Projet en Génie Logiciel

MESURE DE LA TAILLE FONCTIONNELLE EN COSMIC-FFP DES LOGICIELS DEVELOPPES SELON LE PROCESSUS RUP

Présenté Par :

Mongi SIOUD

Encadreurs :

M. Alain Abran

M. Normad Séguin

MGL9701 – Projet en génie logiciel

UQAM - Hiver 2004

Tables de Matières

1. INTRODUCTION 4
1.1 Objectif du document 4
1.2 Vue d'ensemble 4
1.3 Survol du document 5
1.4 Positionnement du Projet 7
1.4 Positionnement du Projet 7
1.4.1 Énoncé du problème 7
1.4.2 Positionnement du travail à réaliser 7
1.4.3 Description des intervenants 10
1.5 Plan de réalisation du Projet 11
CHAPITRE 1 : PRINCIPES ET RÈGLES DE MESURE DE COSMIC-FFP 11
1. Vue d'ensemble de COSMIC-FFP 11
2. Phase de mappage 12
2.1 Identification des couches 13
2.2 Identification des frontières 14
2.3 Identification des Processus Fonctionnels 14
2.4 Identification des Groupes de données 15
2.5 Identification des Attributs de données 15
3 Phase de mesure 16
3.1 Identification des Sous-Processus Fonctionnels 17
3.2 Application de la fonction de la mesure 18
3.3 Agrégation des résultats de la mesure 18
CHAPITRE 2 : UML ET RATIONAL UNIFIED PROCESS (RUP) 19
1. UML 19
1.1 Les concepts UML 19
1.2 Les mécanismes d'extension dans UML 22
2. RATIONAL UNIFIED PROCESS 22
2.1 RUP est piloté par les cas d'utilisation 24
2.2 RUP est centré sur l'architecture 24
2.3 RUP est itératif et incrémental 25
2.4 Le cycle de vie de RUP 26
2.5 Discipline : Expression des besoins (Requirements) 27
2.6 Discipline : Analyse et Conception 30
2.7 Les couches dans RUP 40
CHAPITRE 3 : RAPPROCHEMENT COSMIC-FFP ET RUP 46
1 Discipline : Expression des besoins 46
1.1 Processus de mappage COSMIC-FFP 46
1.2 Processus de mesure de COSMIC-FFP 49
2 Discipline : Analyse et Conception 50
2.1 Analyse 51
2.2 Conception 58
3. Conclusion 59
CHAPITRE 4 : ÉTUDE DE CAS – GESTION GLOSSAIRE 61

1. Énoncé du cas 61
2. Mesure directe de l'application Glossaire à partir des FURs 61
3. Mesure de l'application à partir des artefacts RUP 64
3.1 Discipline : Expression des besoins 64
3.2 Analyse 68
4 Tableau récapitulatif 86
5 Conclusion 87
CHAPITRE 5 : ANALYSE ET EXPLOITATION DES RÉSULTATS 88
1 Processus de mesure 88
1.1 Phase de mappage 89
1.2 Activité : Phase de mesure 92
2 Processus de mesure COSMIC-FFP basé sur UML 93
2.1 Hypothèses 93
2.2 Identification des couches 94
2.3 Identification des frontières 94
2.4 Identification des processus fonctionnels 94
2.5 Identification des groupes et des attributs de données 94
2.6 Identification des sous-processus fonctionnels 95
3 Remarques 95
4 Les faiblesses de cette étude 97
5 Futures recherches 97
6 Conclusion 98
CHAPITRE 6 : DÉVELOPPEMENT DE L'OUTIL CFFP-GAUGE 99
1. Énoncé de positionnement du produit 99
2. Besoins clés des utilisateurs 99
3. Résumé des capacités 100
4. Activités de développement 102
4.1 Formalisation des besoins 102
4.2 Architecture de l'application 104
4.2.1 Schéma général de CFFP-GAUGE 105
4.2.2 Architecture en couche de CFFP-GAUGE 106
4.2.3 Le modèle d'affaire CFFP-GAUGE 108
4.2.3 Gestion de la persistance 109
5. Fonctionnalités de CFFP-GAUGE 109
5.1 L'interface GUI de CFFP-GAUGE 109
5.2 Génération automatique du modèle COSMIC-FFP 109
6. Conclusion 113
Annexe A: 114
Références : 121

