l'approche de conception de système logiciel par assemblage de composants, plus connus sous le nom d'ARCHITECTURE LOGICIELLE.

L'approche LIGNES DE PRODUITS LOGICIELS vise à systématiser la réutilisation tout au long du processus de développement logiciel : de la définition des besoins jusqu'au code final et aux plans de test.

Une Ligne de Produits logiciels (LdP) est définie comme un ensemble de systèmes partageant un ensemble de propriétés communes et satisfaisant des besoins spécifiques pour un domaine particulier. La notion de variabilité est utilisée pour regrouper les caractéristiques qui différencient les produits de la même famille. La gestion de cette variabilité est la première activité pour le développement de lignes de produits. La deuxième activité concerne la construction d'un produit particulier (on parle aussi de dérivation de produit) qui consiste en particulier à figer certains choix vis-à-vis de la variabilité définie dans la LdP.

L'architecture Logicielle ou la conception à base de composant logiciel est une nouvelle discipline dans le génie logiciel. Son objectif principal est de permettre la conception par assemblage de composant. La spécification d'architecture logicielle couvre aujourd'hui les phases abstraites et concrète d'un logiciel. Le développement de logiciel se fait souvent par raffinement successif, à partir d'une spécification très abstraite, pouvant être représentée par une spécification dans un langage formel, en

aboutissant à une étape concrète représentée par une spécification à base de composants dits primitifs. Ces derniers sont réalisés soit à l'aide d'un langage de programmation tel Java ou à l'aide d'un langage de description d'architecture logiciel tel qu'ArchJava ou Fractal.

L'approche Ligne de Produits logiciels s'inspire des principes que nous retrouvons dans l'industrie automobile (le Fordisme), aéronautique et électronique, ou l'activité de production est une activité d'assemblage à base de composants. Il devient donc intéressant pour une ligne de produit de se baser sur une approche adéquation avec ces principes de base qui est l'approche architecture logicielle.

Une ligne de produits logiciels regroupe un ensemble de produits appartenant à un même domaine et caractérisé par des éléments logiciels très proches. Un domaine est un secteur de métier ou de technologies ou de connaissances caractérisé par un ensemble de concepts et de terminologies admises par les utilisateurs de ce secteur. Dans notre projet, le nous avons choisi les applications de gestion d'état civil comme domaine pour notre ligne de produit.

II. Présentation du sujet

Notre travail consiste à mettre sur pied une ligne de produits logiciel pour la gestion des APCs. Cette ligne permet par la suite de générer des applications différentes selon les besoin des utilisateurs.

II.1. Problématique

Les services d'état civil connaissent une utilisation croissante des technologies d'information et de communication, dans le but de produire des services plus efficaces aux citoyens toute en gagnant en termes de temps et de coût. De nos jours, la problématique n'est plus de développer un seul logiciel à la fois mais plutôt de développer une famille de logiciels pour répondre plus rapidement aux demandes croissante.

La problématique principale de ce travail est de concevoir est implémenter une ligne de produit pour les applications d'eAPC. D'autre part, l'utilisation d'une

approche orientée objet pour l'implémentation des applications risque de limiter la possibilité de réutilisation, l'introduction d'autres paradigmes et approches devient nécessaire. Ce qui pose un autre problème qui doit être étudié,

II.2. Objectifs

L'objectif du travail consiste à réaliser les diverses études en vue de mettre sur pied une ligne de produits pour les applications liées à la gestion de l'état civil et qui représentent ce que nous appellerons une eAPC. La ligne de produits permettra de générer des eApplication pour une eAPC.

Il est donc nécessaire de comprendre très bien comment fonctionnent les divers services d'une APC, notamment ceux liés à l'état civil, afin d'extraire les similarités et les aspects non similaire entre les applications d'eAPC.

Par la suite il sera nécessaire de déterminer une architecture de référence pour les eApplications, l'architecture doit être orientée composant. Et les composants conçus vont être réalisé par la suite pour construire la base de composants réutilisables.

Enfin, il sera en question de mettre de dériver quelques applications on se basant sur l'architecture de référence et la base de composants réutilisables.

III. Organisation du mémoire :

Le présent mémoire est structuré en cinq chapitres comme suit :

Chapitre 1 : Dans ce chapitre nous allons donné un aperçu sur la réutilisation logiciel son historique et ses intérêts, par la suite on va parlé de l'approche des lignes de produit logiciel ses concepts et la gestion de variabilité.

Chapitre 2 : Dans le deuxième chapitre on va donné les concepts de base de l'orienté composant et on va parlé de trois modèles à base de composant.

Chapitre 3 : Nous allons faire une étude détaillée du domaine de notre ligne de produit qui est la gestion d'état civil, et on a mis l'accent sur les différentes fonctions de l'APC.

Chapitre 4 : Dans ce chapitre nous allons présenter la modélisation de notre ligne eAPC on va utiliser la méthode FORM pour la modélisation de la ligne, IASA et diagramme d'OVM pour la modélisation de l'architecture orientée composants.

Chapitre 5 : Dans le chapitre 5 nous allons testé le bon fonctionnement de notre ligne par la dérivation de deux applications avec des caractéristiques différentes.

Et nous allons terminer le mémoire par une conclusion générale.

Chapitre 1

La réutilisation logicielle et l'approche des lignes de produits logiciels

1.1. Introduction

Le besoin d'améliorer la qualité et la productivité relatives au développement et à l'entretien d'un système d'information a toujours été une préoccupation pour les gestionnaires en informatique. Parmi les solutions proposées pour atteindre cet objectif, le concept de la réutilisation logiciel est considéré comme un moyen privilégié. Beaucoup d'auteurs en génie logiciel voient dans la réutilisation l'occasion d'améliorer la qualité et la productivité relative au développement et à l'entretien d'un système d'information.

Ces besoins ont donné naissance à l'approche des lignes de produits logiciels qui vise à systématiser la réutilisation tout au long du processus de développement logiciel, de la définition des besoins jusqu'au code final et aux plans de test.

1.2. Historique

Depuis l'origine du génie logiciel la réutilisation logicielle était et est toujours une préoccupation très importante et fondamentale. Au fil des années, plusieurs méthodes de développement logiciel se sont appropries ces principes et ont identifié : les éléments logiciels à réutiliser, les mécanismes de réutilisation (héritage, génération, instanciation, plugins,...) et les phases du cycle de vie logiciel durant lesquelles ces mécanismes de réutilisation sont exploitables.

Au début la méthode de développement était procédurale et l'élément réutilisable était les fonctions et procédures. Par la suite chaque décennie avait son approche de développement et son élément réutilisable :

Par exemple : on a réutilise les classes pour les méthodes basées sur l'approche orientée objet et les composants pour les approches orientées composant...

Mais les résultats attendus afin d'améliorer la productivité et la réduction des coûts n'étaient pas suffisamment significatifs, malgré la multitude d'approches et d'outils ayant fait leur apparition sur le marche. Malheureusement le mécanisme de

réutilisation le plus usité est de « copier coller » des morceaux de code [1]. Cette réutilisation était limitée et opportuniste. Elle se base principalement sur l'expérience du développeur qui identifie un morceau de code, une librairie ou une classe qui ont été développés dans un projet précédent afin de résoudre un problème similaire.

1.3. Les concepts fondamentaux des lignes de produit logiciels

L'approche des lignes de produits logiciels adapte le principe de l'industrie automobile au développement logiciel, différents modèles de voitures sortent des mêmes chaines de montage utilisant les mêmes châssis, les mêmes moteurs et les mêmes plans de tests. Du point de vue purement logiciel, la problématique est d'éviter de redévelopper à partir du zéro un logiciel pour chaque nouveau besoin [1].

En conclusion, ces différents logiciels sont regroupés dans une ligne de produits logiciels dans laquelle on différencie:

- Les éléments logiciels communs « **similaire** » pour tous les membres de ligne de produit.
- Les éléments logiciels variant d'un membre de la ligne de produits à l'autre, aussi appelés « **variabilités** ». Ces variabilités peuvent dépendre de différents facteurs.
- Les **contraintes** existant entre les éléments logiciels. Ces contraintes devront être respectées lorsque ces éléments seront assemblés. Elles ont pour origine principalement des considérations techniques ou des règles imposées par le métier.

Les éléments logiciels sont construits à différents niveaux du cycle de vie logiciel et comprennent notamment des exigences, des schémas de conception (des algorithmes à l'architecture), du code, des programmes de tests, de maintenance, etc.

Au final, le développement d'un nouveau logiciel se limite principalement à combiner ces éléments logiciels en respectant les contraintes techniques et métier.

Une ligne de produits logiciels regroupe un ensemble de produits appartenant à un même domaine et caractérisé par des éléments logiciels très proches. Un domaine est un secteur de métier ou de technologies ou de connaissances caractérisé par un ensemble de concepts et de terminologies admises par les utilisateurs de ce secteur. Une ligne de produits a pour but la mise en commun des travaux de développement, de tests et de maintenance de ces éléments logiciels communs de façon à réduire les coûts de production et de maintenance, réduire les temps de production et améliorer la qualité.

1.4. L'ingénierie des lignes de produit

De point de vue de l'ingénierie logicielle dite classique un processus de développement itératif est composé généralement de quatre étapes successives :

Etape1 : la définition des besoins (Application Requirements).

Etape 2 : La spécification de la solution permettant de satisfaire ces besoins (Application Design).

Etape 3 : L'implémentation de cette solution (Application Coding).

Etape4 : La phase de test de la solution et son évaluation par rapport aux besoins initiaux (Application Testing).

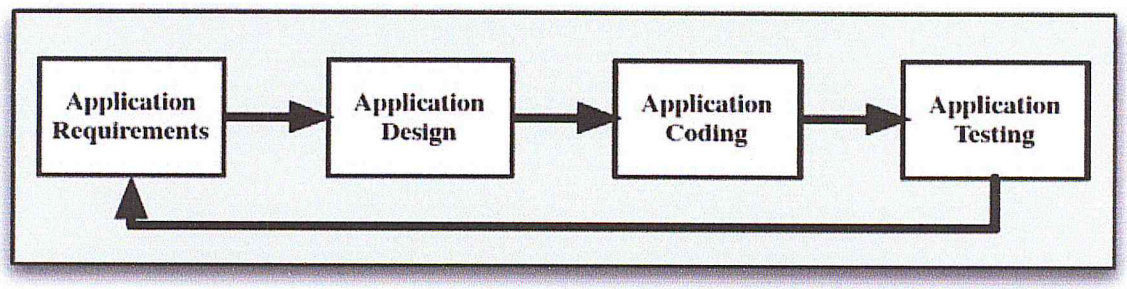


Figure 1.1 : Processus de développement classique [1]

L'ingénierie des lignes de produits logiciel regroupe les activités de développement liées non pas à un logiciel mais à un ensemble de logiciel appartenant à un domaine particulier.

Un processus de développement d'une ligne de produits logiciels est traditionnellement divise en deux processus distincts : Domain engineering et Application engineering :

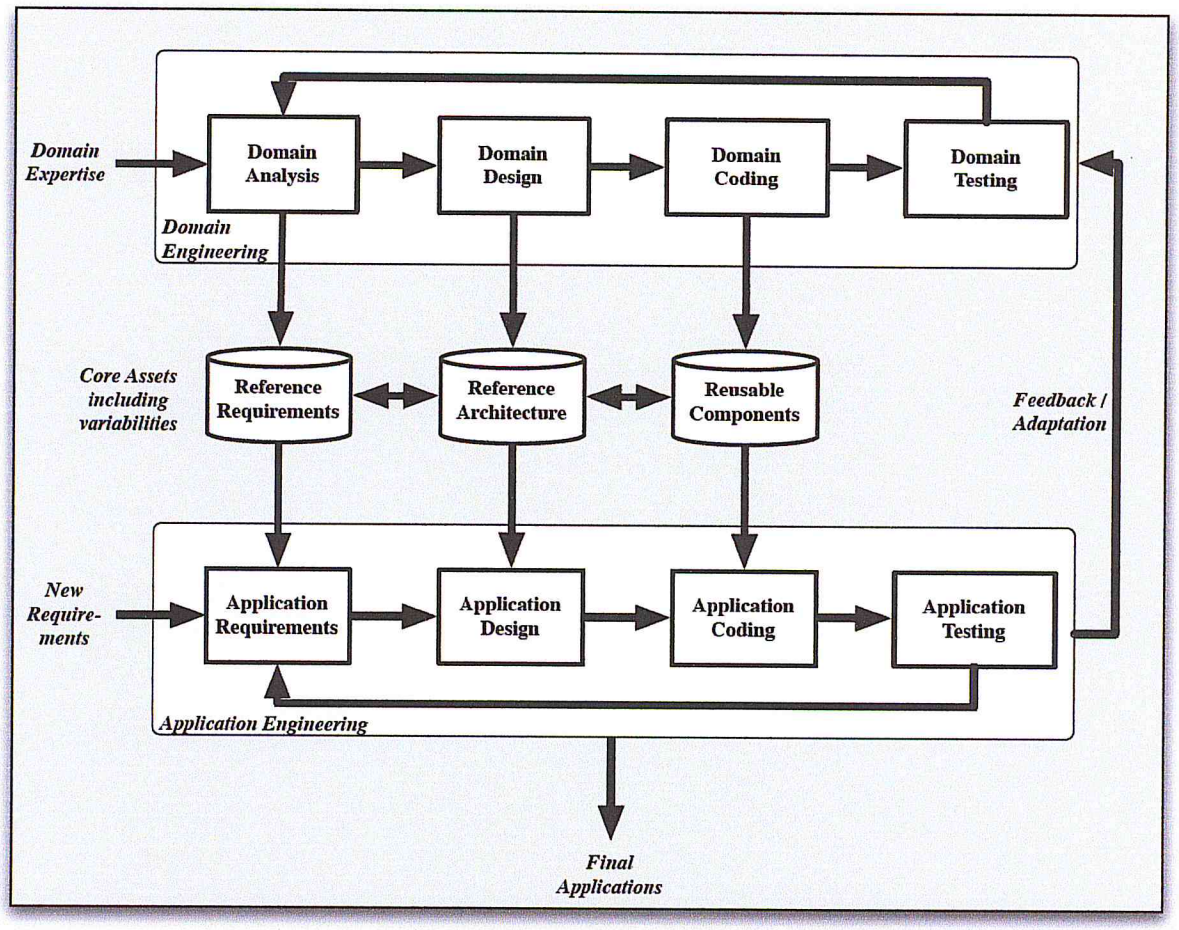


Figure 1.2 : Processus de développement d'une ligne de produit [1]

Le premier processus « **Domain Engineering**» ou «Development for reuse » a pour objectif de construire la base d'éléments réutilisables (« Core Assets »). La principale originalité de ce processus est l'identification des membres de la ligne de produits, des similarités existant entre eux et des variabilités permettant de les différencier. Les principaux outputs de ce processus sont:

Scope : La définition du scope de la ligne de produits consiste souvent en une énumération de noms de produits. La délimitation de ce scope est très importante car un scope trop vaste a tendance à accroître considérablement la complexité et un scope trop restreint ne justifiera pas les investissements consentis pour le développement et la maintenance des éléments réutilisables.

Exigence de référence (Reference Requirement)**:** La définition des exigences de références comprenant les similarités, les variabilités ainsi que les contraintes existant entre elle.

Composant réutilisable (Reusable Component)**:** Un ensemble de composants réutilisables implémentant chaque similarité et variabilités.

Architecture de référence (Reference Architecture)**:** Une architecture de référence décrivant comment ces composants doivent être combinés afin de construire le produit final.

Plan de production (Production Plan)**:** Les plans de production documentant la manière dont l'architecture de référence et les composants réutilisables doivent être utilisés afin de produire efficacement les logiciels finaux.

Le deuxième processus « **Application Engineering** » ou «Development with reuse » a pour objectif de dériver les logiciels finaux à partir des éléments réutilisables. Ce processus s'aligne parfaitement sur le processus de développement classique Cependant, chaque étape est facilitée par la réutilisation des outputs du processus précédent :

Exigences relatives aux demandes (Application Requirements) **:** Durant l'étape de définition des besoins du logiciel final, la réutilisation des exigences de référence permet d'une part de sélectionner les variabilités et d'autre part de se focaliser sur les besoins réellement spécifiques du nouveau logiciel.

Application Design : Concevoir l'architecture du logiciel consiste à configurer l'architecture de référence suivant les variabilités sélectionnées et éventuellement à l'adapter en fonction des besoins spécifiques.

Codage de l'application (Application Coding) **:** L'implémentation se limite principalement à développer de nouveaux composants ou à étendre des composants de référence afin de prendre en compte les besoins spécifiques.

Application Testing : De nombreux tests ont déjà été spécifiés et exécutés pour les composants réutilisables. L'objectif principal de cette phase est de s'assurer que les interactions entre les composants réutilisables et les développements spécifiques ne génèrent pas de comportements inattendus. [1]

Ces deux processus doivent parfaitement s'articuler car ils sont fortement interdépendants. De plus, toutes les lignes de produits logiciels doivent évoluer pour répondre aux besoins de l'utilisateur. Le développement d'un nouveau produit nécessite fréquemment la mise à jour des éléments réutilisables, mise à jour qui impactera l'ensemble de la ligne de produits logiciels. La gestion efficace de ces évolutions n'est envisageable que grâce à des liens de traçabilité reliant ces deux processus ainsi que les éléments réutilisables entre eux.

La complexité de ces processus n'est certainement pas négligeable. Elle dépend principalement du nombre de produits à gérer (scope) et surtout des différences qui existent entre eux (variabilité). La gestion efficace et correcte de la variabilité est un facteur de réussite déterminant lors de l'utilisation d'un processus de ce type.

1.5. La variabilité

Rares sont les éléments logiciels directement réutilisables. Avant de pouvoir réutiliser un élément logiciel, il est nécessaire d'identifier sous quelles conditions celui-ci peut être effectivement réutilisé et comment. Plus l'élément pourra être réutilisé dans des contextes différents, moins son utilisation sera soumise à des conditions et plus sa réutilisabilité augmentera.

La variabilité est la capacité d'un système ou d'un environnement de développement de soutenir la production d'un ensemble d'objets qui diffèrent les uns des autres d'une manière planifiée. [3]

Chaque produit appartenant à une ligne de produits logiciels détermine un contexte particulier, et celui –là est compose d'un ensemble d'éléments logiciels représentés par des « features ».

Les features : L'objectif d'une feature est de délimiter un ensemble d'exigences fortement connexes et directement réutilisables par différents produits finaux. Les features sont successivement décomposées en sous features jusqu'à obtenir des features terminales. Idéalement, chaque feature terminale est associée à un élément logiciel réutilisable (composant, service, …) implémentant les exigences déterminées par le feature correspondante.

En outre, ces features permettent de distinguer les produits les uns des autres. L'analyse de la variabilité permet d'identifier ces features, de spécifier les contraintes les reliant et d'expliciter les alternatives offertes lors de la réutilisation des features. Cependant, de nombreuses sources de variation existent et peuvent apparaître durant toutes les phases du développement logiciel.

En effet, les produits peuvent se différencier par:

- les services qu'ils offrent.
- les structures de données qu'ils manipulent.
- les technologies qu'ils utilisent.
- les flux de contrôles qu'ils doivent respecter.
- les interactions avec leur propre environnement.
- leurs objectifs de qualité, de sécurité.
- …..

1.5.1 La gestion de la variabilité

La gestion de la variabilité joue un rôle essentiel afin de déterminer dans quel contexte, sous quelles conditions et comment une feature peut être réutilisée de manière optimale. La gestion de la variabilité est décomposée en trois activités principales [2] : L'identification, implémentation, l'évolution.

- L'identification de la variabilité qui détermine :
  - Les features terminales qui permettent de distinguer les produits logiciels appartenant à la même ligne de produits (variabilités).
  - les contraintes qui existent entre ces features.
  - où, comment et pourquoi ces variabilités peuvent apparaître (points de variation). Les points de variation correspondent souvent aux features non terminales qui facilitent la structuration des features terminales.
- L'implémentation de la variabilité qui détermine quels mécanismes (héritage, paramètre, configuration, génération, template instanciation, plugins, …) peuvent être utilisés afin de la retranscrire au niveau du code ou de l'architecture.
- L'évolution de la variabilité qui indique comment éviter d'introduire des conflits ou de créer des interactions inattendues entre features lorsque de nouveaux points de variation ou de nouvelles variabilités apparaissent.

La gestion de la variabilité se complexifie très rapidement (exponentiellement) lorsque le nombre de features augmente. C'est pourquoi différents outils logiciels ont été développés afin de faciliter la gestion de la variabilité durant tout le processus de développement.

Les outils de gestion de variabilité offrent principalement trois types de fonctionnalités :

a. Des environnements de développement spécifiques aux lignes de produits logiciels.
b. Des outils de configuration et de gestion du changement destinés aux lignes de produits logiciels.
c. Des outils de tests et de vérification adaptés aux lignes de produits logiciels.

Cependant, même avec les meilleurs outils de gestion de la variabilité, si elle n'a pas été clairement explicitée et documentée dès le départ, sa gestion est clairement impossible. [1]

1.5.2 La modélisation de la variabilité

La modélisation de la variabilité est une technique utilisée afin, d'une part de documenter la variabilité et d'autre part de raisonner sur la variabilité.[1]

Ses principaux objectifs sont :

- Explicitation de la variabilité dès les premières phases du projet.
- Diminution de la complexité liée à la gestion de la variabilité tout au long du processus de développement.

Pour cela de nombreuses méthodes permettent de décrire graphiquement la variabilité. Entre autres **FORM** (Feature-Oriented Reuse Method), c'est une méthode systématique qui cherche à capturer les points communs et les différences des applications dans un domaine; par la suite on utilise les résultats d'analyse pour développer des architectures de domaine et des composants réutilisables. [2]

Il ya trois phases dans la méthode FORM: analyse du contexte, modélisation du domaine (ou feature) et modélisation de l'architecture. Au cours de l'analyse du contexte le scope exacte du domaine est identifié avec son utilisation prévue, et les différentes conditions et leurs interactions possibles avec le monde extérieur est mis; après, durant la phase modélisation du domaine l'utilisateur représente les features déjà identifiées et modélise leurs relations. Un « feature modèle » est crée et de nombreuses spécifications des features cohérentes dérivées du modèle peuvent exister. Une telle spécification de feature est utilisée plus tard pour trouver une architecture de référence correspondante à être réutilisée. [2]

a. L'analyse de contexte et Modélisation du domaine

L'analyse de domaine comprend des activités pour identifier les features des produits, les classer et les organiser en un ensemble de modèles cohérents. Chacune de ces activités est définie ci-dessous :

Identification des features :

L'identification des features implique la connaissance du domaine d'abstraction obtenue à partir des experts du domaine (les utilisateurs, les analystes et les développeurs de systèmes) et d'autres sources telles que les livres, les manuels d'utilisation, des documents de conception et les programmes sources.

Classification des features

Les features sont classées selon les types d'information qu'ils représentent, Les développeurs d'applications font diviser les types de sélection des features en quatre catégories:

- **Les features de capacité** (Capability features) : Ce que le système peut fournir comme services fonctionnels (exprime les fonctions du system, répond au besoin des clients) ou non fonctionnel (n'expriment pas une fonction du logiciel mais plutôt expriment des contraintes sur interface, ou performances).

- **Les feature d'environnement d'exploitation** *(*Operating environment features*)* : représentent les attributs de l'environnement du déploiement du système, inclus à la fois hardware et software (OS, BD, Net …)
- **Les features de la technologie du domaine** (Domain Technology features) : liées directement et spécifiques au domaine d'application.
- **Les features d implémentation** (Implementation features) : représentent les détails d'implémentation à des niveaux inférieurs.

Organisation des features

Les features, en utilisant FORM, sont représentées sous la forme d'un arbre, Différents types de features existent : [2]

- **Root :** ce spécifique feature appelée la racine (*root*) située au sommet de l'arbre. Cette root détermine le point d'entrée du diagramme.
- **optional features :** Si une feature est optionnelle (optional) alors si un de ses parents est sélectionné elle ne l'est pas nécessairement.
- **mandatory features** : Si une feature est obligatoire (mandatory) alors si un de ses parents est sélectionné elle est aussi sélectionnée. La root est par définition toujours obligatoire.

Les features de type père fils sont reliées par défirent types de relations:

- **AND :** Cette décomposition signifie que si le parent est sélectionné alors ses enfants le sont aussi.
- **Alternative :** Cette décomposition signifie que si le parent est sélectionné alors ses enfants peuvent être sélectionnés.
- **implémente-par :** Cette décomposition signifie que le feature fils est implémente par sont parent.

Dans certains cas, on veut pouvoir exprimer des contraintes entre des features qui ne partagent pas le même parent. Ces contraintes peuvent être établies entre toutes les features appartenant au même Feature Diagramme. Elles sont représentées de manière textuelle afin d'éviter tout surchargèrent réduisant la lisibilité du modèle. [1]

Les deux contraintes les plus utilisées sont :

- **Requires**: La contrainte requires entre deux features (f1 requires f2) permet d'exprimer que si la feature f1 est sélectionnée alors f2 doit nécessairement l'être, mais pas inversement.
- **Excludes** : La contrainte excludes entre deux features (f1 excludes f2) permet d'exprimer que si une des deux features est sélectionnée alors l'autre feature ne peut pas l'être en même temps pour le même produit.

Validation du modèle

Le modèle de feature doit être validé avant son utilisation. Il est assez simple. La validation est effectuée par l'instanciation du modèle pour chaque application envisagée dans le domaine d'analyse et vérifier si le modèle instancié représente correctement les features de l'application.

b. Modélisation de l'architecture

En modélisant une architecture, on définir un modèle de composant et on allouer à chaque features un composant du modèle.

1.6. Orthogonal Variability Model

Un modèle de features documente à la fois les features qui font partie de tous les produits d'une ligne de produit logiciel « les features communes» et les features variables ceux qui font partie de certains produits. Par contre, il ne permet pas la modélisation de la variabilité dans les autres vues de système, pour cela nous devons utiliser OVM (Orthogonal Variability Model).

OVM est une approche plus récente utilisée pour documenter la variabilité son principal objectif est de définir explicitement les point de variabilité et les varient d'une ligne de produit logicielle. [4]

L'OVM nous permet aussi de gérer la modélisation de l'architecture où chaque composant réalise une variante n'est prise que dans une application pour laquelle la variante correspondante a été liée. [3]

1.7. Conclusion

L'approche des lignes de produits logiciels est très prometteuse en termes d'économie et de réduction drastique des coûts de développement logiciel. Malheureusement, elle était principalement réservée à certains domaines possédant déjà une grande expérience des lignes de produits « non logiciels » tels que les télécoms ou la construction automobile. Récemment, ces principes ont aussi été appliqués dans d'autres domaines tels que l'e-gouvernement, l'e-commerce ou l'e-banking.

Chapitre 2

Architecture logiciel

2.1. Introduction

La programmation orienté objet représente un progrès important en termes de qualité de programmation et de lisibilité de code mais ne permet pas encore une distinction claire entre le code qui forme la préoccupation centrale de l'application (le code métier) et le code d'interaction avec le système. Ainsi, certains objets ne peuvent être réutilisés à cause de leur forte dépendance avec le système ou de leur forte dépendance avec le reste de l'application.

Aujourd'hui, l'approche architecture logicielle comme nouvelle discipline du génie logiciel, est largement acceptée comme voie prometteuse pour la production de logiciels de haute qualité.

2.2. L'architecture logicielle

L'architecture logicielle est un nouvel axe de recherche dans le génie logiciel. Elle décrit d'une manière symbolique et schématique les différents éléments d'un ou de plusieurs systèmes informatiques, leurs interrelations et leurs interactions. C'est une sorte de génie logiciel basé sur les composants mais un peut plus abstrait car les composants ne sont pas forcement implémenté.

Les deux objectifs principaux de toute architecture logicielle sont la réduction des coûts et l'augmentation de la qualité du logiciel. La réduction des coûts est principalement réalisée par la réutilisation de composants logiciels et par la diminution du temps de maintenance (correction d'erreurs et adaptation du logiciel) et la qualité, par contre, se trouve distribuée à travers plusieurs critères.

2.3. Les concepts fondamentaux d'architecture logicielle

Le composant et le connecteur représentent les concepts de base sur lesquels repose la spécification d'architecture logicielle

2.3.1. Les composants

Un composant représente le concept central en architecture logicielle [9]. Il constitue la brique de base dans le processus d'élaboration de l'architecture d'une application.

Un composant est une unité de composition, développé indépendamment du logiciel dans lequel il sera utilisé. Il peut cependant posséder des caractéristiques qui le rendent réutilisable dans un environnement particulier (i.e. un style d'architecture particulier [10].

Un composant est doté de un ou plusieurs ports (ou interfaces) à travers lesquelles il exprime ses besoins (services requis) et exporte ses services (services fournis) et ses états [5] Il interagit avec le monde extérieur uniquement par ses ports.

Un composant peut être primitif ou composite. Un composite est un réseau de composants interconnectés selon une topologie permettant d'atteindre les objectifs fonctionnels du composite.

2.3.2. Les connecteurs

Vu leur rôle central dans toute architecture et malgré leur importance, les connecteurs n'ont pas reçu une attention suffisante dans les différents travaux de recherche dans le domaine de l'architecture logicielle [6].

Le rôle primordial des connecteurs est d'assurer une communication correcte entre les composants. Un connecteur doit être indépendant de la logique de l'application.

Un connecteur peut être un connecteur d'assemblage ou de délégation :

- ✓ **Un connecteur d'assemblage** permet de connecter un composant qui fournie des services à un composant qui les utilise, on peut en déduire une relation d'*utilisation.*

- ✓ **Un connecteur de délégation** permet de déléguer la réalisation ou le requiert d'un service à un de ses sous-composants, on en déduit une relation de *composition* entre les participants.

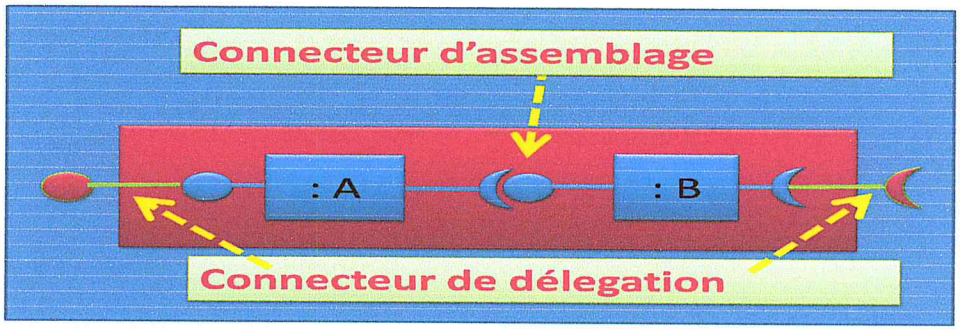


Figure 2.1: types de connecteurs d'un composant

2.4. Les modèle base des composants

2.4.2. Le standards UML2.0

Après le succès de la première version d'UML, la deuxième version notée UML 2.0 introduit la notion de diagramme d'architecture, aussi appelée diagramme de structure composite. UML propose une notation unique pour couvrir toutes les phases du développement logiciel. Avec l'ajout de ce diagramme, UML veut combler une des lacunes de la première version en permettant la spécification de l'architecture d'une application.

Un composant UML 2.0 est une entité modulaire et réutilisable fournissant et requérant des interfaces qui peuvent être potentiellement exposées par l'intermédiaire de ports. Un connecteur est une entité qui relie des ports ou des interfaces de composant. Dans un diagramme d'architecture, une application ou un composant est constitué de l'assemblage d'autres composants par l'intermédiaire de ports interconnectés.

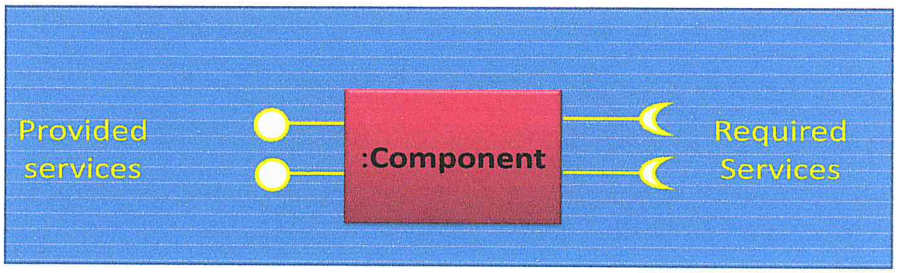


Figure 2.2: Représentation UML d'un composant logicielle

2.4.2. Le modèle a composant *Fractal*

Fractal est un modèle plus léger et plus proche des concepts des langages de programmation. Le modèle de composant Fractal réalisé par France Telecom R&D et par l'INRIA. Fractal vise à autoriser une définition, une configuration et une reconfiguration dynamique d'une architecture à base de composants, ainsi qu'une séparation claire des préoccupations fonctionnelles et non fonctionnelles.

Un composant Fractal est formé d'une membrane et d'un contenu. *La membrane* définit, par l'intermédiaire d'un ensemble d'interfaces, les services requis et fournis par le composant. *Le contenu* d'un composant Fractal permet de différencier deux sortes de composant : les primitifs et les composites.

Le contenu d'un primitif identifie un module logiciel (classe, ensemble de fonctions, *etc.*) permettant de réaliser les services du composant primitif.

Le contenu d'un composite définit un assemblage de composants permettant de mettre en œuvre les services du composite. **[7].**

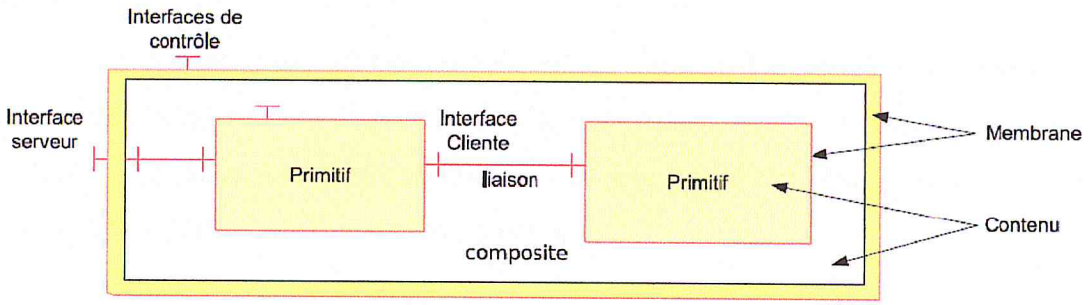


Figure 2.3 : modèle de composant fractal

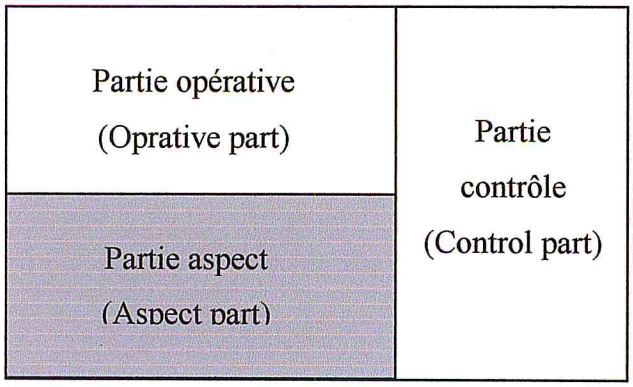


Figure 2.4 : les différentes parties de la vue interne d'un composant selon IASA

La partie opérative: elle contient les composants représentants par leurs interconnexions la logique générale du composite à un moment bien précis.

La partie contrôle: le rôle des composants de cette partie est la réalisation des diverses opération de contrôle sur les composants de la partie opérative et la partie aspect. La partie contrôle contient au moins un composant.

La partie aspect: elle est dédie à contenir les aspects technique d'une application. Une fois ces aspects technique sont instanciés, seront injectés au niveau des composants de la partie opérative.

2.5. Conclusion

Aujourd'hui l'architecture logicielle est reconnue comme une étape fondamentale du développement logiciel, elle apparaît dans tous les projets sérieux. Mais elle reste un domaine récent, il n'existe pas de représentation standard, et aussi il existe peu d'aide pour la conception architecturale ainsi que les méthodes de vérification/validation sont immatures.

Chapitre 3

Introduction au domaine d'étude

3.1. Introduction

Tout d'abord, nous devons donner un aperçu et une meilleure compréhension de ce qu'est la commune et cela facilitera la conception.

La commune algérienne est une institution constitutionnelle, d'après les articles 15 et 16 de la constitution : « Les collectivités territoriales de l'Etat sont la commune et la wilaya… », « l'assemblée élue constitue l'assise de la décentralisation et le lieu de participation des citoyens à la gestion des affaires publiques. »

C'est une collectivité territoriale en bas d'une échelle administrative (après la daïra et la wilaya) créée par la loi.

La commune est l'institution administrative la plus proche des habitants de la commune, ce sont ses services qui s'occupent de la vie quotidienne des citoyens (propreté de la commune, délivre les documents administratifs, …).

3.2. Domaine d'étude

Parmi les services les plus importants de la mairie notre étude s'intéresse particulièrement au service de l'état civil, dont l'organisation est comme suit [10]:

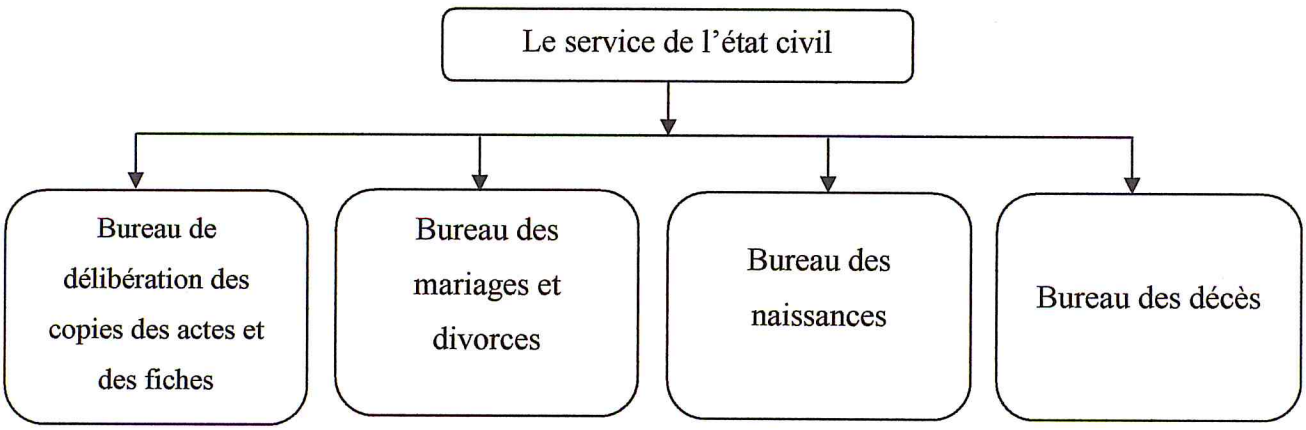


Figure 3.1 : L'organisation du service de l'état

3.3. Gestion du service de l'état civil

La gestion de service d'état civil est base sur :

3.3.1. Les officiers

Le président de la commune peut déléguera un ou plusieurs agents communaux occupants des emplois permanents, âges au moins de 21 ans. Un officier d'état civil peut exerce les fonctionnalités suivantes :

- La réception et la transcription de tout type de déclaration (naissance, décès, mariage,…).
- L'établissement des actes et des certificats à ceux qui ont le droit de requérir des extraits figurant sur les registres.
- Apposer les mentions marginales sur les registres.
- Etablir les statistiques mensuelles, trimestrielles et décennales des naissances, mariage et décès.
- La fermeture et la conservation des registres.

3.3.2. Les registres

Il existe 3 types de registres dans chaque APC le premier pour les naissances, un deuxième pour les décès et le troisième pour les mariages. Un registre concerne une année bien précise et toute les registres son en double.

Au premier janvier de chaque année les registres sont placés au niveau des mairies pour enregistré les événements séquentielles des citoyen. Avant l'exploitation des registres toutes les pages sont numérotées séquentiellement et cacheté par le président de tribunal, chaque registre doit comporter une marge permettent l'apposition des mentions judiciaire.

Les actes sont inscrits sur les registres, de suite, sans aucun blanc ni interligne, il n'y est rien écrit par abréviation et aucun date n'y est mise en chiffre.

A la fin de l'année exactement le 31 décembre les registres doivent êtres fermés. Une copie est conservée au siège de la commune et l'autre copie est conservée au niveau du greffier au tribunal.

3.4. Le fonctionnement dans les services de l'état civil

3.4.1. Les déclarations transcrites sur les registres de l'état civil

a. Déclaration de naissance

Les déclarations de naissance sont établies à l'officier de l'état civil du commun lieu de l'accouchement ou lieu de résidence des parents dans un délai de cinq jours. Respectant ce délai la déclaration sera transcrite directement sur le registre de naissance.

Dépassant le délai la transcription de l'acte portera la mention du jugement.

b. Déclaration de mariage

Si la déclaration du mariage à lieu devant l'officier de l'état civil, avant le mariage et en présence des témoins celui-ci transcrit l'acte sur le champ dans ses registres, il remet aux époux un livret de famille constatant le mariage.

Sinon c.à.d. si la déclaration est faite après un certain temps de mariage, la transcription de l'acte portera la mention de jugement.

c. Déclaration de décès

La déclaration du décès sera directement transcrite sur le registre s'il n'a pas dépassé les 24 heurs après le décès, la déclaration doit être accompagnée par un certificat médical.

Dépassant les 24 heurs la transcription est mentionnée d'un jugement dans le registre.

d. Déclaration de divorce

L'acte du jugement de divorce est obligatoire pour l'établissement d'un acte de divorce.

3.4.2. L'établissement des actes

L'officier d'état civil établit trois types de fichiers d'état civil :

1. Etablissement des actes transcrits sur les registres

a. L'extrait de naissance :

Le demandeur de l'extrait de naissance a le droit de choisir entre sa présence lui-même au service de l'état civil ou l'envoi d'une demande manuscrite par voie postale.

b. Acte de décès

Le demandeur de l'extrait de décès a le droit de choisir entre sa présence lui-même au service de l'état civil de la commune, lieu de décès ou l'envoi d'une demande manuscrite par voie postale.

c. Acte de mariage

Le demandeur de l'acte de mariage a le droit de choisir entre sa présence lui-même au service de l'état civil ou l'envoi d'une demande manuscrite par voie postale.

d. Certificat de non divorce

Ce certificat est établi pour vérifier que la veuve n'a pas divorcé avant le décès de son mari.

2. Etablissement des fiches au vu le livret de famille

a. La fiche familiale

La fiche familiale d'état civil est établie au vu du livret de famille, sans demander d'autres documents, la fiche familiale d'état civil constitue en même temps:

- un extrait de naissance des époux et leurs enfants
- un certificat de vie
- un acte de mariage et un certificat de divorce des époux et, le cas échéant, des actes de décès.

La délivrance de la fiche familiale d'état civil se fait immédiatement et sa duré de validité est d'une (01) année.

b. La fiche individuelle

Pour l'obtention de la fiche individuelle d'état civil, le demandeur doit se déplacer aux services de l'état civil lieu de sa résidence, cette fiche atteste que l'intéressé est en vie.

La fiche individuelle est délivrée immédiatement sur la base du livret de famille.

3. Etablissement des fiches et certificats non transcrits sur les registres d'état civil

a. Le certificat de résidence

La délivrance du certificat de résidence s'effectue sur présentation de l'une des pièces justificatives suivantes : [13]

- Le contrat de location ou un arrêté d'attribution du logement
- La dernière quittance SONELGAZ, ou quittance du service des eaux.
- La décision d'affectation pour les logements de fonction et d'astreinte.
- Une copie du titre de propriété.

Le certificat de résidence doit comporter la mention de sa destination :

- " Valable uniquement pour la carte nationale d'identité "
- " Valable uniquement pour passeport de voyage "
- " Valable uniquement pour la déclaration de création d'un parti politique ".
- …..

La durée de validité de certificat de résidence est d'une (01) année à compter de la date de sa signature.

b. La fiche de résidence

S'agissant d'autres dossiers administratifs, il est procédé à la délivrance d'une fiche de résidence sur une simple présentation par le demandeur, le service de l'état civil se chargera de la transcription de tous les renseignements nécessaires tels que" le prénom, le nom, le degré de la parenté " sur la fiche de résidence qui est contresignée par l'hébergeant.

c. Certificat de preuve d'individualité

En cas des erreurs faites lors de l'établissement des actes transcrits sur les registres un certificat de preuve d'individualité est établi.

Les erreurs sont des erreurs dans les noms ou les dates de naissances, ainsi que la présence des témoins et l'acte de jugement sont obligatoires pour l'établissement de ce certificat.

3.4.3. Mentionner les jugements

Les jugements sont motionnés en général dans les marges des registres

a. Mentionner le naissance

Lorsqu'une naissance n'a pas été déclarée dans le délai légal, l'officier de l'état civil ne peut la relater sur ses registres qu'en vertu d'une ordonnance rendue par le président du tribunal de l'arrondissement dans lequel est né l'enfant, et une mention sommaire est faite en marge de la date de la naissance.

b. Mentionner le mariage

Mentionner dans le registre de naissance et de mariage les informations suivantes :

- Nom et prénom de l'époux.
- Date et numéro de l'acte de mariage.

c. Mentionner le décès

Mentionner la date et numéro d'acte du décès sur la marge des registres de naissance et de mariage.

a. Mentionner et transcrire le jugement de divorce

La date et le tribunal de jugement et le numéro d'acte de divorce seront mentionnés sur le registre de mariage et le registre de naissance.

b. Mentionner et transcrire le jugement de preuve d'individualité

Dans le cas de preuve d'individualité l'ancien et le nouveau (nom, date de naissance) et la date et le tribunal de jugement seront transcrits sur le registre de naissance. [12]

3.5. Conclusion

Dans ce chapitre nous avons fait une étude détaillée sur les fonctions des services d'état civil. Cette étude va nous permettre par la suite de faire la conception du domaine de notre ligne de produit.

Chapitre 4

Ingénierie de domaine

4.1. Introduction

Dans ce chapitre nous allons concevoir notre ligne de produit eAPC.

Comme on a déjà présenté dans le chapitre 1, Nous utilisons la méthode FORM pour modéliser notre ligne de produit. FORM passe par trois étapes : analyse du domaine, modélisation du domaine et modélisation d'architecture. Chacune des étapes est détaillée ci-dessous.

4.2. Analyse du domaine

La première étape de la méthode FORM consiste à faire une étude du domaine de notre ligne de produit logiciel (faite au chapitre 3).