1. INTRODUCTION

1.1 Objectif du document

Ce document est le rapport final du travail de synthèse du cours MGL9701 « Projet en génie logiciel» du programme 3821 « Maîtrise en génie logiciel» à l'Université du Québec à Montréal « UQAM ». Il est aussi le résultat d'une activité de recherche en domaine du génie logiciel, et particulièrement dans la mesure de la taille fonctionnelle des applications développées ou en cours de développement. Il contient les différents artefacts nécessaires pour mettre en valeur le résultat de cette activité. Une lecture de ce document peut ne pas être séquentielle, car il contient des chapitres plus ou moins indépendants. Un connaisseur de COSMIC-FFP, UML et RUP peut ne pas lire les deux premiers chapitres.

1.2 Vue d'ensemble

Actuellement, COSMIC-FFP (norme ISO 19761) définit dans son processus de mesure une phase de mappage et une phase de mesure. La phase de mappage contient plusieurs activités qui consistent à mapper les exigences fonctionnelles utilisateurs (FUR) pour créer un modèle COSMIC-FFP. Ce modèle sera une entrée pour la phase de mesure, permettant de mesurer la taille fonctionnelle du logiciel répondant aux FURs.

Le projet, défini par le présent document, s'intitule « Mesure de la taille fonctionnelle en COSMIC-FFP des logiciels développés selon le processus RUP ». Il consiste à identifier, dans sa première partie, de nouvelles règles pour les différentes activités des phases du processus de mesure de COSMIC-FFP. Ces règles ne sont pas basées directement sur des FURs, mais sur des artefacts du processus de développement RUP, qui sont eux-mêmes basés sur les exigences des utilisateurs. Cette méthode de mappage n'est pas nouvelle pour COSMIC-FFP, car pratiquement, les experts de mesure utilisent cette technique pour mesurer des applications déjà développées. Elle a l'avantage d'épargner l'effort supplémentaire consacré à la mesure, puisque la démarche est presque la même, et qui consiste à la formalisation des besoins des utilisateurs et ouvre la porte à l'automatisation de la mesure de la taille fonctionnelle, car la majorité des artefacts des processus de développement est produit actuellement à l'aide des plateformes de développement tels que Rational Rose et Together.

La deuxième partie du projet consiste à automatiser ces nouvelles règles de mappage pour générer automatiquement le modèle COSMIC-FFP et effectuer une mesure automatique en utilisant les artefacts de RUP élaborés avec Rational Rose.

Quelques travaux antérieurs ont été effectués pour répondre à ce besoin. Les particularités de ce projet, par rapport aux autres, sont :

- Redéfinir toutes les règles de mappage de COSMIC-FFP, en particulier l'identification des couches.
- Explorer tous les artefacts élaborés dans les principales activités des disciplines d'ingénierie de RUP à savoir « Expressions de besoins » et « Analyse et Conception », dans l'objectif d'utiliser les plus valables pour identifier les éléments de COSMIC-FFP.
- Redéfinir toutes les activités des différentes phases du processus de mesure COSMIC-FFP en utilisant les nouvelles règles de mappage.
- Développement d'un outil en Java et XML qui exploite les artefacts élaborés lors des activités RUP en utilisant Rational Rose pour générer automatiquement le modèle COSMIC-FFP, et déterminer, par conséquent, la taille fonctionnelle du logiciel développé ou en cours de développement.

1.3 Survol du document

Ce document est composé de deux parties :

- **Partie I**: une étude théorique qui consiste à identifier les nouvelles règles de mappage appelées aussi règles de rapprochement. Elle contient le positionnement du travail et 4 chapitres :
  - **Chapitre 1**: ce chapitre comprend une description de toutes les activités de différentes phases du processus de mesure COSMIC-FFP. Il s'agit d'une traduction directe du manuel de mesure Version 2.1 du 3 Mai 2001 [6]. Cette description est importante, car elle nous permet de trouver les règles de rapprochement en se basant sur des références sûres.
  - **Chapitre 2 :** dans ce chapitre on expose les différents aspects d'UML et de RUP en se basant sur des références considérées comme les plus importantes [8], [9] et [10]. Cette description est importante, car elle nous permet de rapprocher ces différents aspects avec celui de COSMIC-FFP.