Après l'étude du domaine des APCs on peut décomposer les fonctions de notre ligne de produit en trois types de fonction :

a. Les fonctions liées à l'APC :

   Ce sont les fonctions principales de notre ligne et qui représentent les fonctions des APC ordinaire fournies au citoyen comme :

   - les transcriptions des déclarations(les déclarations de naissance, de mariage, de décès et de divorce)
   - Les transcriptions des mentions (les mentions de naissance, de mariage, de décès et de divorce)
   - l'établissement des actes (touts types d'actes: acte de naissance, de mariage, fiche familiale, …)

b. Les fonctions annexes ou modules :

   - Ce sont les fonctions qui permettent au citoyen de participer à l'amélioration de son APC à travers les publications des documents ou la participation à un sondage ou même de faire des réunions en ligne avec les membres de l'APC ou les citoyens.
   - Les fonctions qui facilitent au citoyen le retrait des formulaires officiels.

c. Les fonctions d'applications:

Ce sont les fonctions de gestions (ces fonctions sont très importantes) :

- gestion des comptes utilisateurs et des profils
- gestion des lieux
- Configuration de système : cette application donne à une application son nom, l'utilisateur principal et la description de la wilaya.
- La communication verticale et horizontale : la communication verticale c'est la communication d'une APC avec d'autres APC d'une même wilaya ou d'autre (en cas des mentions hors wilaya), La communication avec les tribunaux au cas des mentions judiciaires (les quatre type de mentions) et la communication avec les hôpitaux en cas des déclarations des naissances ou décès. La communication horizontale c'est avec la daira, la wilaya ou même avec le ministère de l'intérieur pour la sauvegarde et la récupération des données.
- La gestion des authentifications qui permet l'authentification des actes produits et des utilisateurs et la vérification si un document est falsifies ou pas.
- Le suivi et la validation : ses principaux objectifs sont la validation des déclarations et des mentions en plus des fonctions auxiliaires comme le suivi des responsabilités des utilisateurs (faire le lien entre les fonctions et les utilisateurs qui sont fait pour que chaque utilisateur prend sa responsabilité).et la calibration d'état de sortie qui consiste a ajuste l'impression des actes.

En plus des fonctions :

d. Notre ligne de produit peut générer des applications pour des systèmes d'exploitation différents (sous Unix ou Windows). Et peut utiliser différent type de base de données pour la sauvegarde des données.

e. Pour implémenter les fonctions on peut utiliser différents type d'authentification.

f. Avec trois protocoles de communication ou plus.

4.3. Modélisation du domaine

Aprés l'analyse du domaine on arrive à la deuxième étape de la méthode FORM: la modélisation du domaine (la création de Feature modèle).dans cette étape on peut extraire les features de chaque couche et les lier.

Notation FORM des feature et des relations entre features

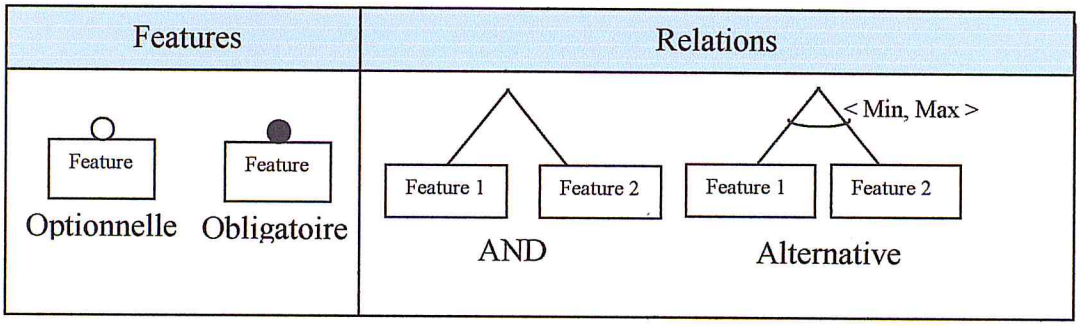


Figure 4.1 : notation de feature modèle selon la méthode FORM

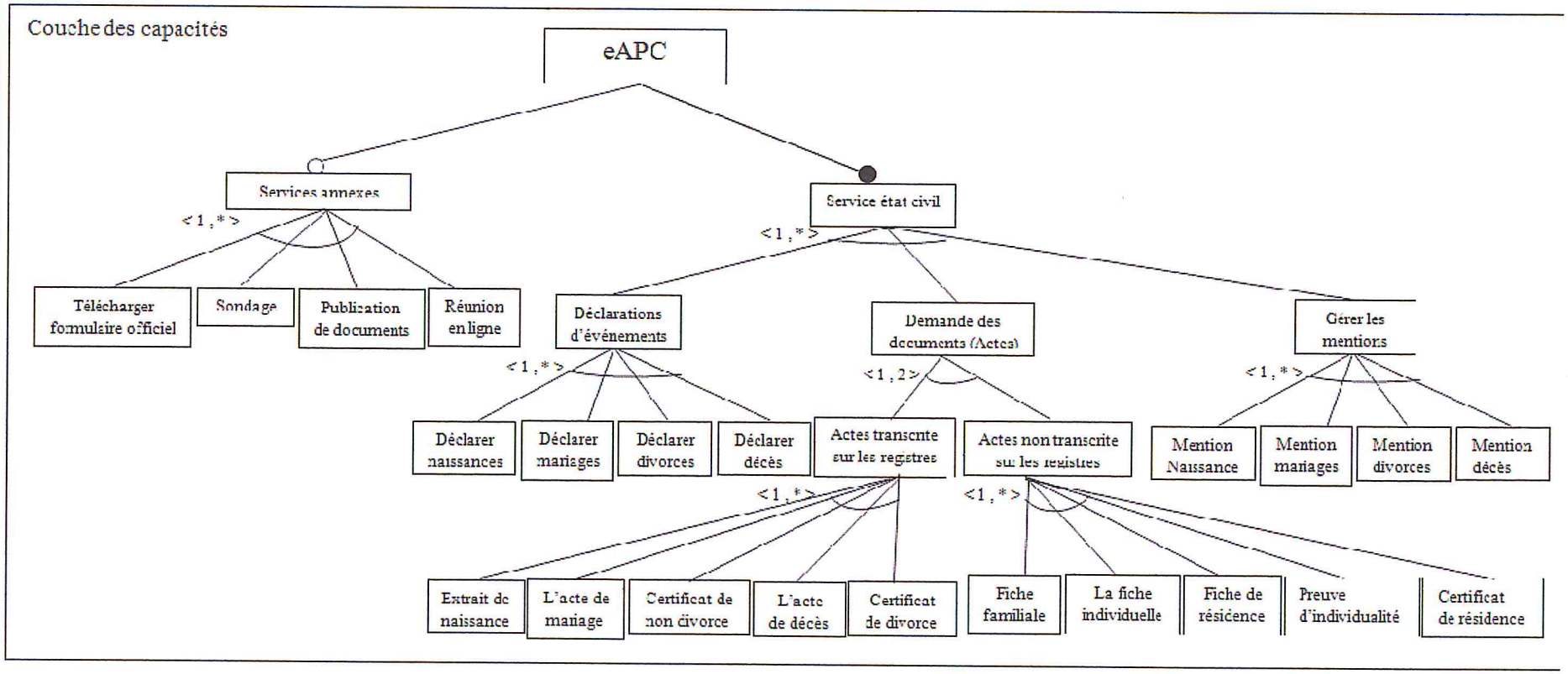
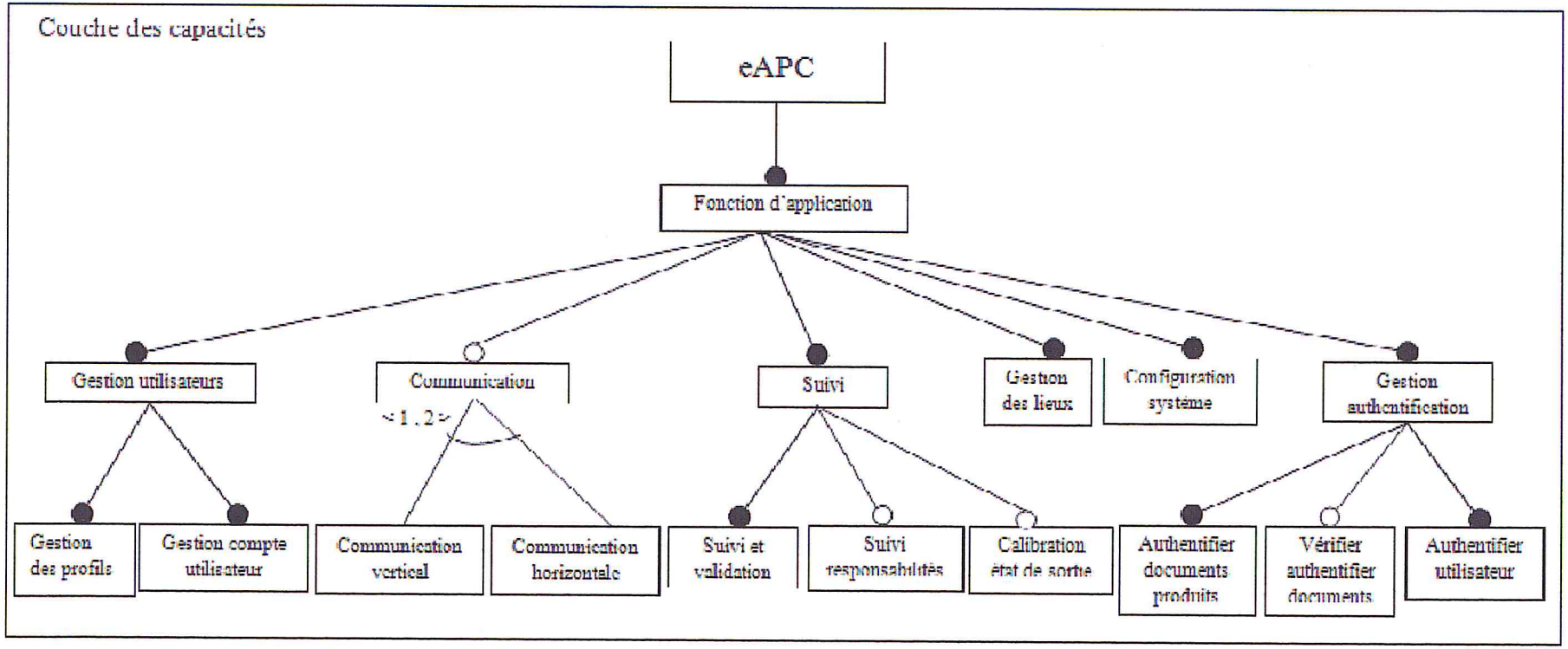


Figure 4.2 : Feature modèle pour la ligne de produit eAPC (1ère partie)



Authentification documents produits Requérir établissement des actes
Suivi et validation Requérir déclaration d'événements
Suivi et validation Requérir Gestion des comptes utilisateur
Gérer les mentions Requérir Communications
Gérer les mentions Requérir Déclaration d'événements

Gestion des profils **Implemented by** Utilisateurs
Gestion authentification **Implemented by** authentification
Communication **Implemented by** Outils de communication

Figure 4.3 : Feature modèle pour la ligne de produit eAPC (2ème partie)

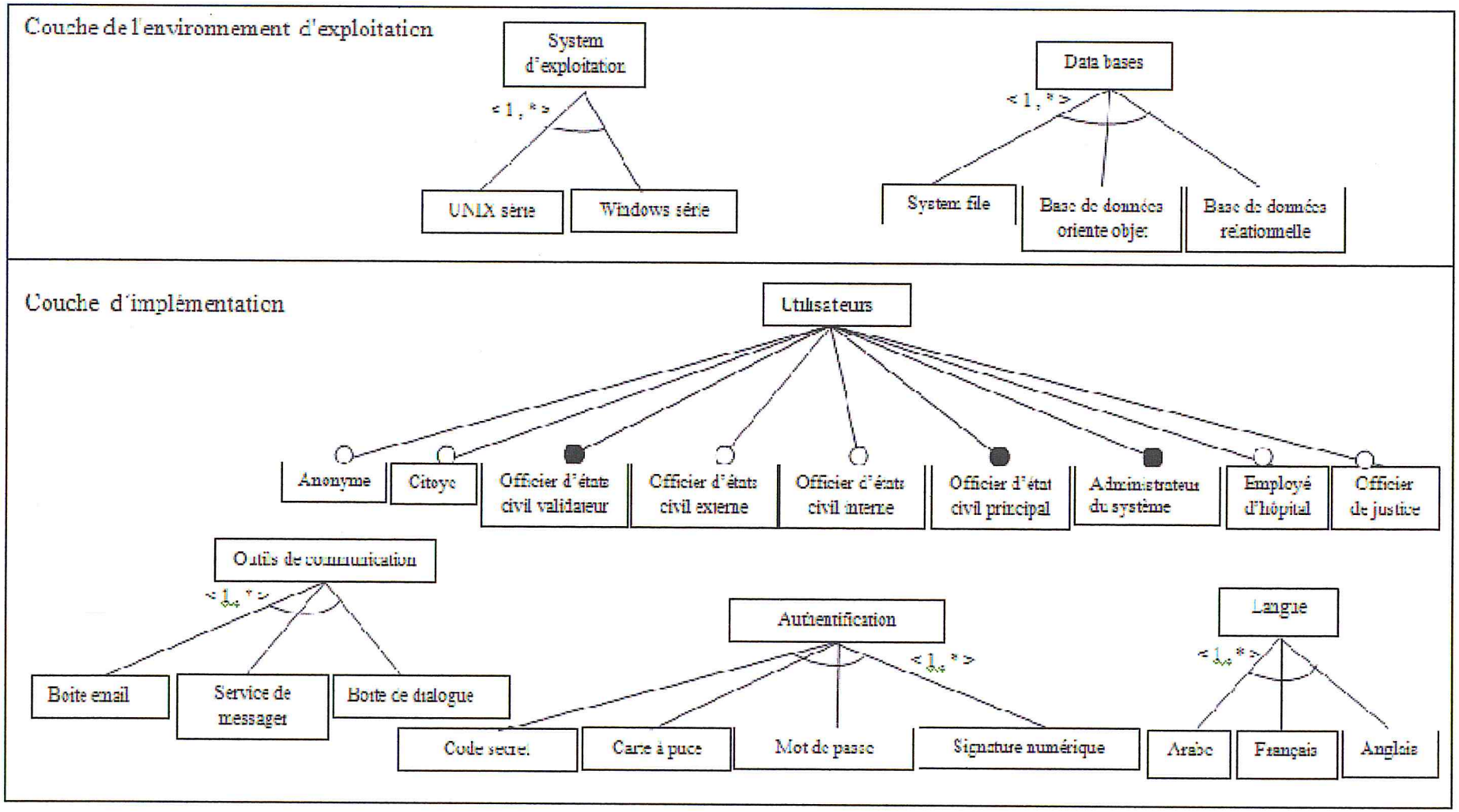


Figure 4.4 : Feature modèle pour la ligne de produit eAPC (3ème partie)

4.4. Modélisation de l'architecture :

Dans la troisième étape nous allons présenter l'architecture de référence à base de composants et connecteurs pour notre ligne de produit. Nous utilisons la notation IASA pour l'architecture et pour représenter la variabilité nous utilisant la notation OVM.

Pour pouvoir modéliser la variabilité dans l'architecture de référence, il faut extraire les points de variabilité et les variantes, pour ce faire, nous avons suivi l'algorithme FM2OVM (Figure 4.4). FM2OVM permet le passage du diagramme de features vers le diagramme OVM [4]. Le modèle cible ne représente que la partie variable liée par la suite à l'aide des liens de traçabilité avec l'architecture, la partie commune est représenté directement par des composants.

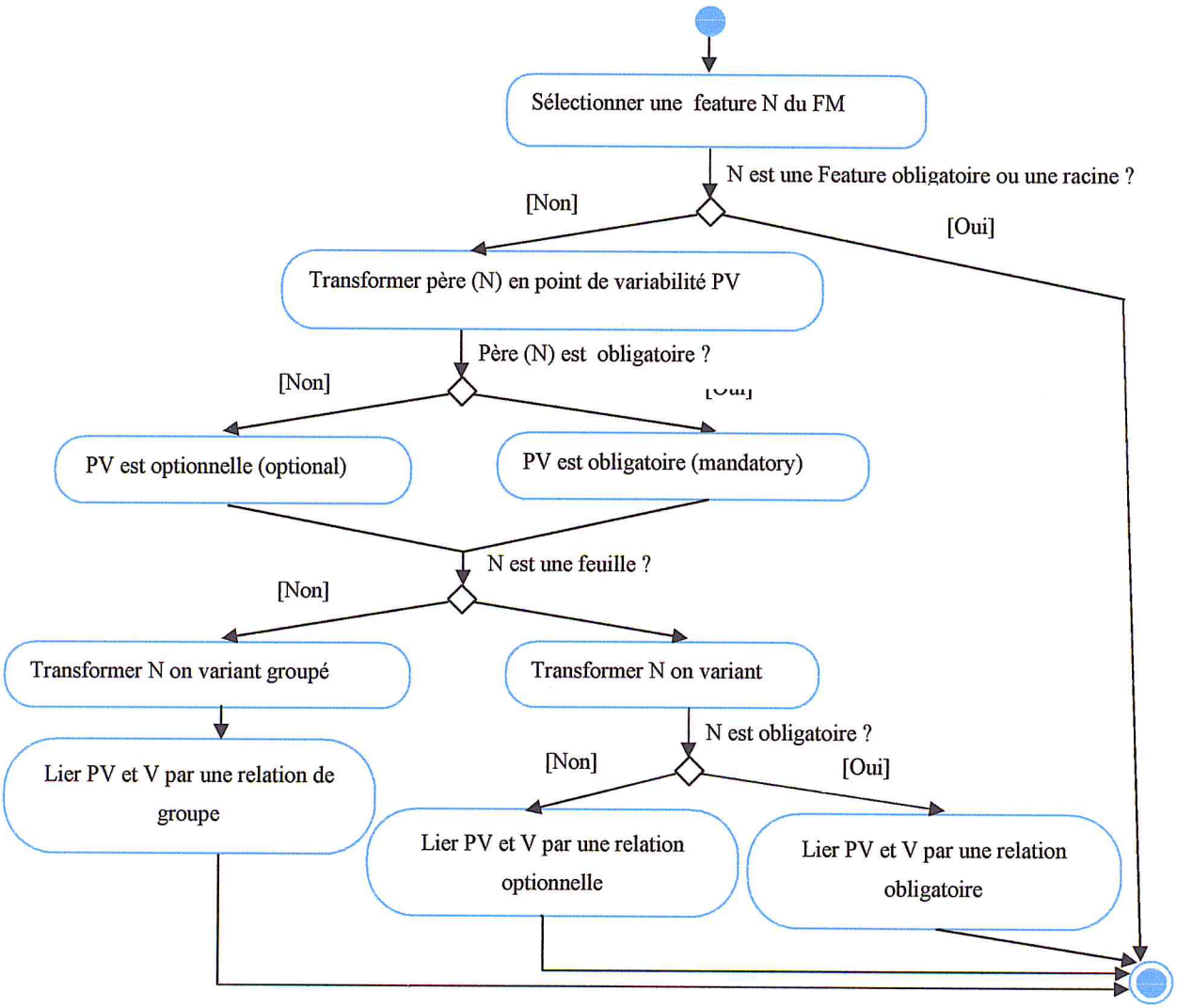


Figure 4.5 : diagramme d'activité FM2OVM [4]

Notation de diagramme de variabilité et du diagramme orient Composant selon IASA

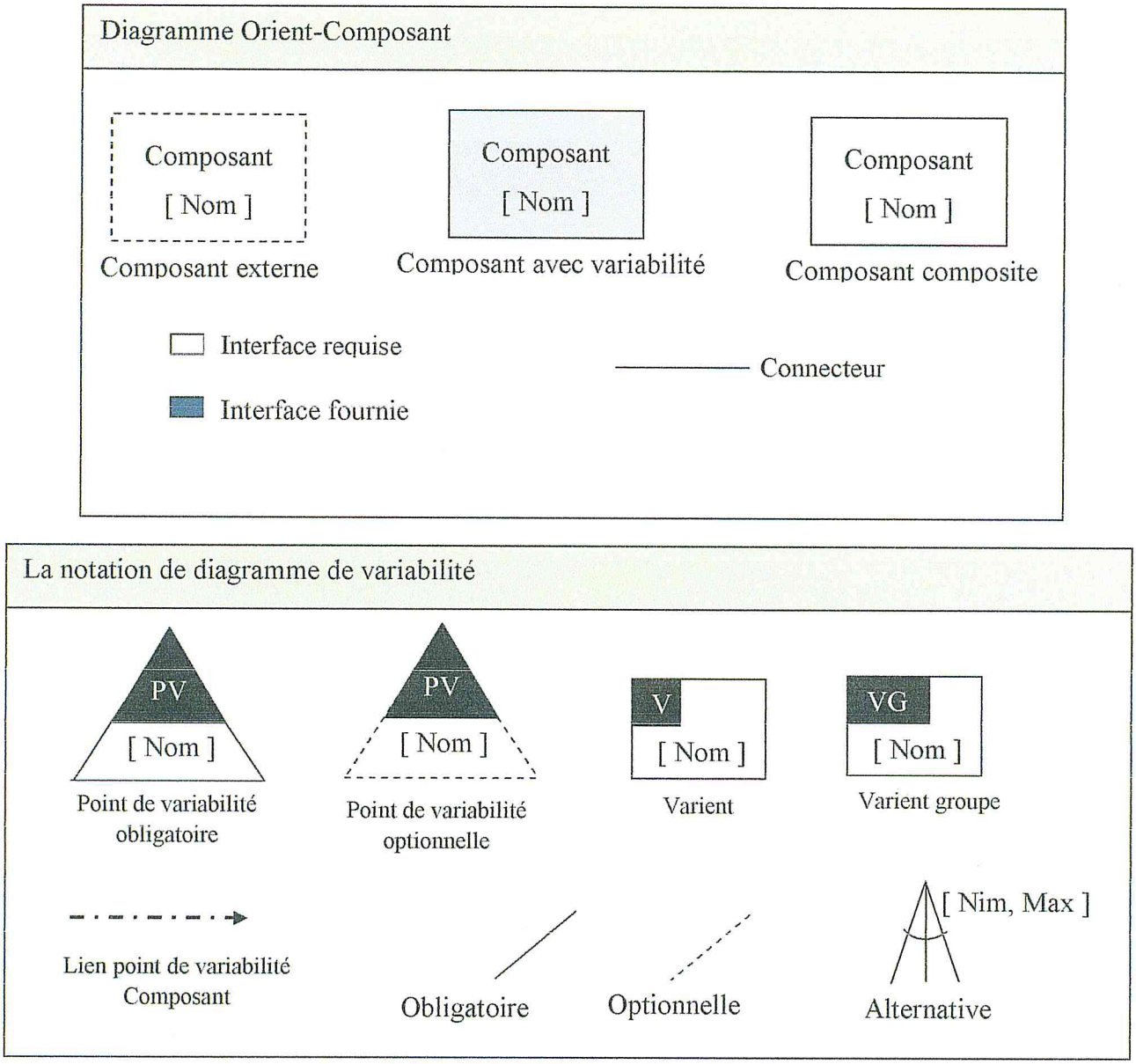


Figure 4.6 : Notation de diagramme de variabilité et de diagramme orient Composant selon IASA

4.4.1. Architecture globale de la ligne de produit APC :

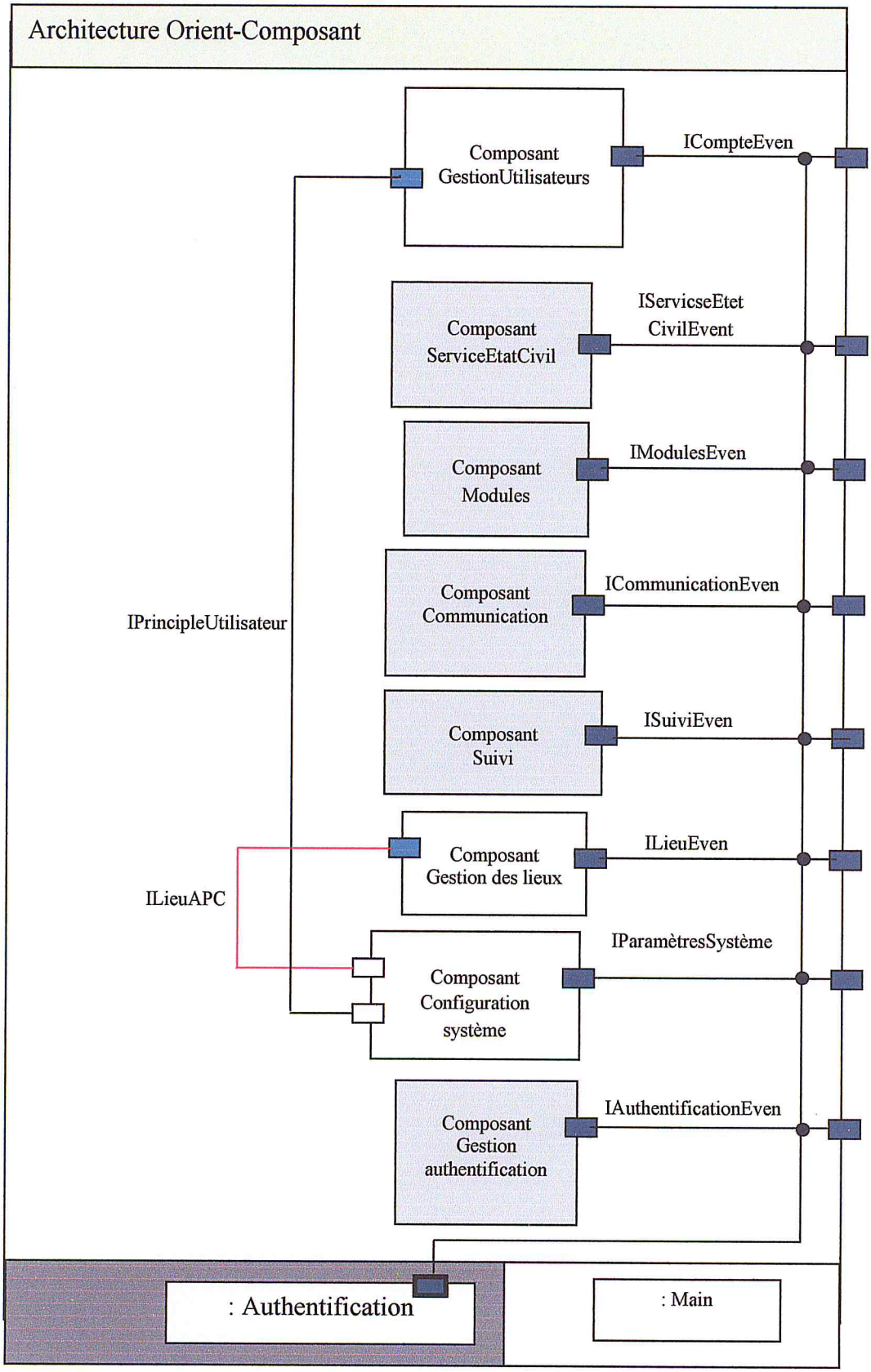


Figure 4.7: Architecture globale de la ligne de produit eAPC

4.4.2. Raffinement des composants

a. Le composant service d'état civil

C'est un composant composite qui s'occupe des fonctions de service d'état civil pour une APC :

1- La gestion des déclarations.
2- La gestion des mentions.
3- La gestion des demandes des actes.

Ces tâches seront effectuées à l'aide des composants ComposantDeclaration_Evenement, ComposantDemandeDesDocuments, ComposantGérerLesMention. Et le composant externe ComposantUtilisateur qui sert à garder l'utilisateur qui a exécuté la fonction demandée.

La modélisation de la variabilité pour le composant service d'état civil nous permet d'extraire plusieurs raffinements. (On peut aller jusque à 8 raffinements différents tel que les trois composants déjà cités au dessus varies d'un raffinements à un autre cependantle ComposantUtilisateur reste commun pour tous les raffinements) par la suite on va donner quatre raffinements différents possibles.

1. Premier raffinement du composant service d'état civil

Dans la première figure (Figure 4.8) on va présenter le composant avec les trois fonctions fondamentale de l'APC (déclarations d'évènements, demande des documents et gérer les mentions).

2. Deuxième raffinement du composant service d'état civil

Par la suite dans la figure (Figure 4.9) un raffinement avec deux fonctions déclaration d'évènements et demande des documents.

3. Troisième raffinement du composant service d'état civil

Le troisième raffinement avec les services déclaration des évènements et gestion des mentions (Figure 4.10)

4. Quatrième raffinement du composant service d'état civil

Le quatrième raffinement avec le composant déclarations d'évènement tout seul. (Figure 4.11).

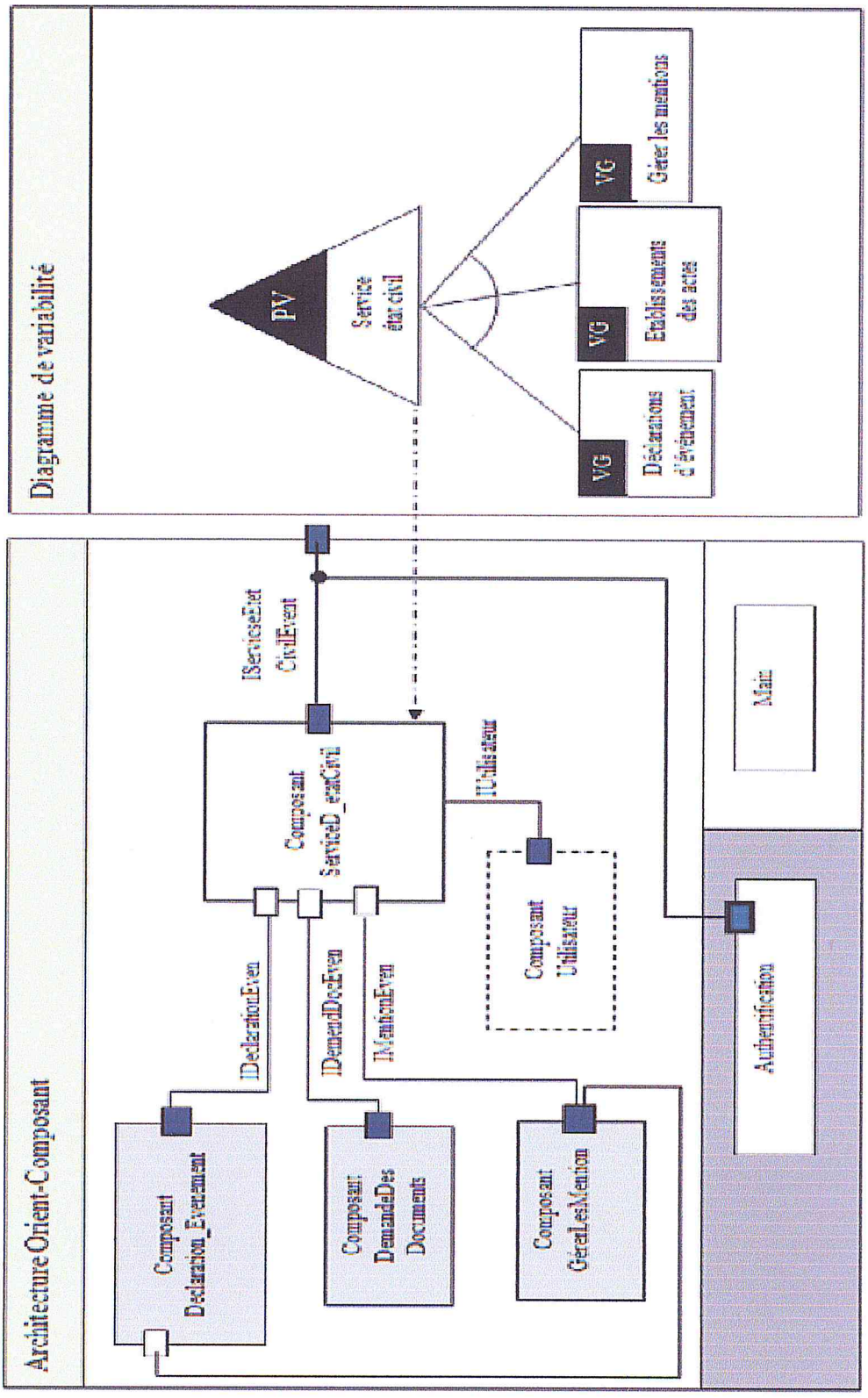


Figure 4.8 : Composant Service d'état civil

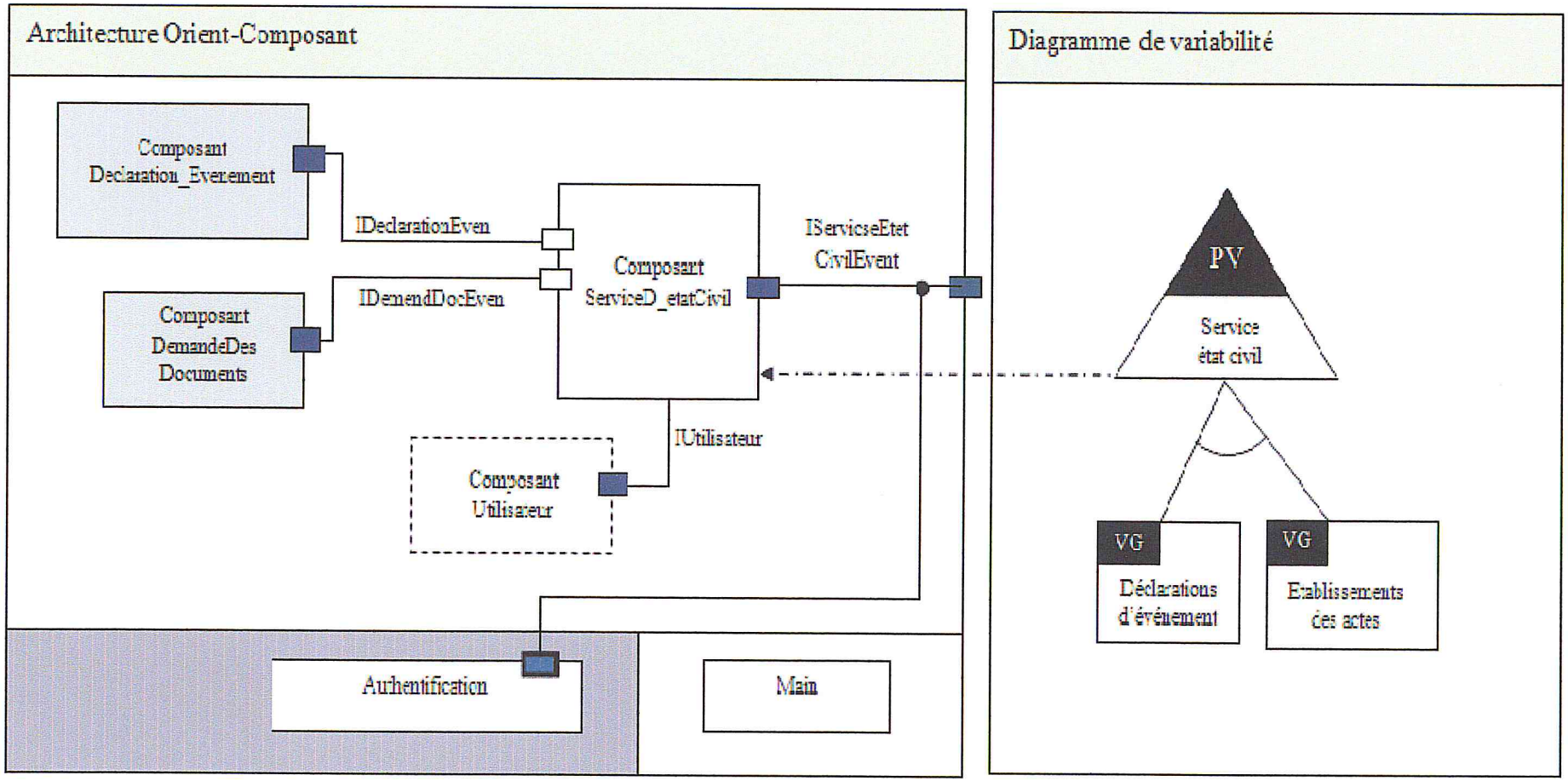


Figure 4.9: Composant service d'état civil avec deux varient déclaration d'événement et demande document

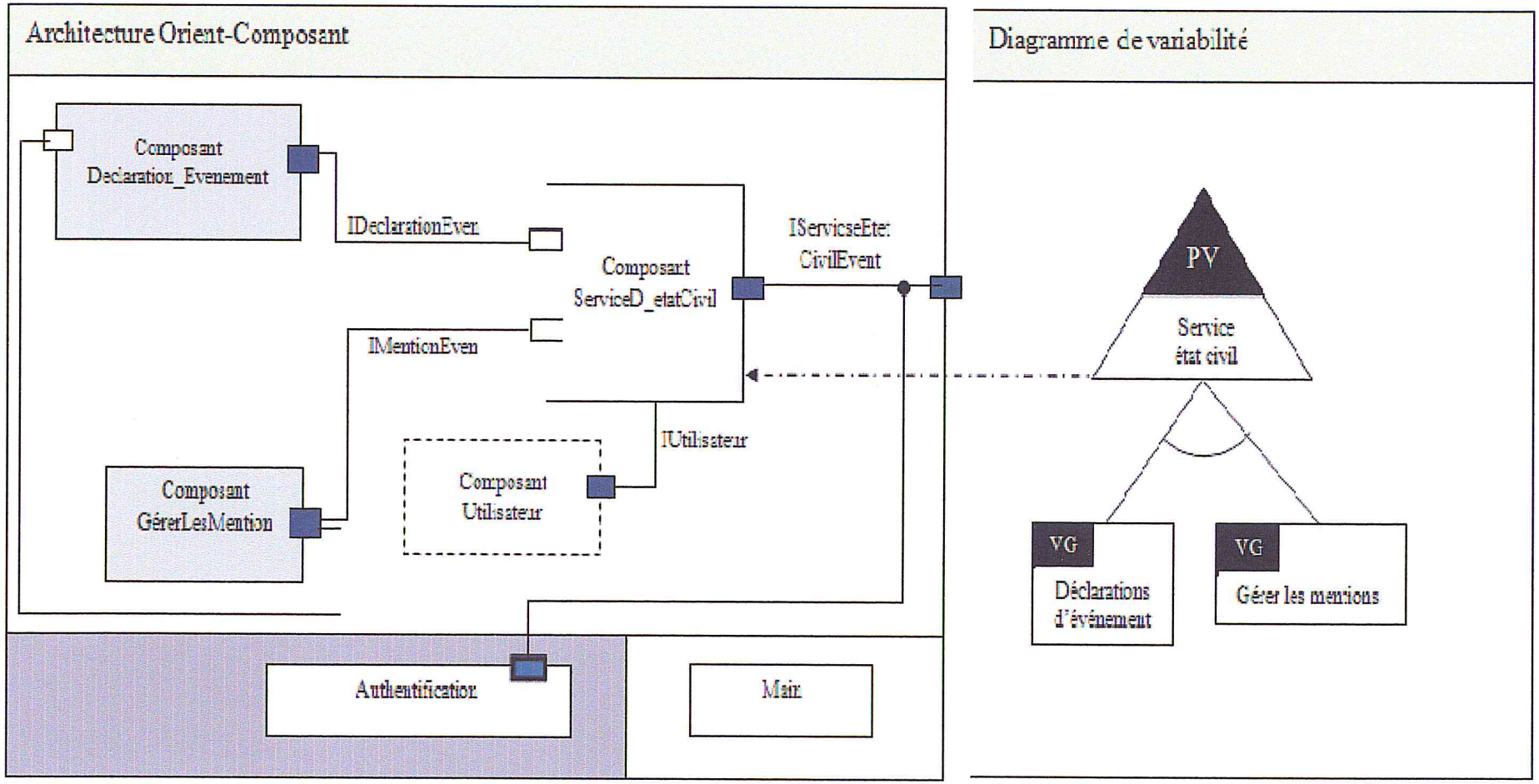


Figure 4.10 : Composant Service d'état civil avec deux varient déclaration des événements et gestion et mentions

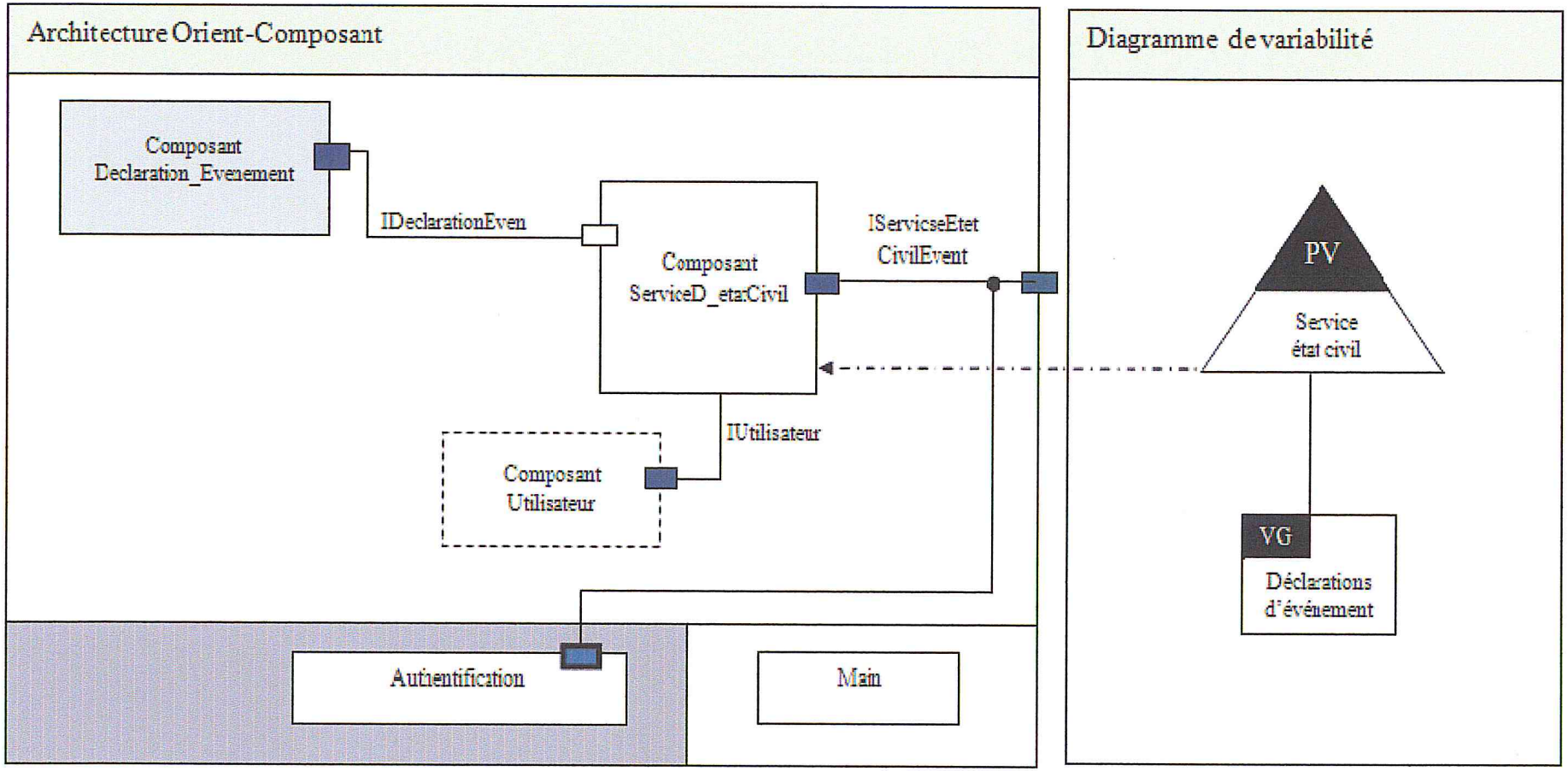


Figure 4.11 : Composant Service d'état civil avec une seul varient déclaration des événements

b. Le composant déclaration d'évènement

Le composant déclaration d'évènement est un composant composite qui se charge de l'ajout, la suppression et la modification des différents types de déclarations (naissance, mariage, divorce, décès).

Ces fonctions sont exécutées à l'aide des composants : ComoposantNaissance, ComposantMariage, ComposantDivorce, ComposantDécès.

On peut avoir jusqu'aux 16 raffinements différents, dans notre cas on nous allons présenter l'exemple général qui comporte tous les composants composites.

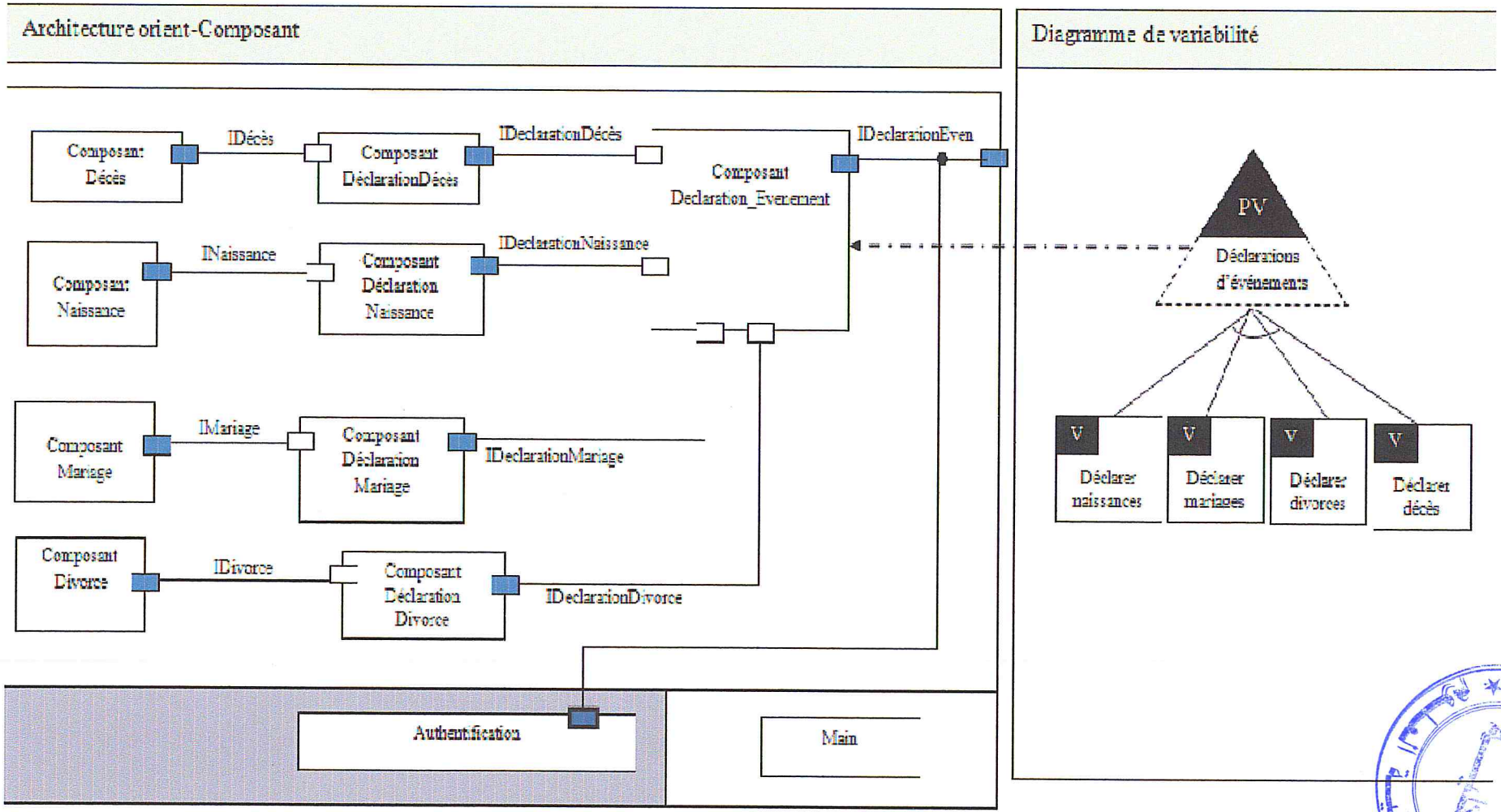


Figure 4.12 : Composant Déclaration d'événement

c. Le composant demande des documents

Le composant demande des documents est un composant composite qui se charge de la gestion d'une demande d'actes. Il gère les demandes des actes transcrits sur les registres et les actes non transcrits.