- **Chapitre 3 :** en exploitant les chapitres 1 et 2, on exécute, ici, les différentes activités du processus de mesure COSMIC-FFP, en utilisant les artefacts produits dans les disciplines « Expressions de besoins » et « Analyse et Conception » de RUP. Les résultats, ce sont les différents éléments du modèle COSMIC-FFP qui peuvent être identifiés à partir de ces artefacts. Ceci nous ramène à proposer une première vision sur les règles de rapprochement.
- **Chapitre 4** : pour pratiquer, tout en testant les règles de rapprochement identifiées dans le chapitre précédent, on étudie un cas simple : Gestion Glossaire. Celui-ci permet d'appliquer toutes les règles.
- **Chapitre 5** : dans ce chapitre, on redéfinit, au départ le processus de mesure, par la description de toutes ces activités en se basant sur les règles de rapprochement. Puis, en s'inspirant de ce processus, on propose une première ébauche d'une formalisation des activités du processus actuel en utilisant le langage UML. À la fin, on expose les points forts et les points faibles de ce travail.

- **Partie II :** Application des règles identifiées dans la première partie pour le développement de l'outil CFFP-GAUGE.

Cette partie contient le chapitre 6 consacré au développement de l'outil CFFP-GAUGE. On définit au départ les besoins utilisateurs spécifiant ses fonctionnalités et un résumé de ses capacités. Puis, on décrit quelques activités de développement qui produisent certains artefacts utiles pour comprendre son architecture. Enfin, on détaille ses principales fonctionnalités à savoir la représentation graphique du modèle COSMIC-FFP et la génération automatique de ce modèle à partir des artefacts RUP élaborés avec Rational Rose.

1.4 Positionnement du Projet

1.4.1 Énoncé du problème

Le problème de	Mesure automatique de la taille fonctionnelle d'un logiciel selon la norme ISO-19761 COSMIC-FFP.
Affecte	❑ L'estimation et la mesure de la taille fonctionnelle d'un produit logiciel. ❑ L'estimation de l'effort de développement d'un produit logiciel.
Cela a un impact sur	❑ La fiabilité des résultats d'une mesure manuelle de la taille d'un produit logiciel. ❑ La dépendance envers les experts de mesure.
Une solution qui aura du succès pourra	❑ Augmenter la fiabilité des résultats de mesure due à la standardisation de l'application des concepts et des règles de mesure. ❑ Réduire les frais supplémentaires qui surchargent le coût d'un projet dû au recours à des experts externes de mesure de la taille fonctionnelle.

1.4.2 Positionnement du travail à réaliser

L'estimation de la taille d'un produit logiciel dans les premières phases du développement est une nécessité pour l'estimation de l'effort de développement, l'évaluation de la qualité et essentiellement pour la gestion des contrats d'impartition. Actuellement, il existe plusieurs façons d'estimer cette taille. On distingue essentiellement trois méthodes qui sont utilisées, à savoir l'estimation du nombre de lignes de code (COCOMO), de la taille fonctionnelle (IFPUG, COSMIC-FFP) ou le nombre de cas d'utilisation. Une combinaison entre ses méthodes est aussi possible, telles que les cas d'utilisation et le nombre de lignes de code, les cas d'utilisation et la taille fonctionnelle, la taille fonctionnelle et le nombre de lignes de code. Avoir une estimation en nombre de lignes de code fiable dans les premières phases de développement est un objectif difficile à atteindre. Estimer l'effort de développement selon le nombre de cas d'utilisation est aussi difficile, due à la variation de la taille des cas d'utilisation selon les produits logiciels à développer (le nombre et la complexité des scénarios des cas d'utilisation). La méthode de la taille fonctionnelle et particulièrement COSMIC-FFP montre son efficacité ces derniers temps pour résoudre ces problèmes de tous les types de logiciels aussi bien temps-réel qu'embarqués,