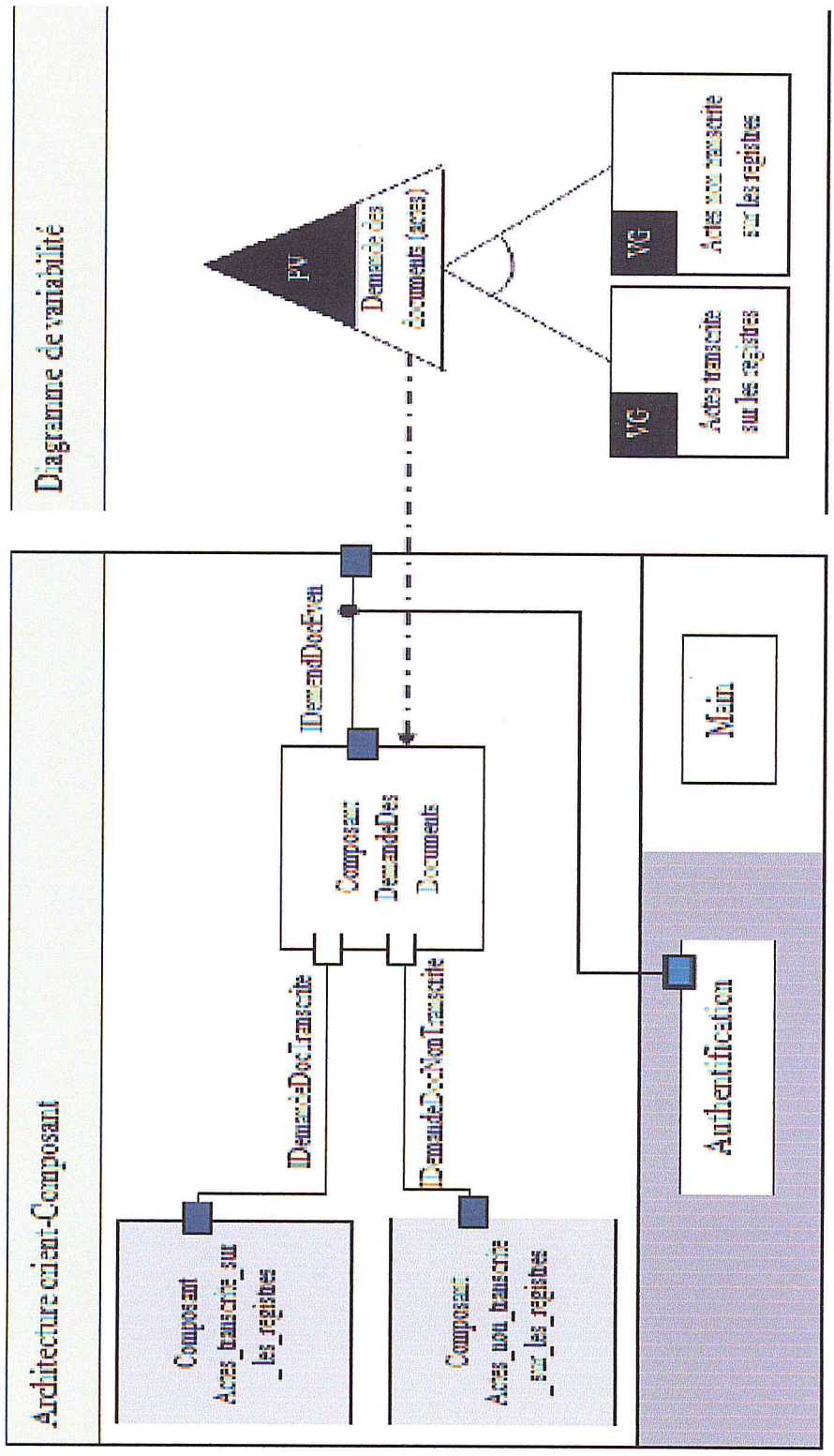


Figure 4.13: Composant demande documents

d. Le composant gérer les mentions

Le composant gérer les mentions est un composant composite qui se charge de l'ajout, la suppression et la modification des différents types de mention (naissance, mariage, divorce, décès).

Ces fonctions sont exécutées à l'aide des composants : ComoposantNaissance, ComposantMariage, ComposantDivorce, ComposantDécès.

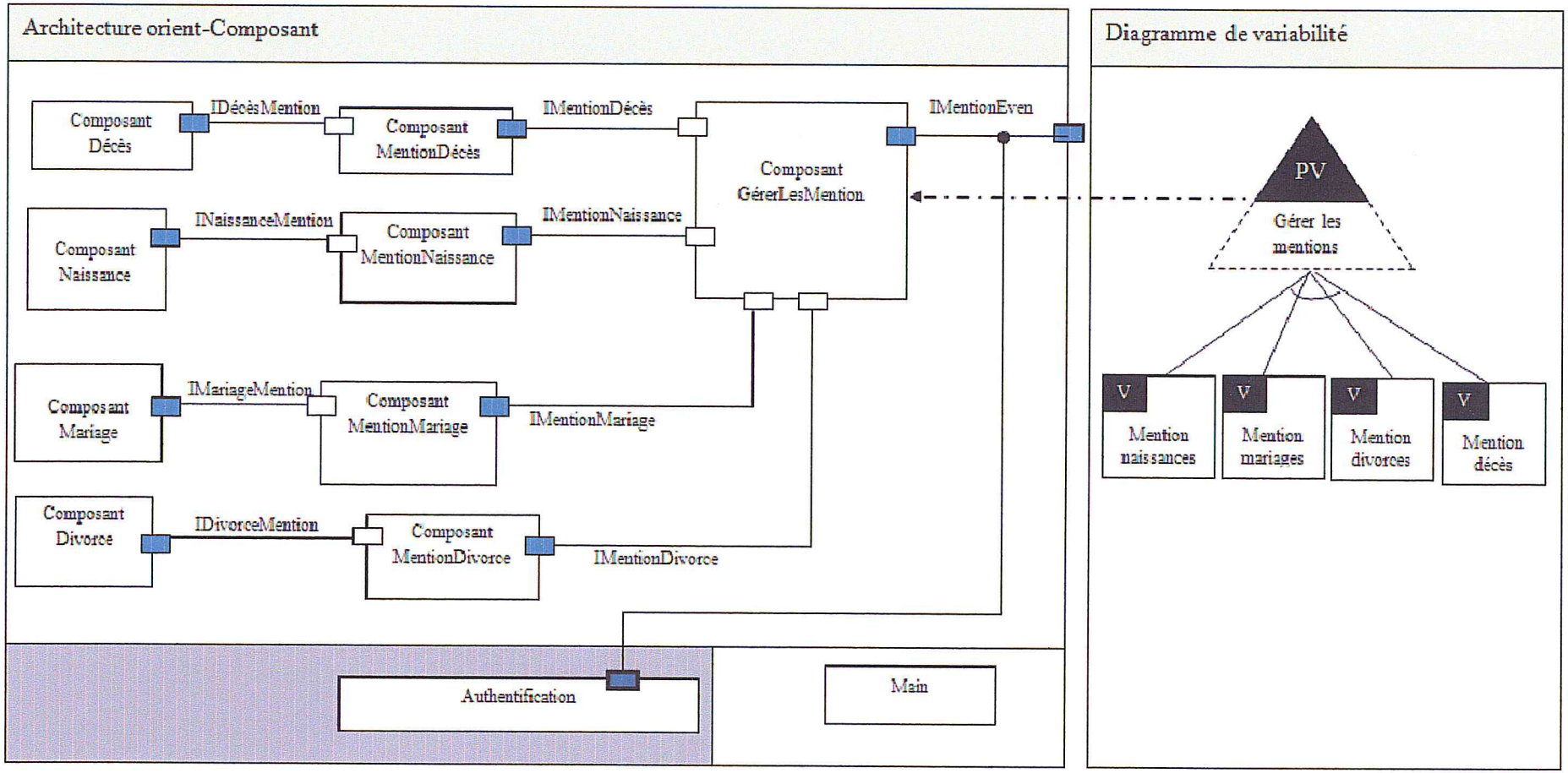


Figure 4.14 : Composant Gérer les mentions

e. Le composant actes transcrits sur les registres

Le composant actes transcrits sur les registres est un sous composant du composant demande des documents qui se charge de l'établissement des actes de naissance, de mariage, de divorce, de décès, et de non divorce. (Figure 4.15)

Il utilise ComoposantNaissance, ComposantMariage, ComposantDivorce et ComposantDécès pour récupèrer les informations des actes demandés.

f. Le composant actes non transcrits sur les registres

Le composant actes non transcrits sur les registres est un composant composite qui se charge de l'établissement des actes non transcrits sur les registres (fiche familial, la fiche individuelle, preuve d'individualité, certificat de résidence, fiche de résidence) (Figure 4.16)

Il utilise ComposantFiche_familial, ComposantLa_Fiche_ individuelle, ComposantPreuve_d_individualité, ComposantCertificat_de_résidence et ComposantFiche_de_résidence pour l'établissement des actes.

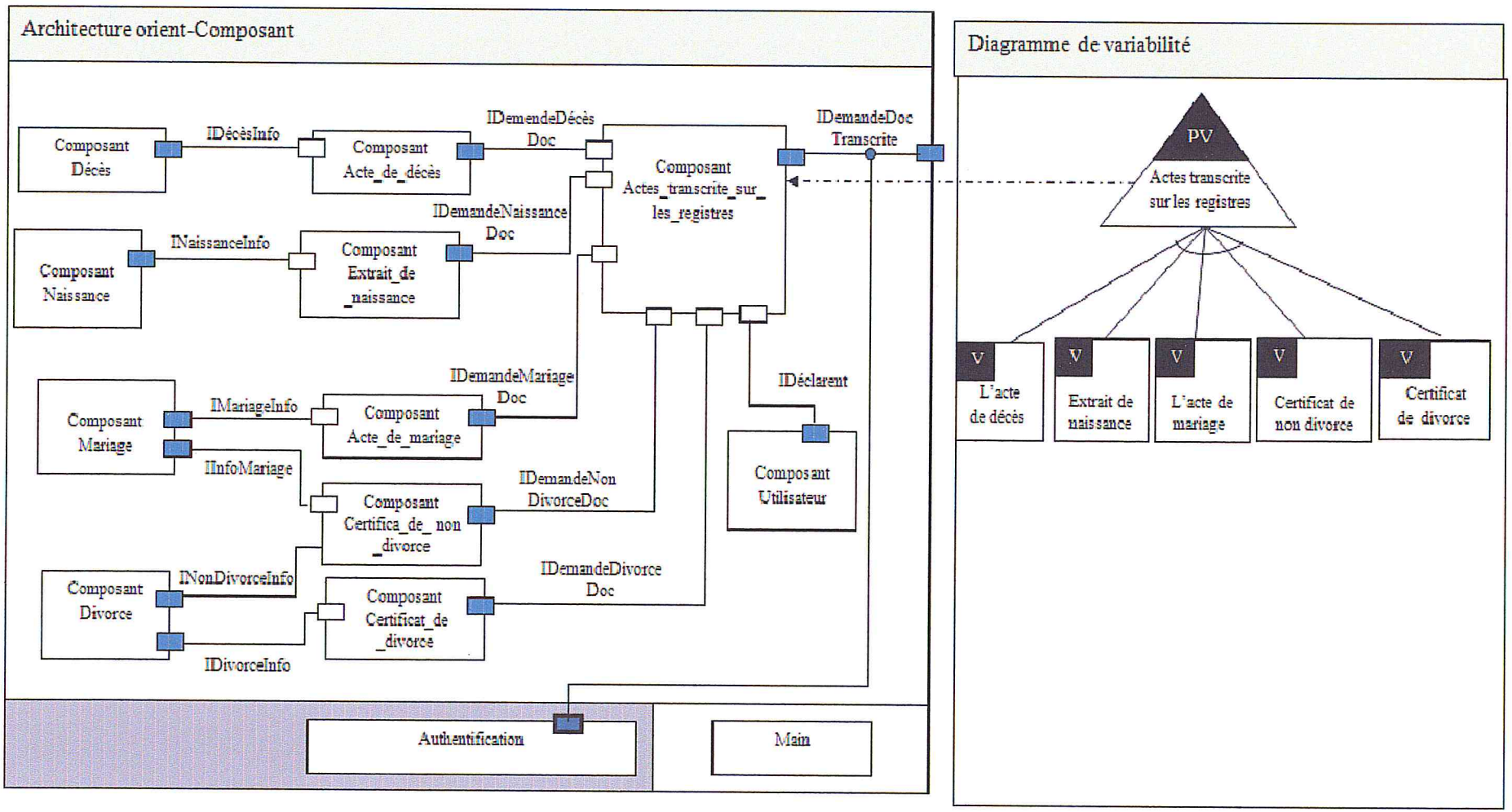


Figure 4.15 : Composant actes transcrite sur les registres

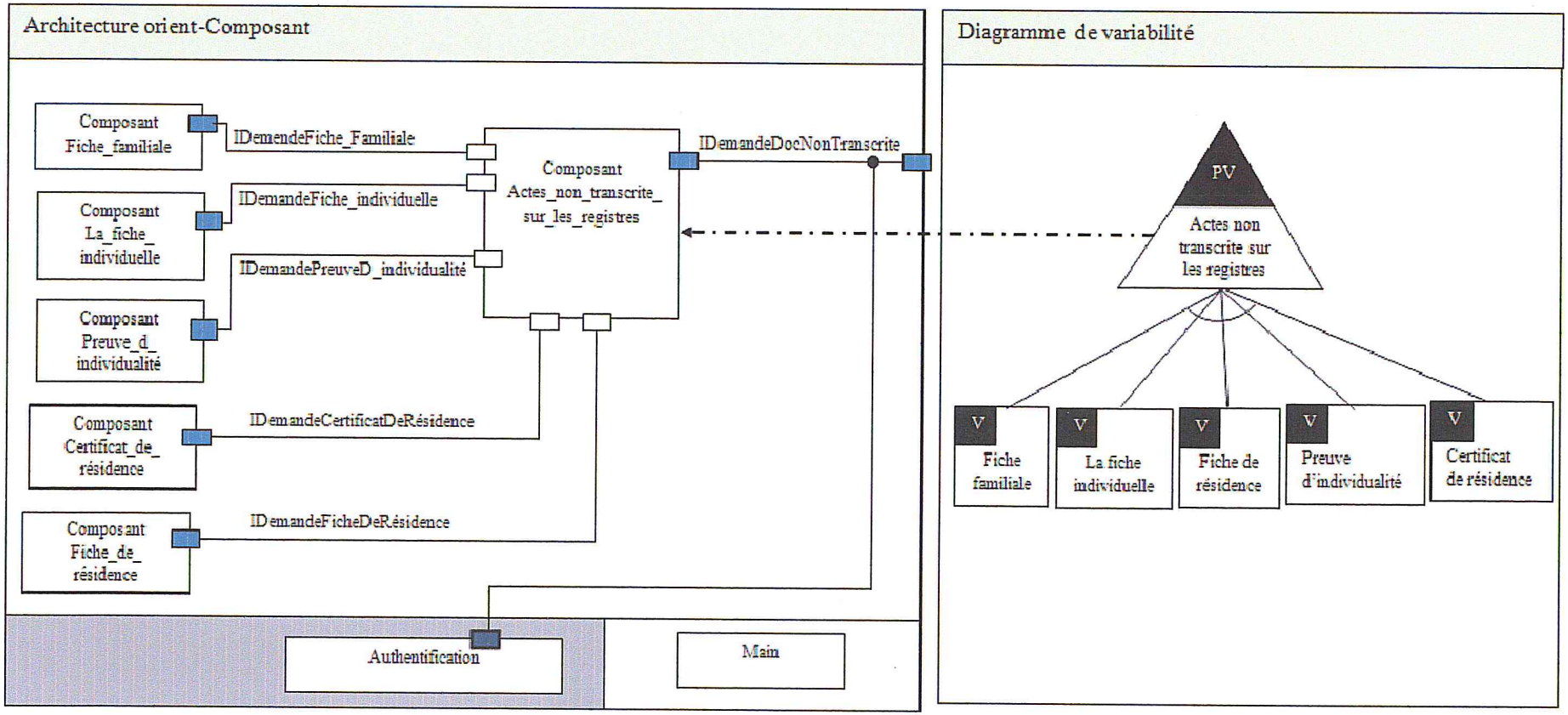


Figure 4.16 : Composant actes non transcrite sur les registres

g. Le composant Modules

Le composant modules se charge des fonctions de communication d'un citoyen avec son APC.

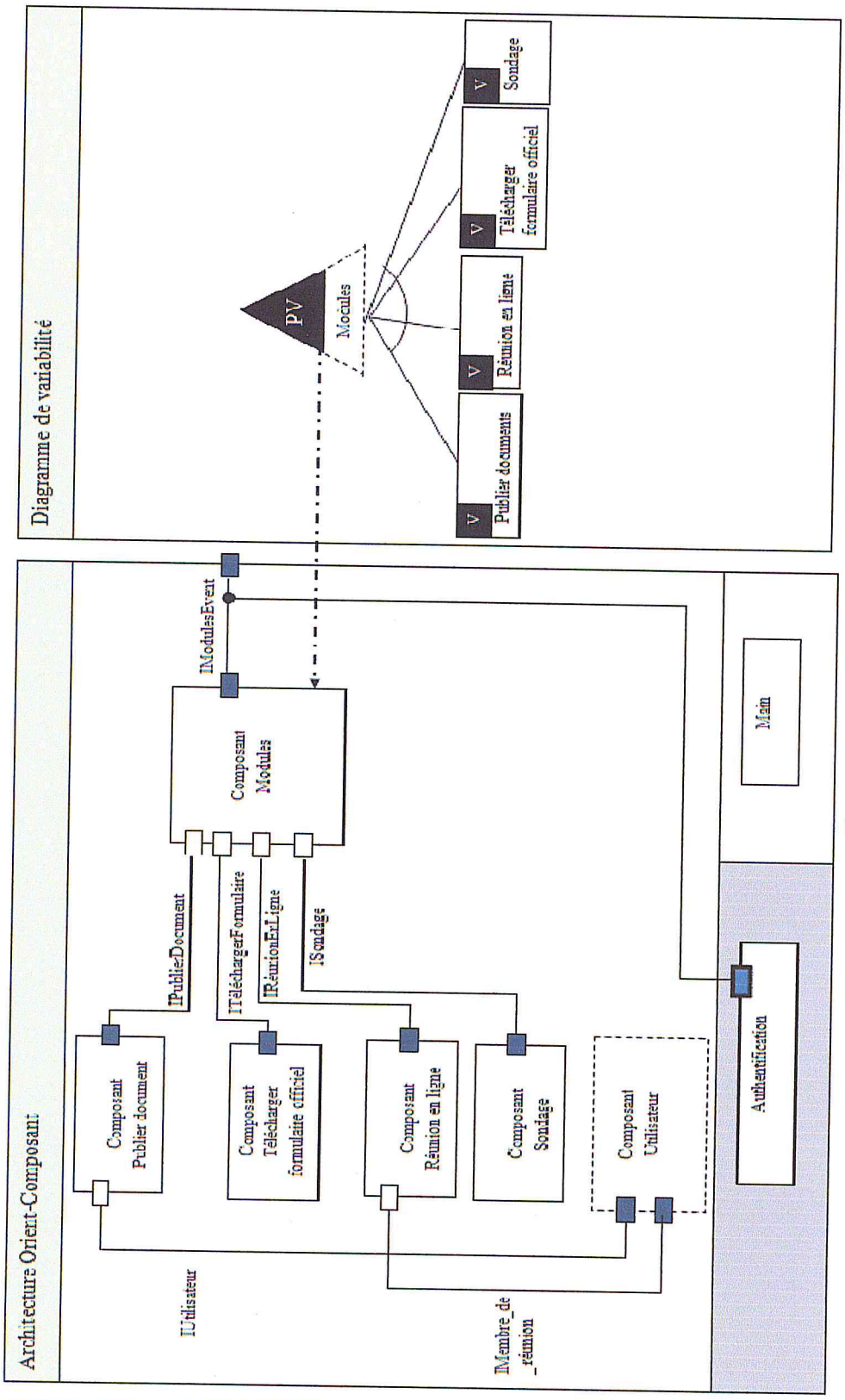


Figure 4.17 : Composant Modules

h. Le composant authentification

Le composant authentification se charge de :

1- L'authentification de l'utilisateur.
2- L'authentification du document produit
3- Vérifier l'authentification d'un document pour montrer les documents falsifiés. (Figure 4.18)

i. Le composant suivi et validation

Le composant suivi et validation se charge de :

1- La validation des déclarations des naissances, des mariages, des divorces et des décès.
2- La validation des mentions des naissances, des mariages, des divorces et des décès.
3- Le suivi des responsabilités des utilisateurs.
4- les fonctions de calibration d'états de sortie. (Figure 4.19)

j. Le Composant de communication

Ce composant s'occupe des fonctions de communication entre les APCs d'une même Wilaya ou autre et aussi de la communication avec les tribunaux et les hôpitaux grâce au ComposantCommunicationHorizontale.

Et s'occupe aussi de la communication d'une APC avec son daira, sa wilaya ou le ministère de l'intérieur. (Figure 4.20)

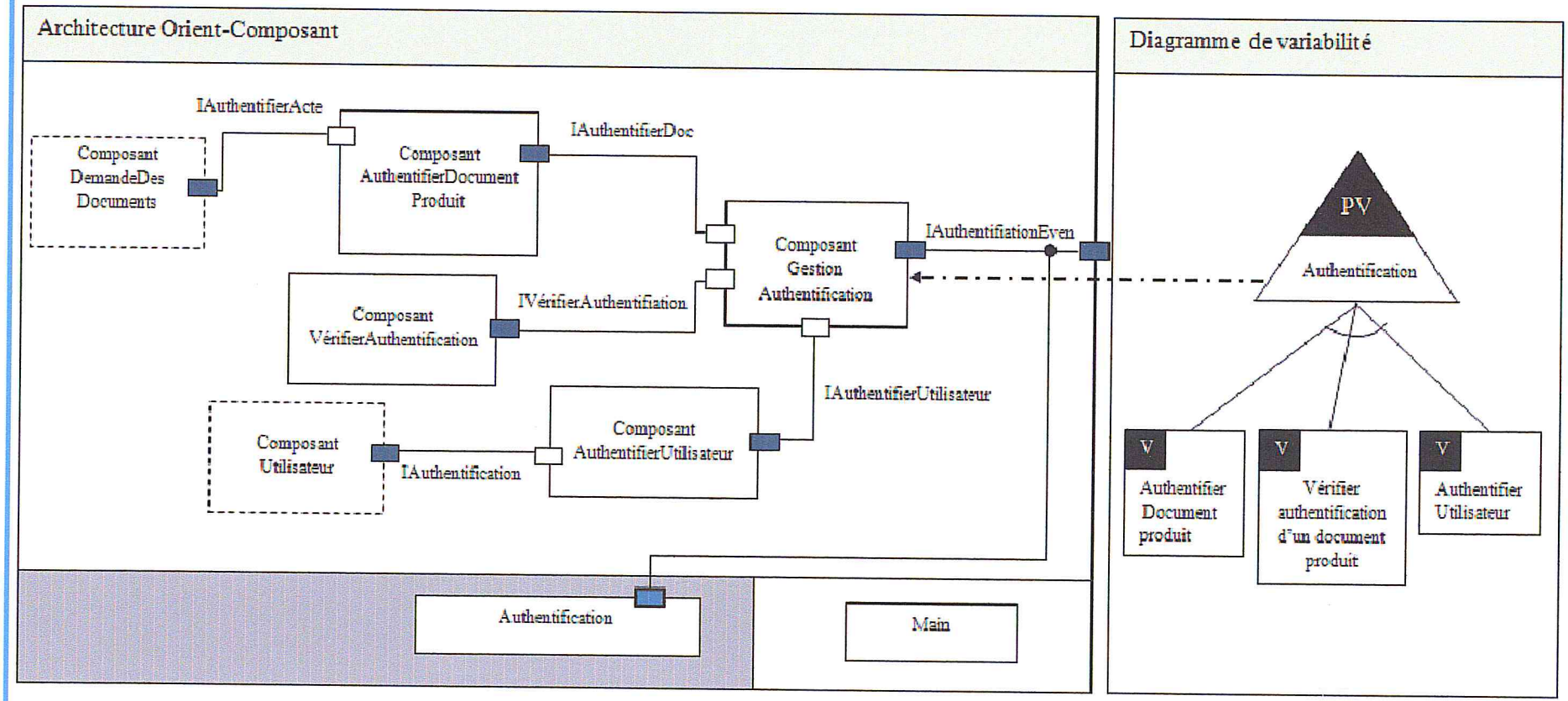


Figure 4.18 : Composant authentification

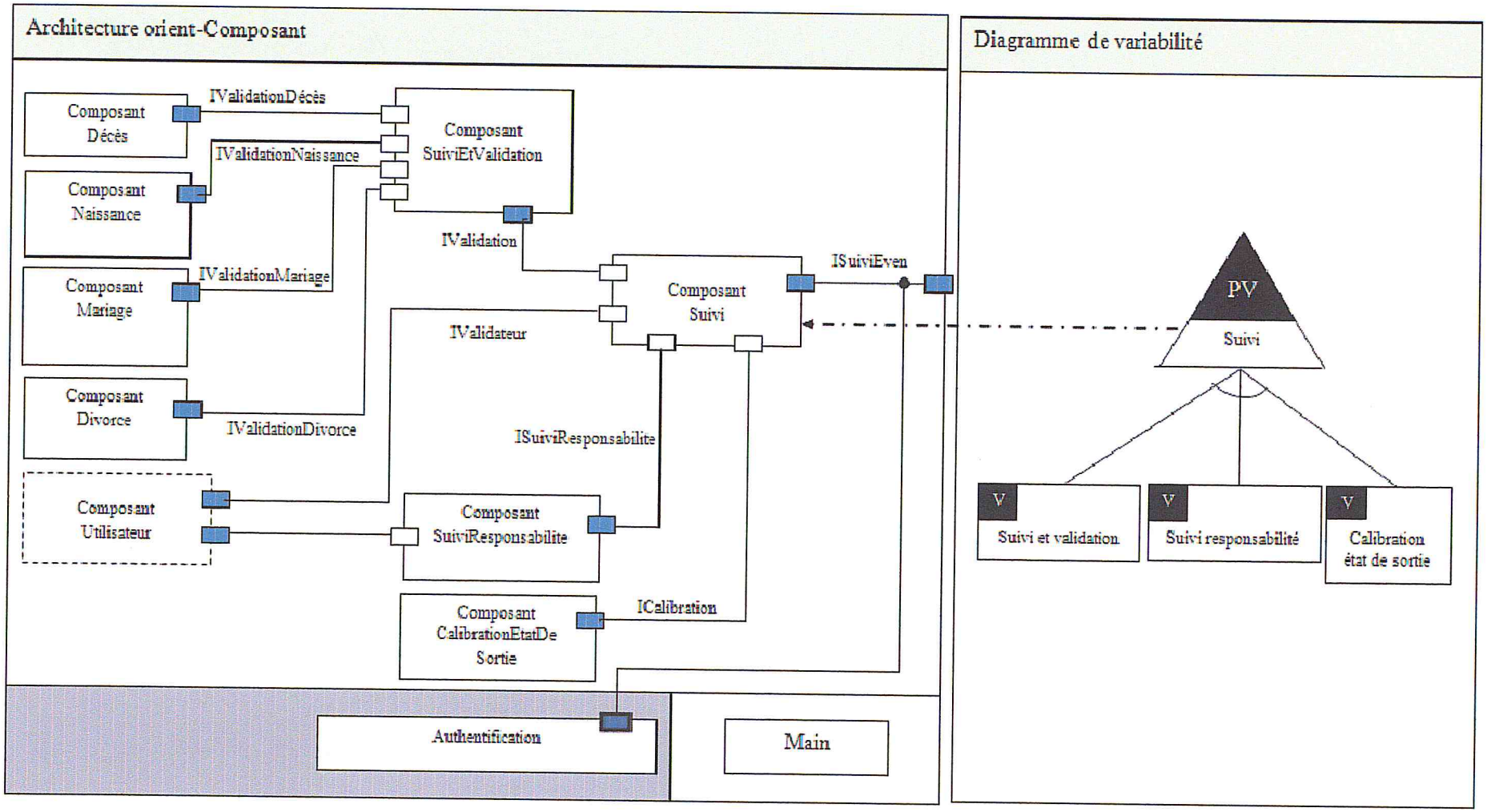


Figure 4.19 : Composant Suivi et validation

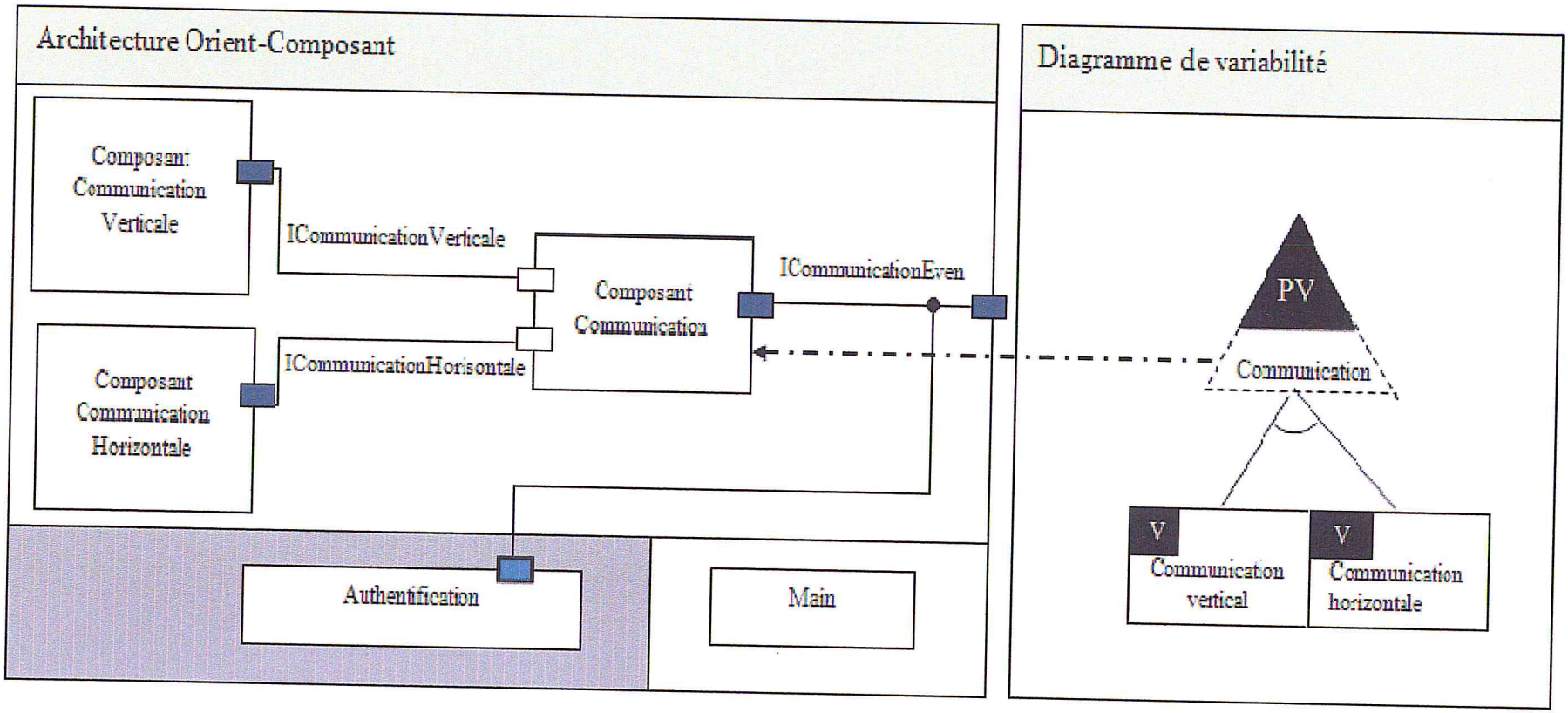


Figure 4.20 : Composant Communication

4.5. Réalisation du domaine

Après la modélisation de l'architecture on doit implémenter tous les composants de notre ligne de produit eAPC. Ainsi pour chaque nouvelle application on n'aura qu'a d'assembler les composants existants. En cas ou il y'a des besoins spécifique aux utilisateurs, on doit les prendre en compte en implémentant de nouveaux composants qui lui correspond, ces derniers doivent être gardé dans la bases des composant réutilisables pour être réutilisé par la suite en cas de nécessité.

Dans le cadre de notre projet de fin d'étude, nous ne pouvons pas réalisé tout les composants possibles, pour cela, nous avons choisi de réaliser les composants communs a tout les membre de la ligne de produit en plus de quelques variantes.

Les composants communs :

Les composants communs dans notre ligne de produit sont :

ComposantGestionUtilisateur qui s'occupe des fonctions suivantes :

1- Ajout, la modification et la suppression des comptes utilisateur.
2- La gestion des profils des utilisateurs.

ComposantConfigurationSysteme : ce composant est le noyau de chaque application générée à partir de notre ligne car c'est au niveau de ce composant qu'on précise la commune concernée et les gestionnaires de l'application.

ComposantGestionDesLieux s'occupe des fonctions de la gestion des lieux.

4.6. Conclusion

Dans ce chapitre nous avons présenté la conception de notre ligne de produit. Le résultat c'est l'architecture de référence qui contient de la variabilité, cette architecture nous permettra par la suite de dériver plusieurs types d'applications selon les besoin spécifiques et ce que nous allons présenter dans le chapitre suivant.

Chapitre 5

Ingénierie d'application

5.1. Introduction

Après la conception du domaine de notre ligne de produit et la réalisation des composants réutilisables, nous allons dérivés quelques applications pour montrer l'intérêt de l'utilisation de cette approche. Dans notre projet on se contente à deux exemples d'application, détaillées dans la suite de ce chapitre.

5.2. La preière application

5.2.1. L'analyse d'application

La première application dérivée à partir de notre ligne de produit s'occupe des déclarations des naissances avec des interfaces graphiques en français et authentification des utilisateurs à l'aide de mots de passe. Le system d'exploitation Windows et utilise une base de données relationnelle.

5.2.2. Modélisation d'application

Le feature modèle de cette application est le suivant :

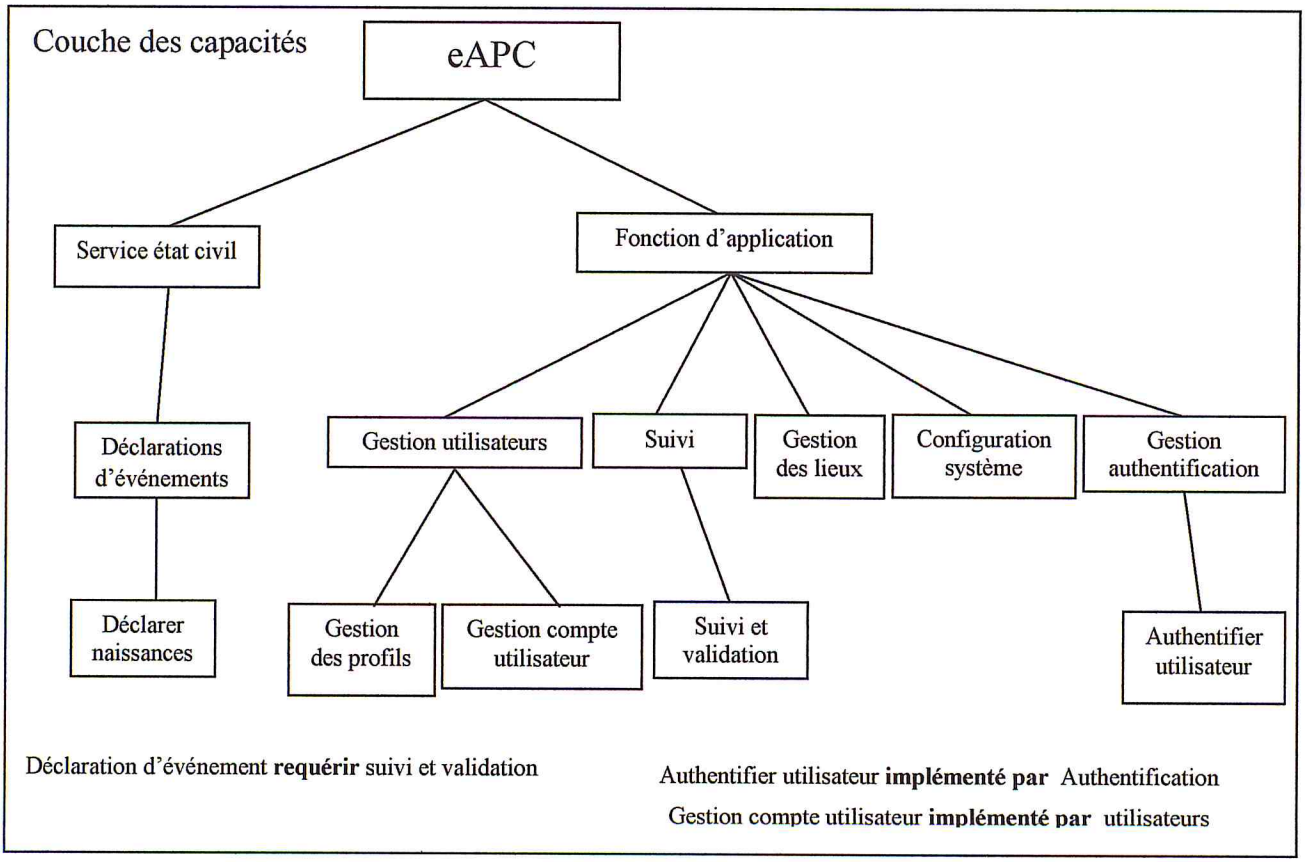


Figure 5.1 : Feature modèle pour la première application (1ère partie)

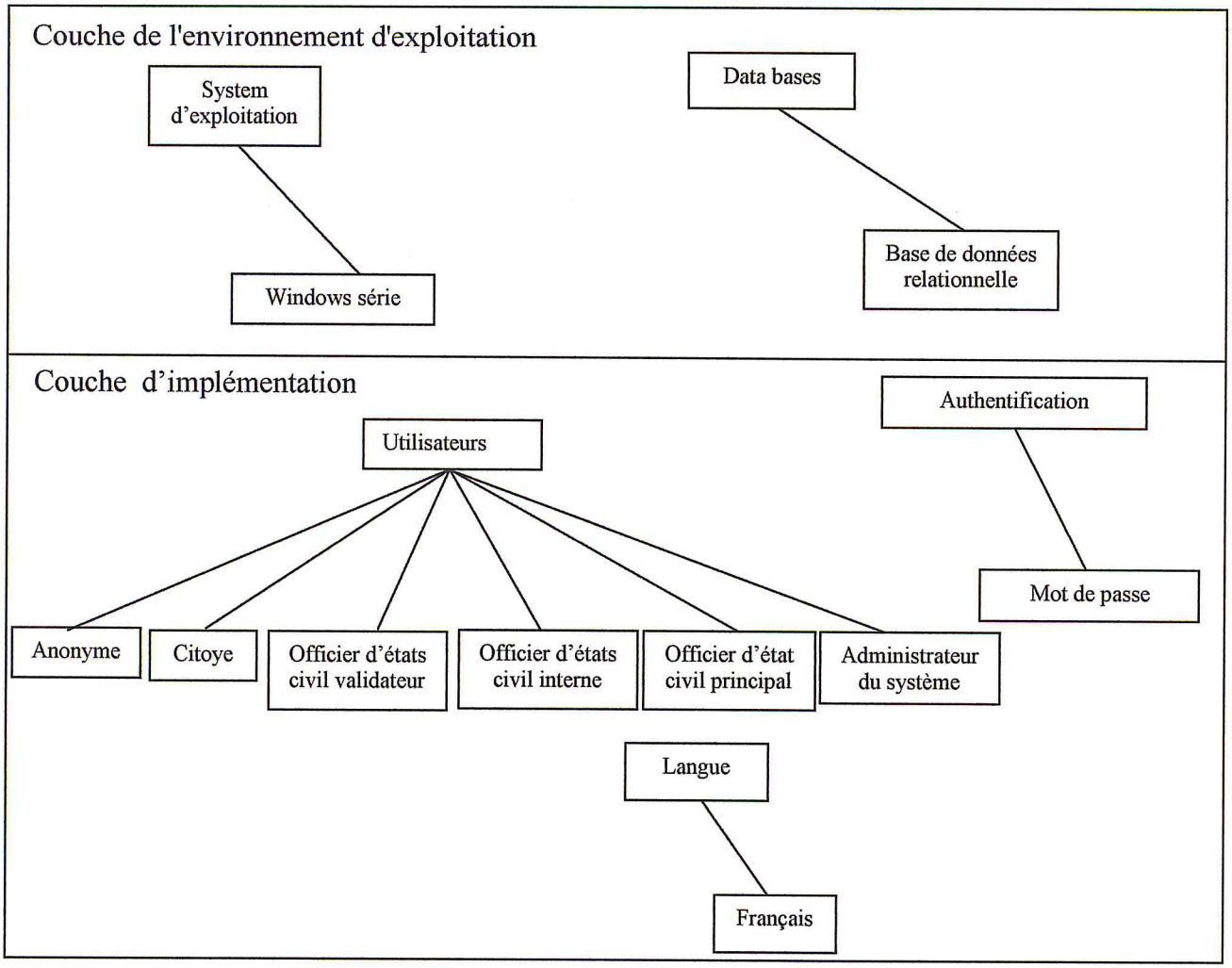


Figure 5.2 : Feature modèle pour la première application (2ème partie)

5.2.3. Modélisation de l'architecture d'application

Pour l'architecture nous avons utilisé l'architecture à base des composants avec la notation d'IASA déjà définie au chapitre précédent.