de gestion ou système [1]. COSMIC-FFP s'appuie sur les spécifications fonctionnelles des utilisateurs pour estimer la taille fonctionnelle du logiciel. Les méthodes contemporaines de développement de logiciels traduisent ces spécifications dans les premières phases de développement en deux formes : spécifications formelles en utilisant des langages tels que Z, Z-Object, … ou spécifications semi-formelles selon les cas d'utilisation, en utilisant particulièrement le langage UML. Les méthodes formelles sont peu utilisées sauf pour des systèmes critiques car ils sont coûteux, complexes et exigent un niveau de compétence très élevé de la part des concepteurs. L'utilisation d'UML devient de plus en plus un standard pour les processus actuels de développement, tel que RUP, car c'est une notation qui permet de représenter un système dans toutes les phases de développement de la spécification jusqu'à l'implémentation. COSMIC-FFP devient peu à peu un standard d'estimation de la taille fonctionnelle surtout après son adoption par le comité ISO comme la norme 19761:2003. Mais, elle continue à être utilisée manuellement et le résultat obtenu dépend du niveau de compréhension du mesureur des concepts et des règles de la méthode.

Pour l'automatisation du COSMIC-FFP, H. Diab et al propose, en 2001 [2], une méthode qui permet la mesure du nombre de unités de taille fonctionnelle des systèmes temps réels en utilisant des spécifications formelles écrites en langages ROOM (Rela-time Object Oriented Modeling). Bévo et al., Aout-2001 [3] puis Jenner, 2001 [4] proposent un système qui utilise les modèles UML pour extraire la taille fonctionnelle du logiciel à développer. La première méthode reste efficace pour des systèmes temps réels spécifiés en utilisant le langage ROOM. La deuxième est importante pour tous les types de système et demande un effort considérable pour l'arrimage entre les concepts de COSMIC-FFP et les modèles UML. Les règles d'arrimage proposées jusqu'à maintenant ne permettent pas de trouver l'équivalence de tous les concepts de COSMIC-FFP dans les diagrammes UML standard. UML c'est un langage qui permet de traduire les spécifications, par exemple, sous forme des modèles standard compris par tous les développeurs. Ce langage peut être enrichi, lorsqu'il est configuré par un processus de développement, par la définition des nouveaux stéréotypes permettant de modéliser tous les artefacts générés par le processus. « UML fournit un cadre conceptuel commun mais laisse aux différents acteurs du marché toute la liberté pour développer leurs méthodes et leurs outils » [7]. Par exemple, pour prendre en charge la notion de couches dans la décomposition architecturale, RUP définit le stéréotype « Layer », qui devient un élément essentiel dans le langage UML (configuration d'UML pour RUP). Pour trouver l'équivalence de tous les concepts de COSMIC-FFP dans les modèles UML, il est plus important de faire l'arrimage de ces concepts et les artefacts traduits

sous forme des modèles UML d'un processus de développement. En pratique, un mesureur COSMIC-FFP a recours souvent aux documents d'analyse et de conception pour pouvoir identifier la taille fonctionnelle du logiciel en cours de développement ou déjà développé; or ces documents sont probablement réalisés selon la notation UML, mais avant tout, sont certainement des artefacts d'un processus de développement. En plus, la mesure n'est pas nécessairement une activité ponctuelle ; la taille fonctionnelle doit être réajustée au fur et à mesure que le processus de développement avance, et par conséquent, les modèles UML seront modifiés ou d'autres modèles apparaîtront, apportant plus de détails et permettant de trouver d'autres équivalences aux concepts de COSMIC-FFP. L'intégration de l'activité de la mesure dans un processus de développement résout certains problèmes et apporte aussi plusieurs avantages au processus lui-même, tels que l'estimation de l'effort de développement et son ajustement dans les phases du processus pour la planification du projet. La solution d'automatisation n'est pas une procédure d'arrimage simple entre les concepts de COSMIC-FFP et ceux d'UML, mais elle doit se faire entre le modèle de COSMIC-FFP et les modèles produits sous formes d'artefacts d'un processus de développement et élaborés avec le langage UML. Azzouz et Abran, 2003 [5] intègrent l'activité de mesure avec COSMIC-FFP dans le processus RUP, sauf que certains concepts de COSMIC-FFP n'ont pas trouvés leurs équivalences tels que les couches et les événements.

Les objectifs de ce travail sont de trouver, d'abord, les meilleurs artefacts des modèles produits dans les différentes activités de RUP et élaborés selon la notification UML, et qui permettent de générer le modèle de COSMIC-FFP ; puis de définir les différentes règles de rapprochement entre ces artefacts et les éléments du modèle; enfin de redéfinir les activités de différentes phases du processus de mesure en utilisant ces règles.

1.4.3 Description des intervenants

Intervenant	Description	Responsabilité	Critères de succès
Alain Abran & Normand Seguin : Encadreurs du projet	Bonne expérience dans l'encadrement et l'orientation des équipes de recherche.	❑ Supervisent les activités de l'équipe du projet. ❑ Contrôlent et orientent l'équipe de recherche dans le bon sens pour assurer l'avancement.	❑ Les règles relatives à un projet de recherche sont bien respectées. ❑ Le projet respecte les délais fixés dans le plan du projet. ❑ Le résultat est convaincant et améliore la procédure de recherche dans le domaine de la mesure automatique de la taille fonctionnelle.
Alain Abran & Jean-Marc Desharnais : Experts de mesure.	Co-auteurs de la méthode COSMIC-FFP.	❑ Expliquer les principes et les règles de mesure de la méthode COSMIC-FFP. ❑ Aider l'équipe de projet à la bonne application des règles de recherche des unités de taille fonctionnelle. ❑ Vérifier que les règles et les principes de COSMIC-FFP sont bien respectés	❑ Les principes et les règles de mesure de COSMIC-FFP sont respectés. ❑ La taille fonctionnelle générée automatiquement par l'outil correspond bien à la taille réelle d'un logiciel.
Luigi Iuliani : Expert du processus RUP.	Bonne expérience de développement selon le processus RUP.	❑ Encadrer l'équipe de projet pour la compréhension des différents modèles UML générés dans les différentes phases du processus RUP. ❑ Assister les développeurs à la bonne exploitation des artefacts de RUP. ❑ Vérifier que les modèles UML ont été exploités correctement.	Les modèles UML de RUP sont exploités convenablement et permettent, sans aucune modification, à un résultat correct de mesure de la taille fonctionnelle
Mongi Sioud : Responsable du projet.	connaît les règles et les principes de COSMIC-FFP, ainsi que le Processus RUP et maîtrise l'aspect de développement des projets.	Orienté par les encadreurs et contrôlé par les experts de la mesure et de RUP, il doit mener le projet jusqu'à la fin sans faille ni erreur.	❑ Le résultat du projet améliore le processus de mesure automatique des unités de taille fonctionnelle. ❑ La satisfaction de tous les intervenants et les utilisateurs du projet. ❑ La satisfaction des correcteurs du projet et l'obtention d'une bonne note

1.5 Plan de réalisation du Projet

Étape	Activité	Temps consommés
Analyse de problème	❑ Étude des principes, des concepts et des règles de mesure de COSMIC-FFP. ❑ Étude du processus RUP et les modèles UML élaborés lors de ses activités. ❑ Étude de rapprochement entre les concepts de COSMIC-FFP et ceux des modèles UML.	30 jours
Définition des règles de rapprochement	❑ Définition des règles ❑ Expérimentation avec une étude de cas	18 jours
Développement de CFFP-GAUGE	❑ Spécifications, Analyse et Conception ❑ Implémentation et Test	15 jours
Rédaction du rapport	❑ Rédaction du rapport du projet	21 jours

CHAPITRE 1 : PRINCIPES ET RÈGLES DE MESURE DE COSMIC-FFP

1. Vue d'ensemble de COSMIC-FFP

La méthode de mesure COSMIC-FFP applique un ensemble de règles et des procédures sur un logiciel perçu d'une perspective de ses exigences fonctionnelles utilisateurs. Le résultat de l'application de ces règles et procédures est une valeur numérique qui représente la taille fonctionnelle du logiciel d'une perspective utilisateur [6].