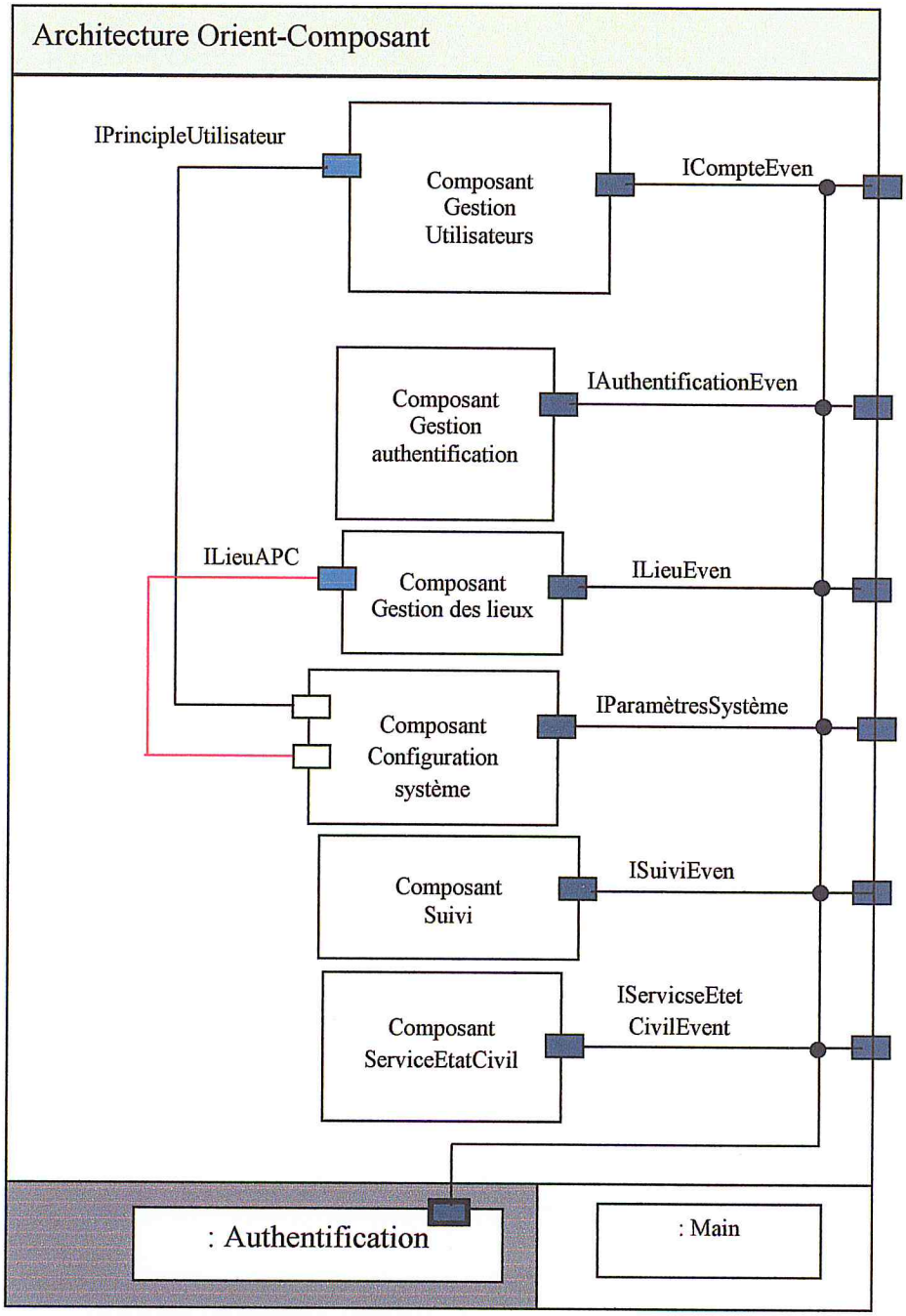


Figure 5.3 : l'architecture globale pour la première application

5.2.4. Raffinement des composants

a. Composant service état civil

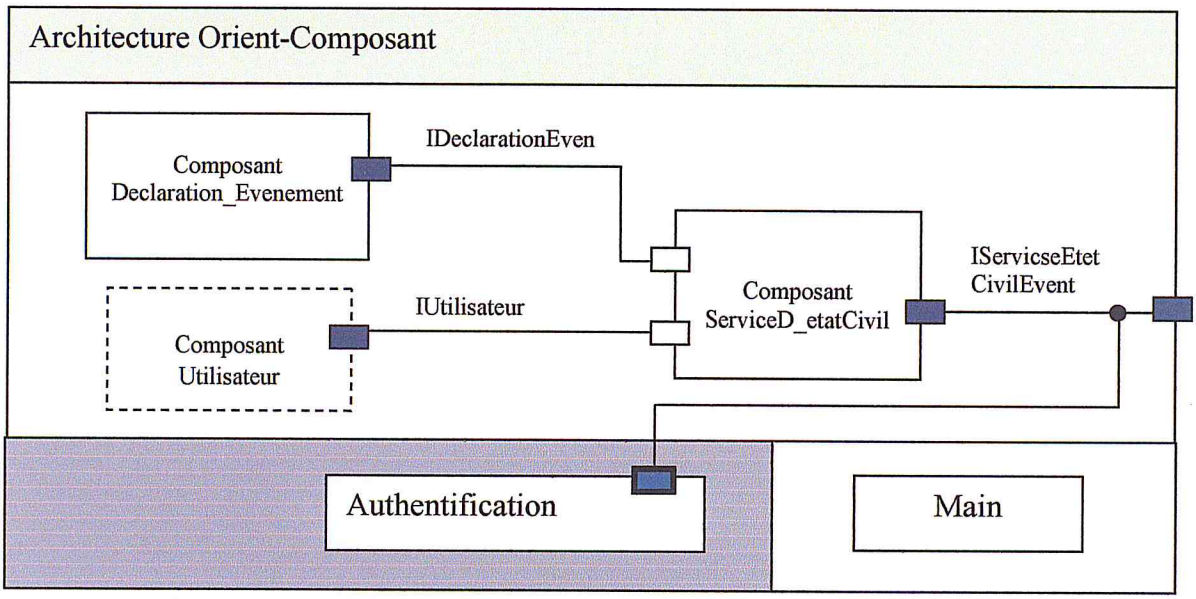


Figure 5.4 : Composant service d'état civil pour la première application

a. Composant déclaration d'événement

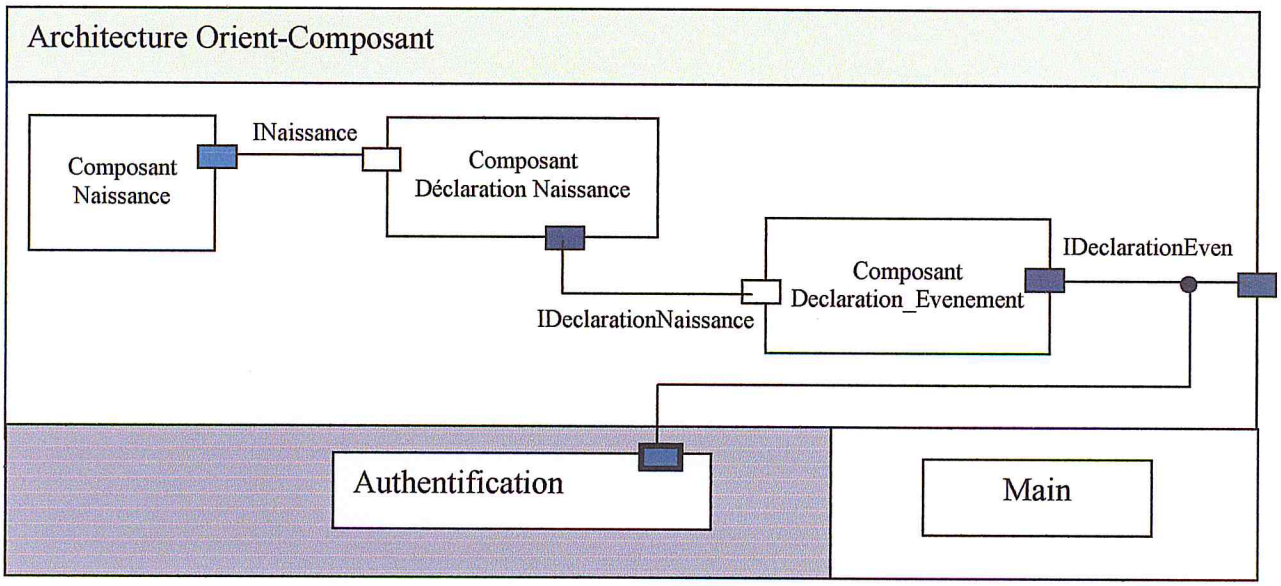


Figure 5.5 : Composant déclaration d'événement pour la première application

b. Composant suivi

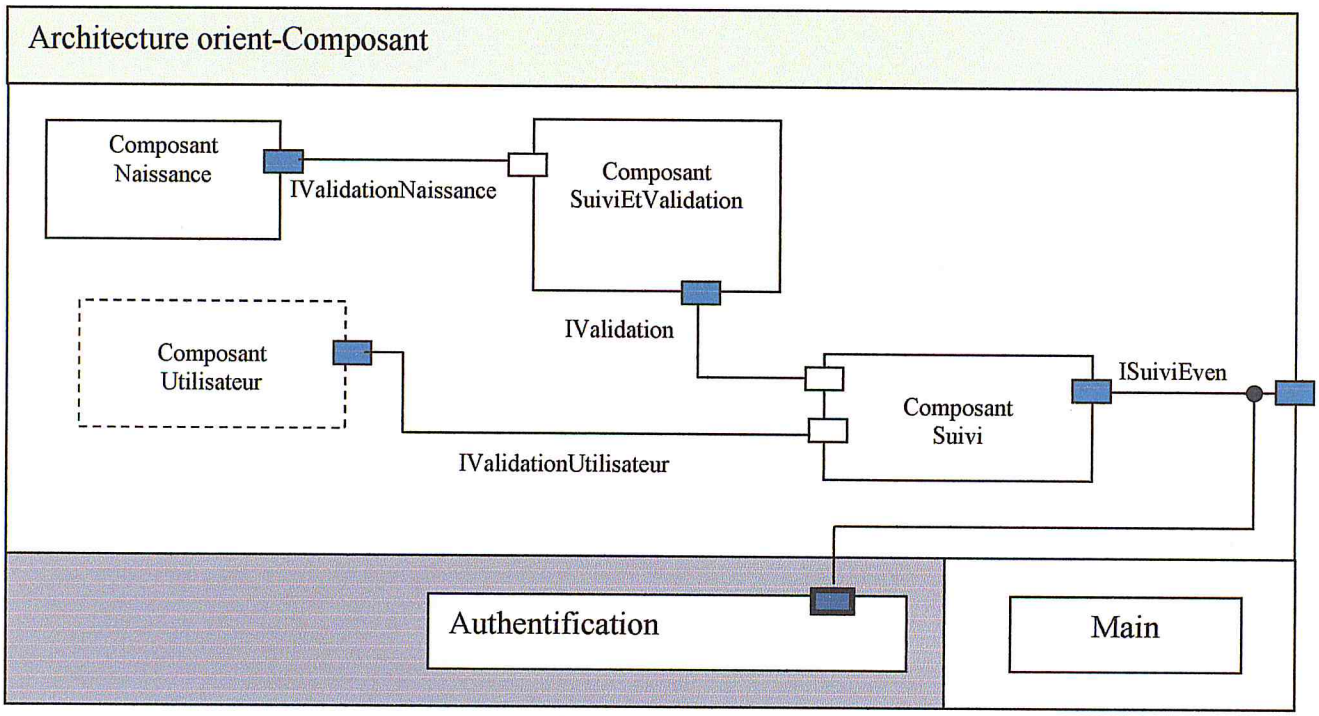


Figure 5.6 : Composant Suivi pour la première application

c. Composant gestion d'authentification

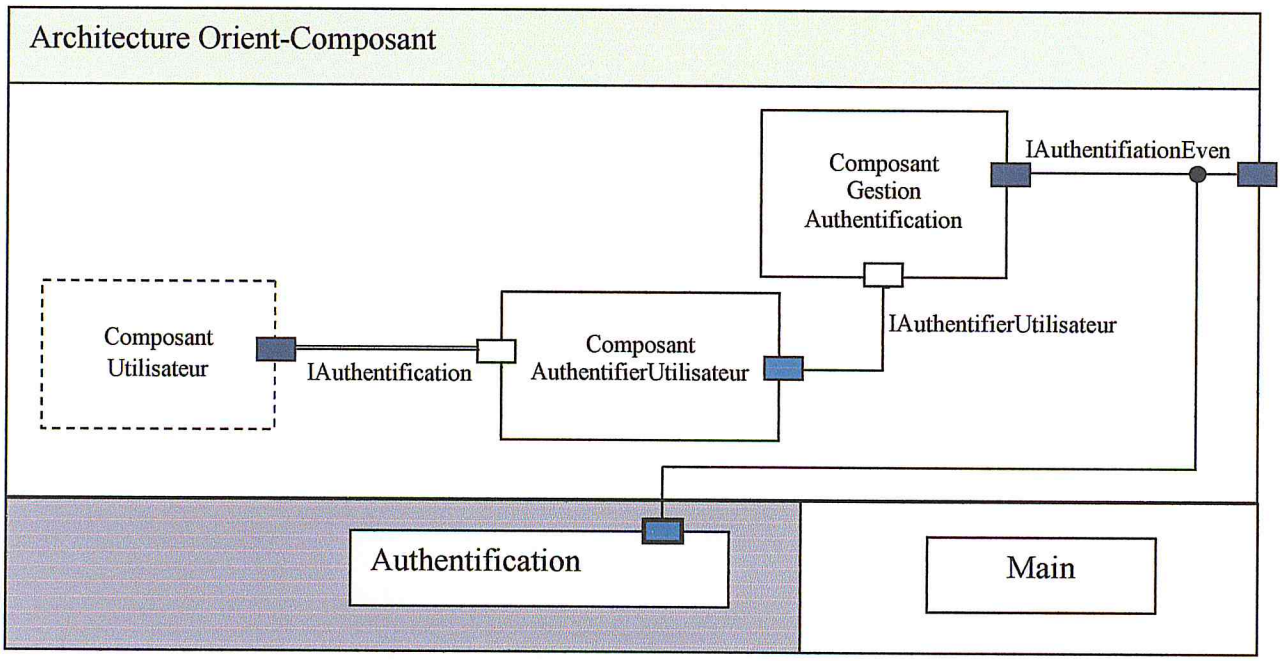


Figure 5.7 : Composant gestion d'authentification pour la première application

5.2.5. Réalisation d'application

Les applications dérivées à partir de notre ligne de produit sont des applications web qui suivent le modèle de conception MVC, pour cela nous avons utilisé Java EE «Java Enterprise Edition » qui permet la création d'applications web.

Java EE facilite le développement d'applications web déployées et exécutées sur un serveur d'applications.

- la couche Modèle est constituée d'objets Java se charge des traitements à effectuer sur les données et de leur stockage ;
- la couche Vue est constituée de pages JSP se charge de la présentation des données pour l'utilisateur et de l'interaction ;
- la couche Contrôle est constituée de servlets se charge d'orienter les requêtes entrantes vers les traitements et vues correspondants.

Pour le stockage des données on choisie MySQL. Car il est le SGBD le plus répandu.

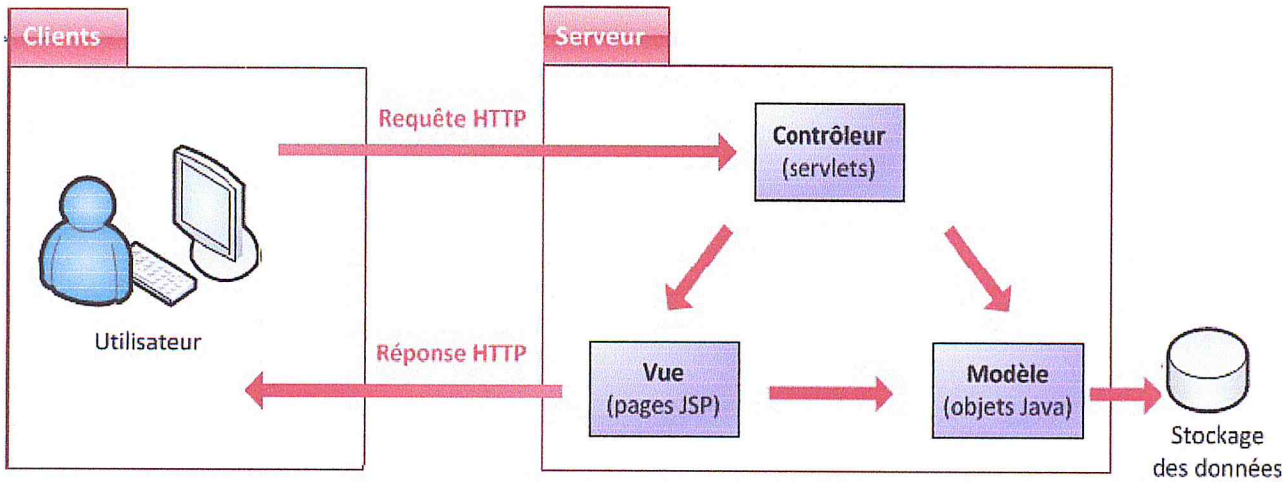


Figure 5.8 : Le modèle de conception MVC (Modèle Vue Contrôle)

Pour faire fonctionner une application web Java EE, nous avons besoin de un serveur d'applications. Nous avons choisie d'utiliser **Tomcat**, car c'est un serveur léger, libre, multiplateforme et complet.

5.2.6. Les interfaces de la premier application

Les figures suivantes présentent quelques interfaces de la première application :

Les étapes de configuration du system :

1- Configuration de l'APC concernée.
2- Configuration administrateur.
3- Configuration validateur.
4- Fin de configuration.
5- La page d'accueil

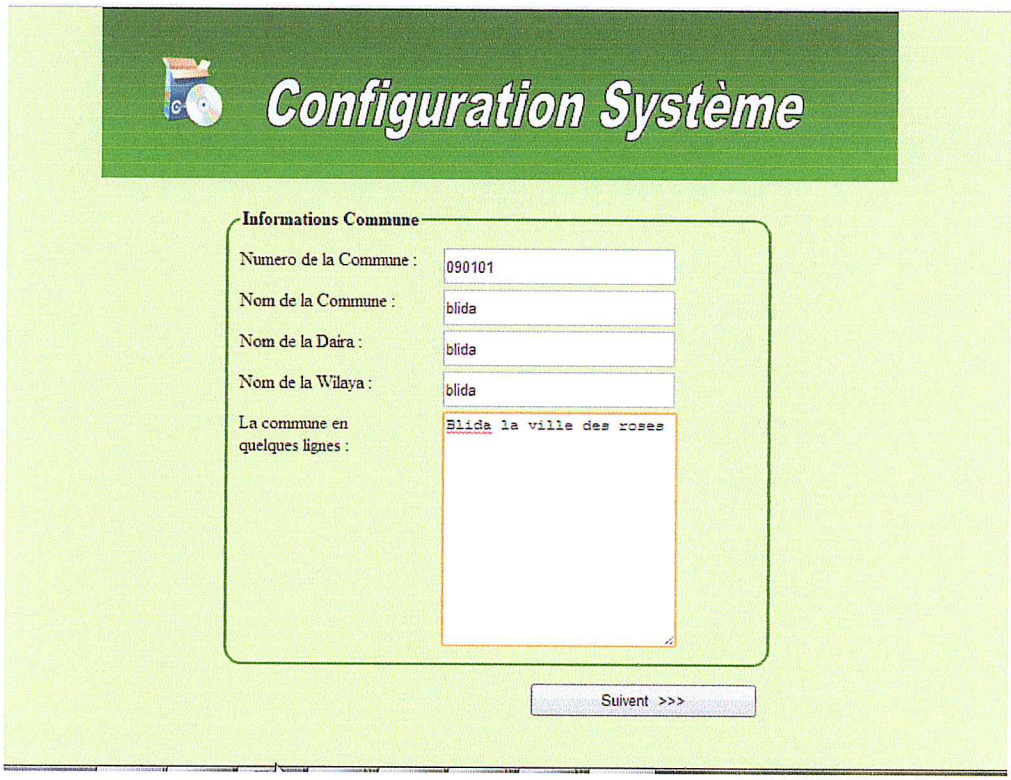


Figure 5.9 : 1ère étape de configuration du système (information essentiel de la commune concerné)

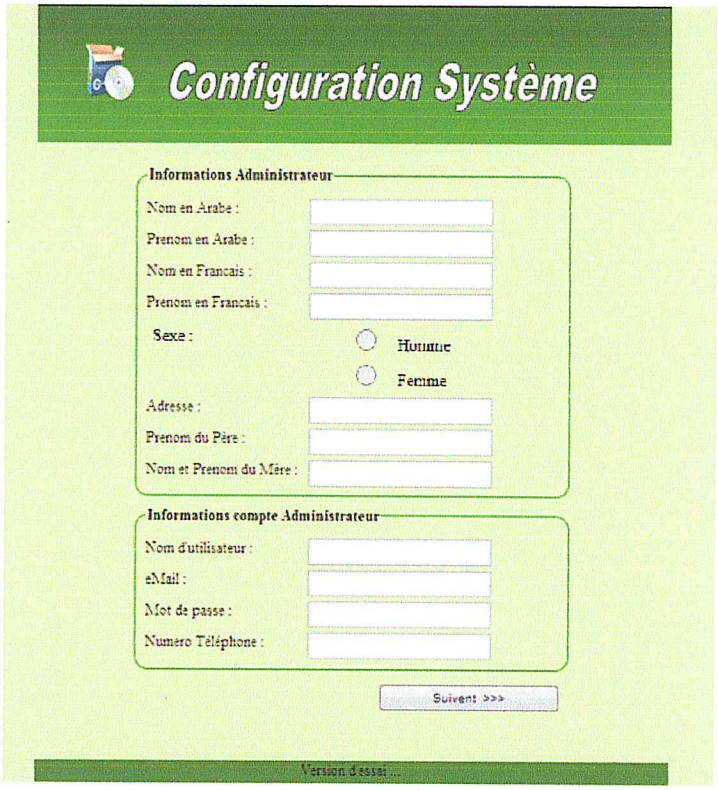


Figure 5.10 : 2^{eme} étape de configuration du système (configuration administrateur)

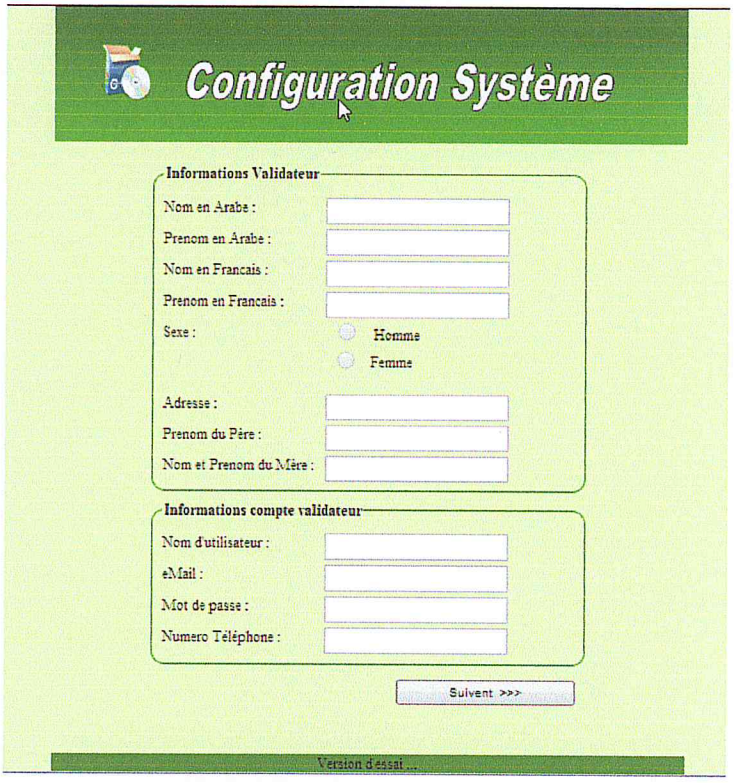


Figure 5.11 : 3^{eme} étape de configuration du système (configuration validateur)

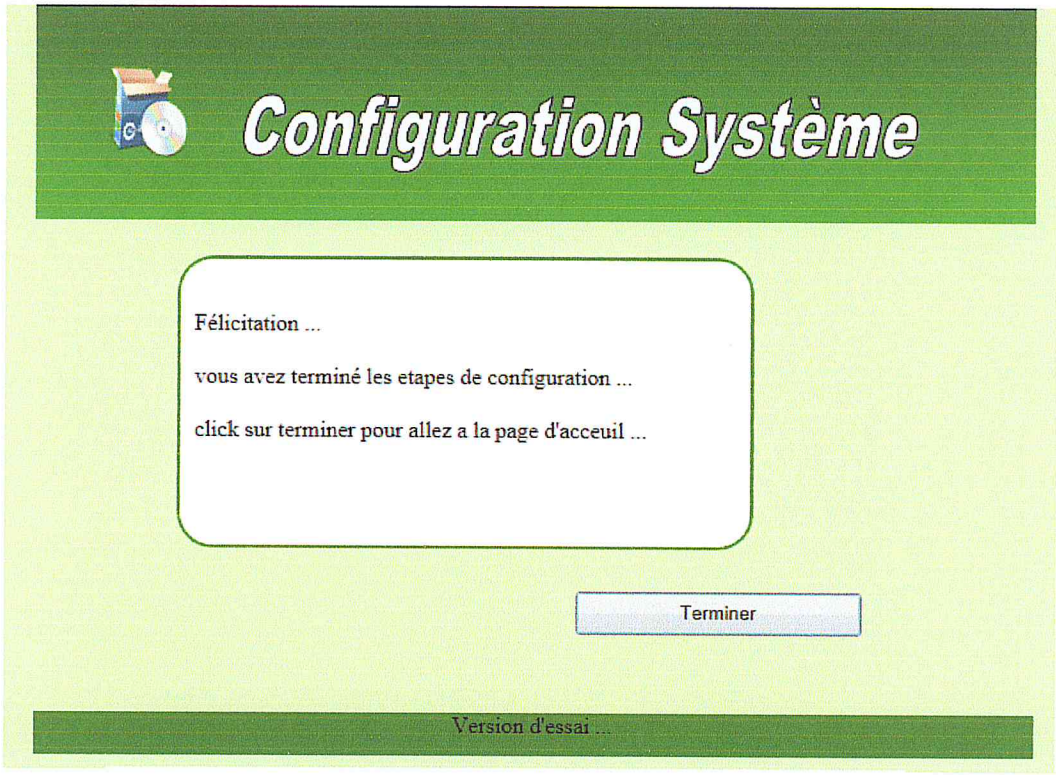


Figure 5.12 : Interface fin de configuration du système

Figure 5.13 : la page d'accueil de la première application

5.3. La deuxième application

5.3.1. L'analyse d'application

Pour le deuxième exemple nous avons dérivé une application qui fait les déclarations des mariages et les demandes des actes transcrite sur les registres, les interfaces graphiques de cette application sont on arabe et l'authentification des utilisateurs est par des mots de passe. Le system d'exploitation Windows et utilise une base de données relationnelle.