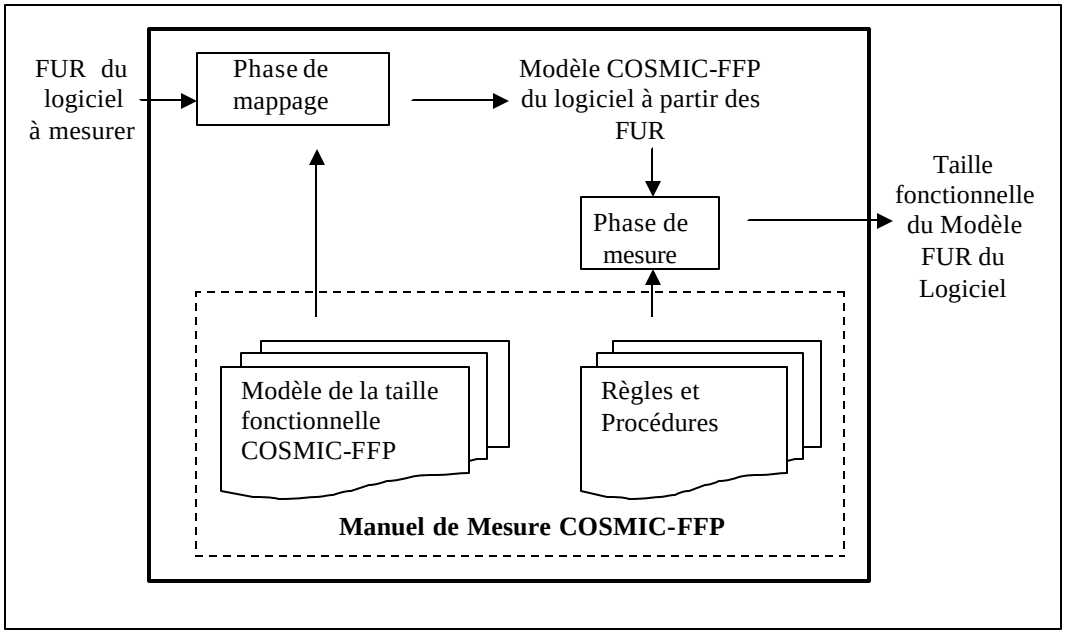

Figure 1.1 : Modèle du processus de mesure COSMIC-FFP [6]

Une exigence fonctionnelle utilisateur (FUR) est une expression ISO désignant un sous-ensemble des exigences utilisateurs, basée sur une perspective utilisateur. Une FUR représente les pratiques et les procédures à exécuter par le logiciel pour accomplir les besoins de l'utilisateur. Les FURs excluent les exigences de qualité et de technique [6].

Les exigences fonctionnelles utilisateurs d'un logiciel à mesurer doivent être mappées à un modèle spécifique en utilisant les concepts de COSMIC-FFP, puis on mesure la taille en utilisant les règles et les principes de la méthode.

2. Phase de mappage

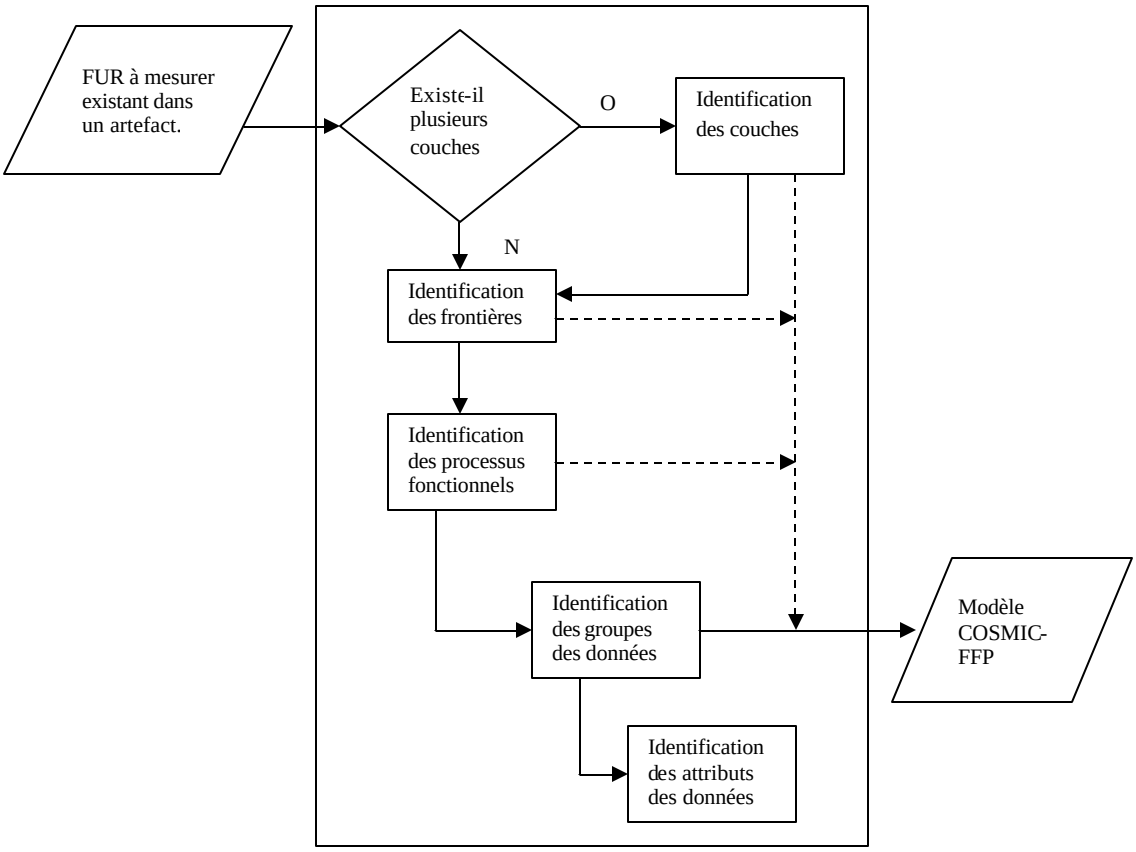

Figure 1.2 : Processus de mappage de COSMIC-FFP [6]

La procédure de mesure est applicable à plusieurs types d'artefact dérivés à partir des exigences fonctionnelles utilisateurs. Les règles de mesure doivent être spécifiques à chaque type d'artefact utilisé.

2.1 Identification des couches

Définitions	Une couche est le résultat d'une partition fonctionnelle de l'environnement logiciel incluant tous les processus fonctionnels exécutés dans le même niveau d'abstraction. Dans un environnement multicouches, un logiciel dans une couche échange les données avec un logiciel dans une autre couche à travers ces processus fonctionnels respectifs.
Principes [6]	a) un logiciel dans toutes les couches livre des fonctionnalités à ses utilisateurs. b) Un logiciel dans une couche subordonnée fournit des services fonctionnels au logiciel des couches clientes. c) Un logiciel dans une couche subordonnée peut être exécuté sans assistance du logiciel d'une couche cliente. d) Un logiciel dans une couche cliente peut ne pas s'exécuter correctement si le logiciel dans une couche subordonnée de laquelle il dépend n'est pas exécuté comme il faut. e) Un logiciel dans une couche cliente n'utilise pas nécessairement toutes les fonctionnalités fournies par le logiciel dans une couche subordonnée. f) Un logiciel dans une couche subordonnée peut être une couche cliente d'une perspective d'une troisième couche subordonnée. g) Les logiciels dans les couches client et subordonnées peuvent partager et échanger physiquement des données. Cependant, le logiciel dans chaque couche doit interpréter différemment les données. h) Un logiciel qui partage les données avec un autre logiciel n'est pas considéré dans des couches différentes s'il interprète de façon identique les données qu'il partage.
Règles [6]	a) les paquetages des services fonctionnels tels que les SGBD, les interfaces GUI, les systèmes d'opérations et les unités drivers sont considérés généralement comme des couches distinctes. b) Si un logiciel est conçu en utilisant un paradigme architectural connu, utiliser ce paradigme pour identifier les couches. c) En cas de doute, utiliser le concept de couplage pour distinguer entre les interactions des couches.