5.3.2. Modélisation d'application

La modélisation de cette application est la suivante :

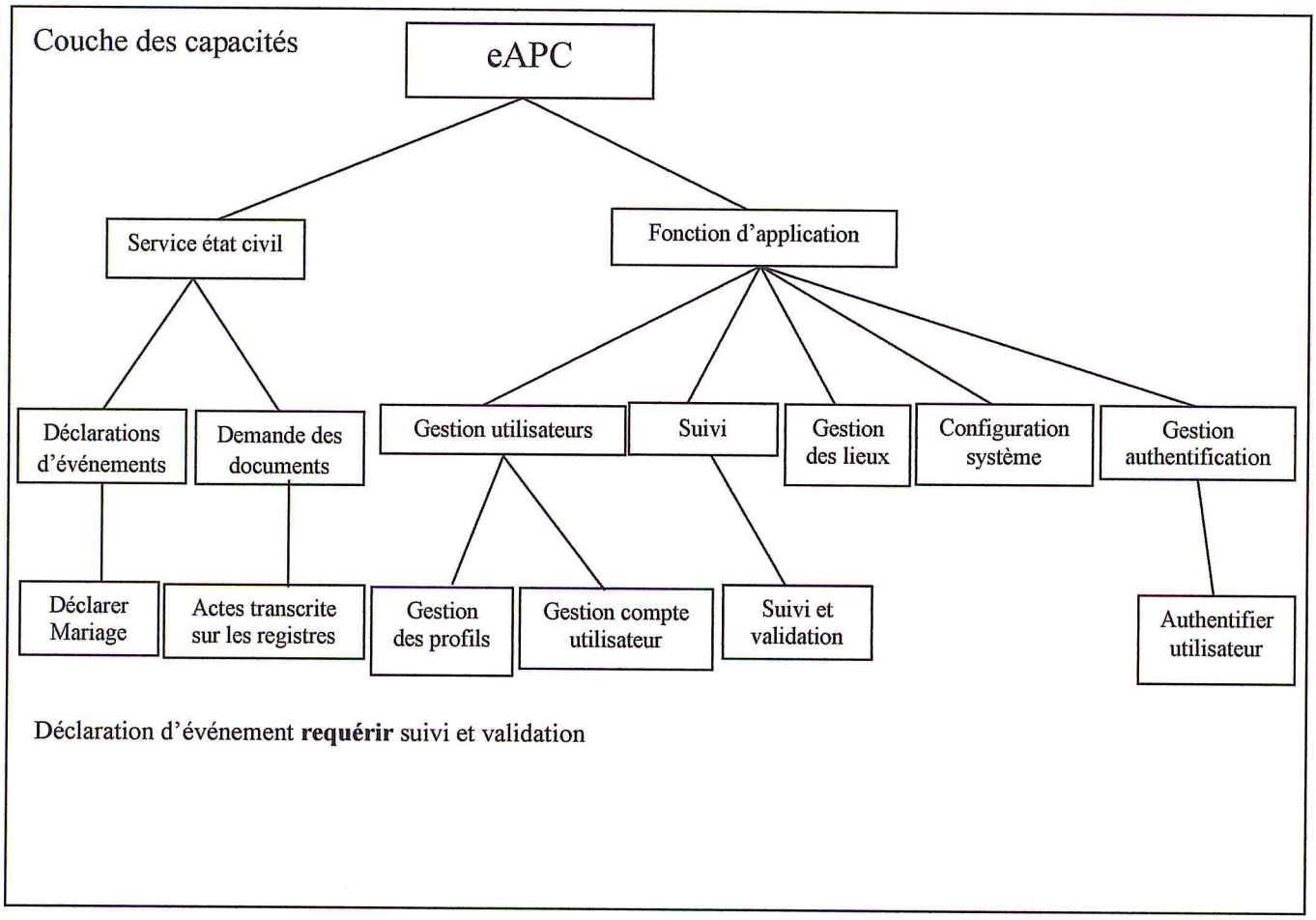


Figure 5.14: Feature modèle pour la deuxième application (1ère partie)

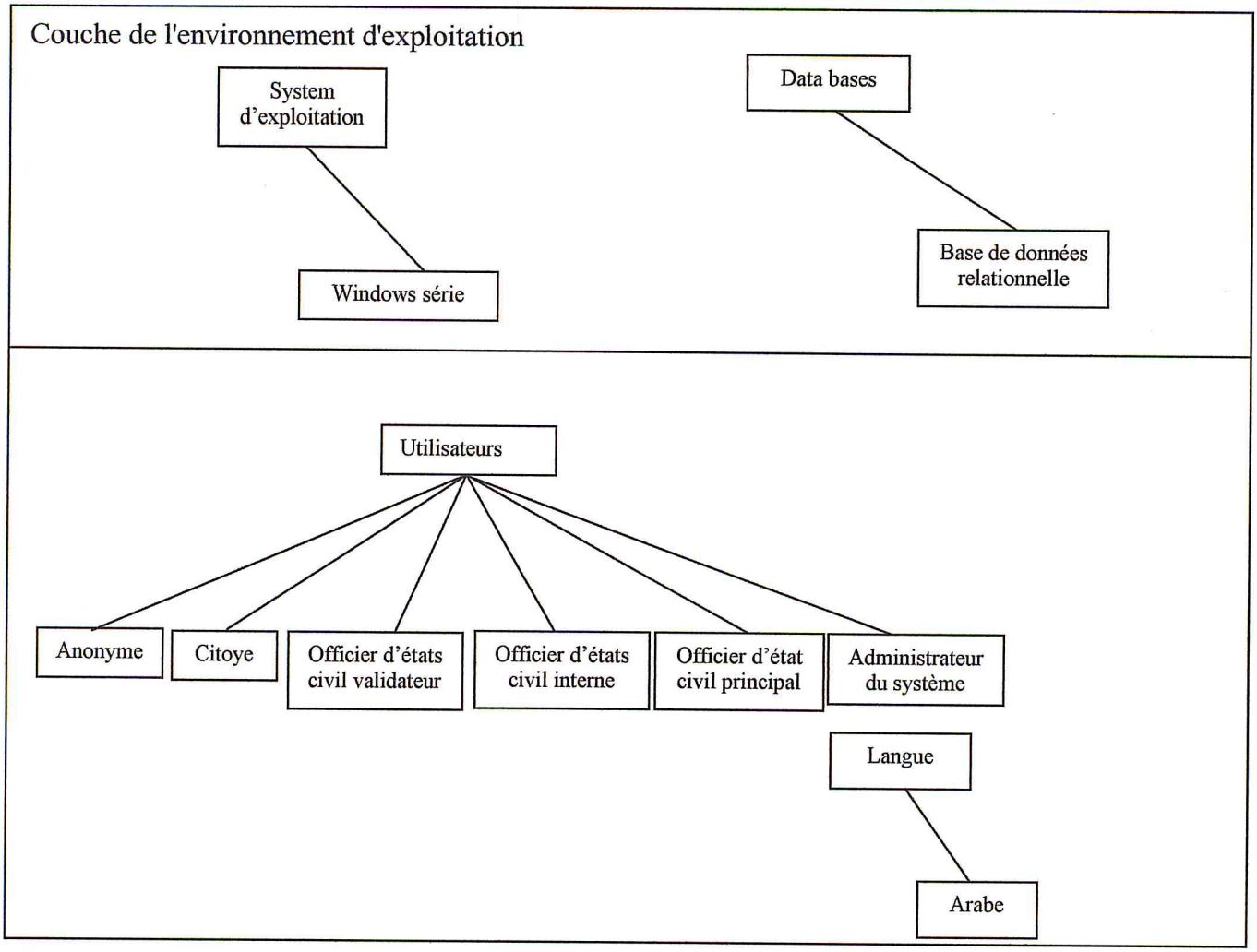


Figure 5.15: Feature modèle pour la deuxième application (2eme partie)

5.3.3. Architecture d'application

C'est la même architecture de la première application avec deux changements au niveau des raffinements des composants service d'état civil et déclaration d'événement.

5.3.4. Raffinement des composants

a. Composant service état civil

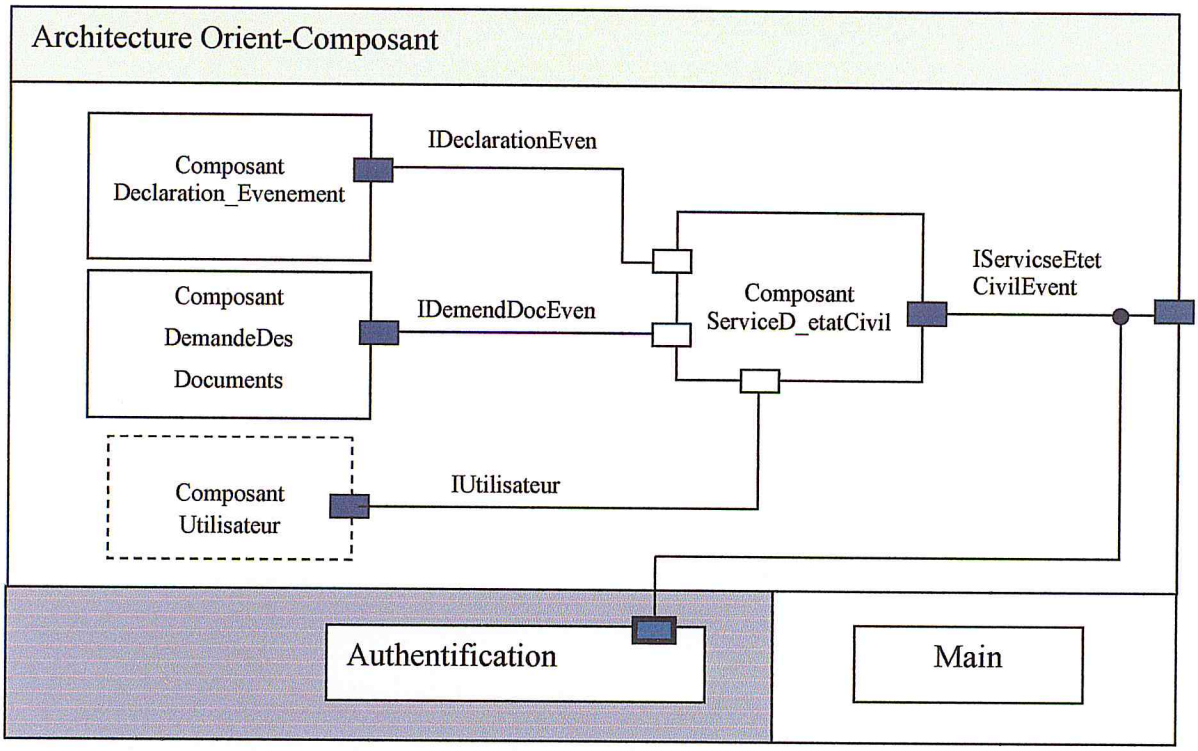


Figure 5.16 : Composant service d'état civil pour la deuxième application

b. Composant déclaration d'évènement

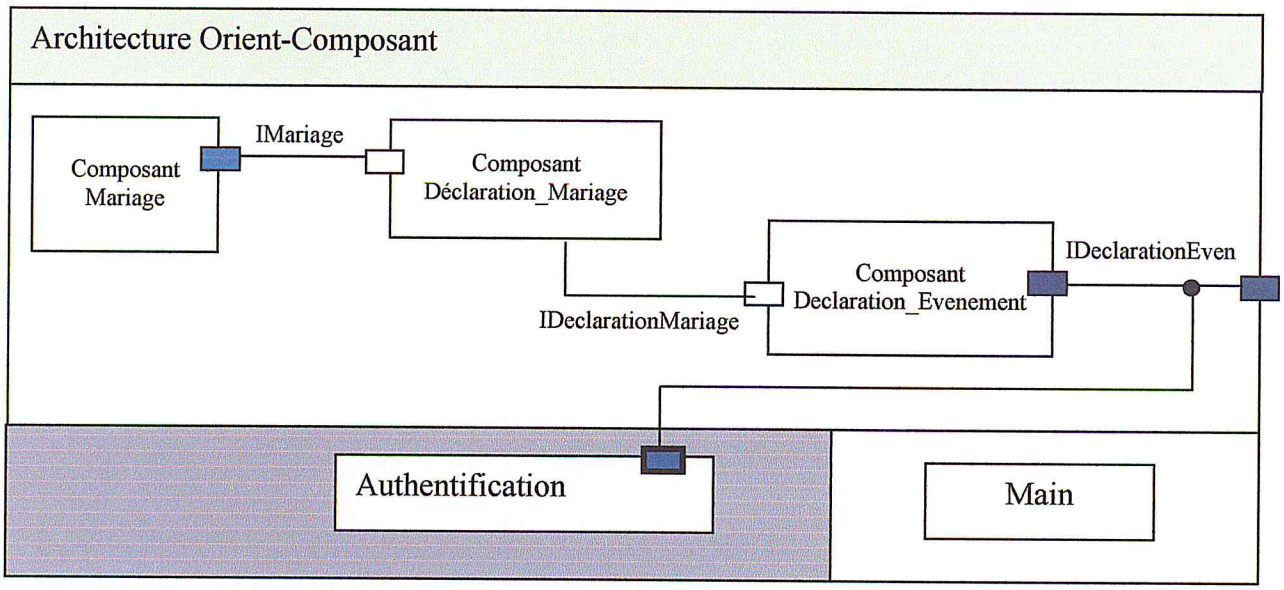


Figure 5.17 : Composant déclaration d'événement pour la deuxième application

c. Composant demande des documents

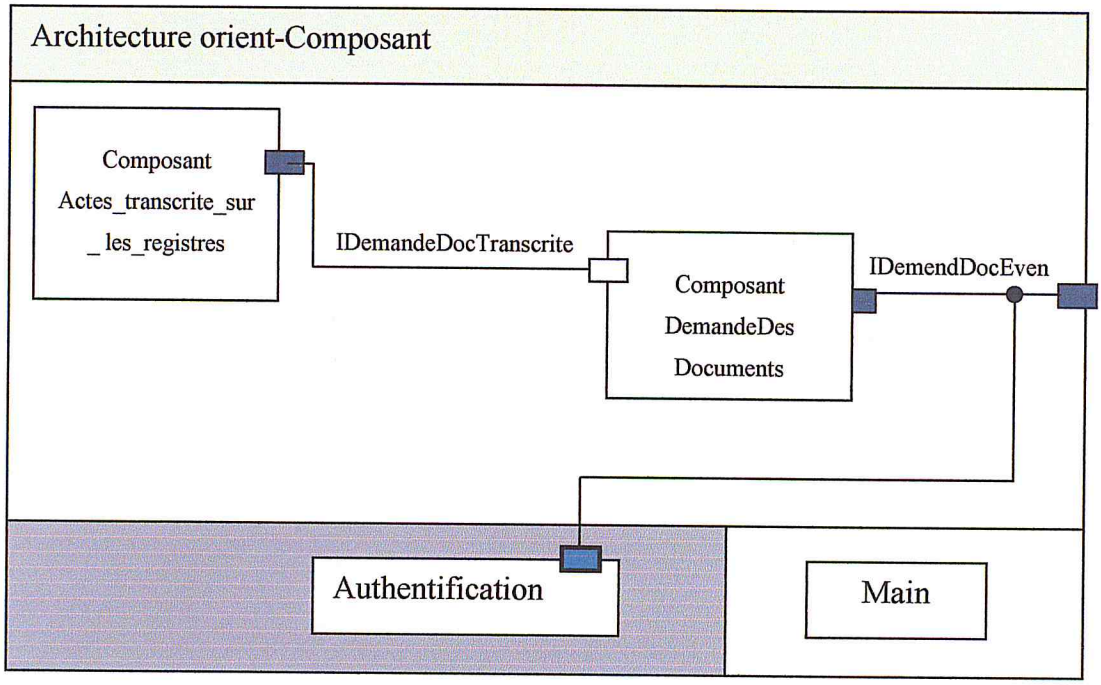


Figure 5.18 : Composant Demande Des Documents pour la deuxième application

5.3.5. Réalisation d'application

Pour la réalisation de la deuxième application nous avons utilisé les mêmes outils de développement et le même modèle de conception MVC.

5.3.6. Les interfaces de la deuxième application

Les figures suivantes présentent quelque interfaces de la deuxième application :

1- Page configuration de la commune concerne.
2- Page d'accueil.
3- Page demande de documents

Figure 5.19 : Configuration deuxième application (1ère etape)

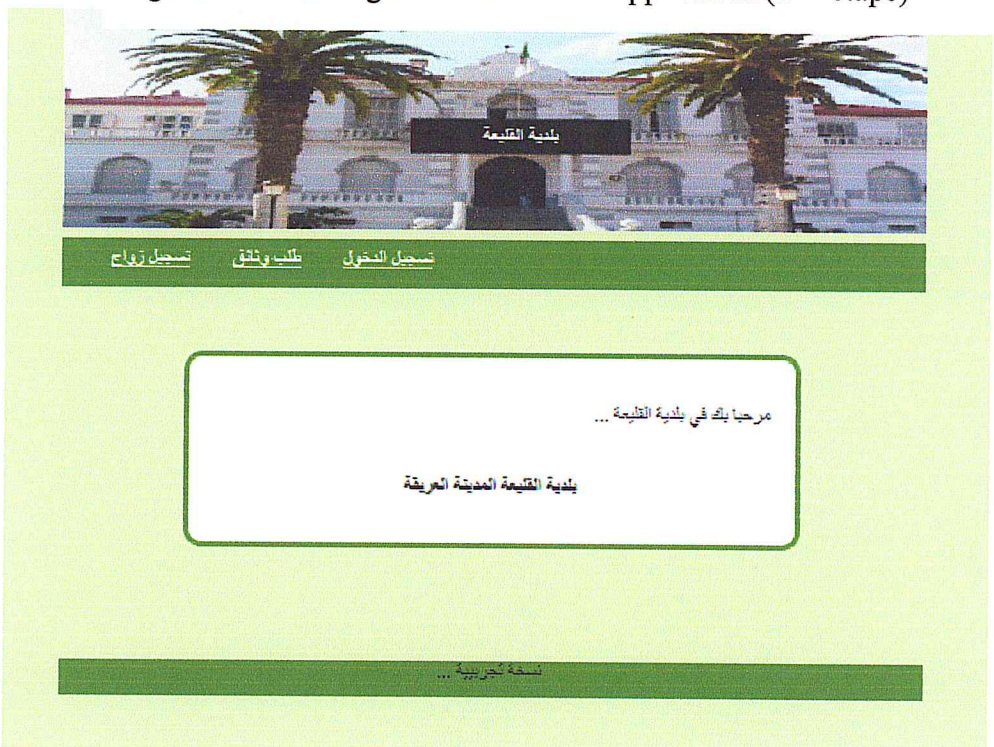


Figure 5.20 : page d'accueil deuxième application

Figure 5.21: Interface demande documents deuxième application

5.4. Conclusion

Dans ce chapitre nous avons vu la dérivation de deux applications à partir de notre ligne de produit. La première application est un exemple pour la commune de Blida avec la fonction de déclaration de naissance, et la deuxième concerne l'APC de kolea avec les fonctions déclaration de mariage et demande des documents.

Conclusion générale

Le travail présenté dans ce mémoire est le résultat d'environ une année de recherche et de travail. Pendant ce temps, nous avons faite des études bibliographiques des deux approches ligne de produit logiciel et architecture logiciel dans le but de métriser les concepts de base.

Notons que le domaine des lignes de projet est un domaine de recherche récent ce qui nous a présenté un premier défi. Les travaux dans ce domaine sont rares et même lorsqu'ils existent on les trouve dans des articles de conférences qui ne dépassent pas quelques pages i.e., ils ne contiennent pas suffisamment de détails pour nous aidez. Le principe des lignes de produit est lui-même un autre défi au lieu de penser à une application du domaine d'étude il fallait penser à tout ce qu'on peu avoir comme membre de notre ligne de produit logiciel.

Afin de renforcer la réutilisation de nos éléments logiciels nous avons choisi d'utiliser une approche orientée composant. Une étude bibliographique a été faite aussi pour cette approche, où nous avons choisi IASA pour la modélisation de notre architecture de référence.

Après l'étude bibliographique nous avons entamé le processus de développement de notre ligne de produit pour le domaine des APC. Ce processus est divisé en deux processus : ingénierie de domaine et ingénierie d'application. Le deuxième processus a été appliqué pour des applications simplifié pour montre le bon fonctionnent des résultats du premier processus.

Ce travail nous a permet d'une part, d'entamer un nouveau domaine de recherche qu'est les lignes de produit logiciel. Et d'autre part, d'améliorer nos connaissances en matière d'architecture logicielle et de développement d'applications web avec Java. Il nous a permis aussi d'acquérir une certaine expérience dans le domaine de l'orienté composant. Nous espérons que notre travail aura été à la hauteur de vos espérances.

Les Perspectives :

Les perspectives envisagées sont :

- Test des composants réutilisables on dérivants d'autres membres de la LdP.
- Implémentation des variantes qui peuvent être utilisés par d'autres membres de notre LdP.
- Le développement d'une méthode de dérivation des membres de la LdP qui pourrai être automatique ou semi automatique.

Références bibliographiques

[1] Jean-Christophe, « Les lignes de produits logiciels Réutilisation et variabilité », Publication périodique de Smals Juin 2009.

[2] Kyo C. Kang, Sajoong Kim, « FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures ».

[3] Felix Bachmann and Paul C. Clements: Variability in Software Product Lines, technical report CMU/SEI, 2005.

[4] Feature Model to Orthogonal Variability Model Transformation towards Interoperability between Tools. David Benavides and Antonio Ruiz University of Seville, ETSI Informatica Spine.

[5] R. Roshandel and N. Medvidovic, Multi-View Software Component Modeling for Dependability, in *Architecting Dependable Systems II, Lecture Notes in Computer Science*, vol. 3069, pp. 286-304, 2004 (ISSN 0302-9743).

[6] N. R. Mehta, N. Medvidovic and S. Phadke, Towards a Taxonomy of Software Connectors. *ICSE '00*, Limerick, Ireland, May 2000.

[7] Olivier Barais, « Construire et Maitriser l'Evolution d'une Architecture Logicielle µa base de Composants », Thèse de doctorat, l'université des Sciences et Technologies de Lille, 2005.

[8] Djamal BENNOUAR et abderrezak HENNI, « A Review of an Aspect Oriented Architecture Description Language » The Meditaranen Journal of Computers and Networks,Vol 6,N° 1, pp15-22, 2010.

[9] D. Garlan.: Formal Modeling and Analysis of Software Architecture: Components, Connectors, and Events. Proceedings of the 3rd Internation School on Formal Methods for the Design of Computer, Communication and Software Systems: Software Architectures, SFM 2003. 2804 (2003) 1–24].

[10] Nenad Medvidovic, Architecture-Based Specification-Time So ftware Evolution, Phd Thesis, UNIVERSITY OF CALIFORNIA, IRVINE, 1999.

[11] W. Seddaoui : Conception et réalisation selon une approche MVC mettant en œuvre le Framework Struts2, d'un composant représentant l'état civil d'une APC, PFE d'Ingénieur d'état, 2009, Université Saad Dahleb de Blida.

[12] Abdelhakim BAOUYA et Mohamed MAHDJOUB : Conception Orientée Aspect et par composants d'une application d'E-Gouvernement, PFE Master 2010/2011, Université Saad Dahleb de Blida.

[13] http://www.interieur.gov.dz/Services Publics et procédures/