2.2 Identification des frontières

Définitions [6]	❑ Une frontière d'une pièce de logiciel est la limite conceptuelle entre cette pièce et son environnement dans laquelle elle opère. La frontière permet au mesureur de distinguer, sans ambiguïté, ce qui est inclus à l'intérieur du logiciel à mesurer et ce qui fait partie de son environnement. ❑ Un Utilisateur d'une pièce logiciel peut être un être humain, un logiciel ou une unité matérielle (device) qui interagit avec le logiciel.
Principes [6]	a) Par définition, il existe une frontière entre chaque paire de couches adjacentes identifiées; et il peut y avoir une frontière entre des pièces de logiciel distinctes dans la même couche si elles échangent des données en utilisant une communication « peer-to-peer ».
Règles [6]	a) Commencer par identifier les événements déclencheurs, puis identifier les processus fonctionnels initiés par ces événements. Une frontière réside entre les événements déclencheurs et ces fonctions. b) Identifier les unités E/S utilisées par le logiciel à mesurer pour interagir avec chaque type d'utilisateurs. Établir un rapport entre les unités identifiées et les fonctions fournies par le logiciel. Une frontière réside entre ces fonctions et ces unités. c) Pour les logiciels temps-réels et techniques, utiliser les concepts de couches pour identifier les frontières.

2.3 Identification des Processus Fonctionnels

Définitions [6]	❑ Un processus fonctionnel est un ensemble unique de mouvements des données (Entry, Exit, Read, Write) implémentant un ensemble cohésif et logiquement indivisible des FURs. Il est déclenché directement ou indirectement à travers un « Utilisateur » par un « Événement » et il se termine quand il aura exécuté tout ce qui est exigé en réponse à l'événement déclencheur. ❑ Un Événement déclencheur se produit en dehors de la frontière du logiciel à mesurer et initie un ou plusieurs processus fonctionnels. S'il existe plusieurs couches séparées par des frontières, les événements déclencheurs peuvent se produire dans une couche et initier des processus fonctionnels appartenant à une autre couche.
Principes [6]	a) Un processus fonctionnel est dérivé au moins d'une FUR identifiable b) Un processus fonctionnel est initié lorsqu'un événement déclencheur se produit. c) Un processus fonctionnel contient au moins deux mouvements de données, une Entrée et une sortie ou une écriture. d) Un processus fonctionnel ne peut pas contenir plus qu'un seul état d'attente auto-induit (qui peut se produire quant il est accompli). e) Un processus fonctionnel appartient à une, et seulement une, couche.
Règles [6]	a) les sous-ensembles d'un événement déclencheur ne sont pas considérés comme des événements déclencheurs différents. b) Dans le contexte d'un logiciel temps réel, un processus fonctionnel est aussi initié par un événement déclencheur. Il se termine quand un état asynchrone est atteint (équivalent à un état d'attente auto-induit).

2.4 Identification des Groupes de données

Définitions [6]	❑ Un groupe de données est un ensemble d'attributs des données distinct, non vide, non ordonné et non redondant où chaque attribut de donnée décrit un aspect complémentaire du même objet d'intérêt. Un groupe de données est caractérisé par sa persistance. ❑ La persistance d'un groupe de données est décrit par le combien de temps le groupe des données est retenu dans le contexte d'une FUR. On distingue trois types de persistance : o Transitoire : le groupe de données ne survit pas après la transaction qui l'utilise. o Court : le groupe de données survit après la transaction qui l'utilise, mais il ne survit pas après que le logiciel qui l'utilise cesse d'opérer. o Indéfini : le groupe de données survit même après que le logiciel qui l'utilise cesse d'opérer. ❑ En pratique COSMIC-FFP distingue entre un groupe de données Transitoire de Persistant (Court ou Indéfini).
Principes [6]	a) Un groupe de données doit être matérialisé dans le système informatique qui supporte le logiciel. b) Chaque groupe de données doit être unique et distinguable à travers sa collection unique des attributs de données. c) Les groupes de données sont directement liés aux objets décrits dans les FUR.
Règles [6]	a) Pour un logiciel de type MIS, un groupe de données est identifié par chaque entité trouvée dans le modèle normalisé de données du logiciel à mesurer. Ce sont habituellement des groupes de données présentant une persistance indéfinie et le logiciel doit stocker les données des entités concernées.

2.5 Identification des Attributs de données

Définitions [6]	❑ Un attribut de données est la plus petite parcelle d'information, dans un groupe de données identifié, portant une signification de la perspective des FURs. Il existe trois types d'attributs de données, dont deux seulement sont valides pour COSMIC-FFP : une donnée qui décrit ou représente le monde réel des utilisateurs et une donnée contextuelle qui spécifie quoi, quand et comment une donnée est traitée par le logiciel à mesurer. Par contre la donnée invalide est celle qui est créée pour le besoin d'implémentation technique.
Principes [6]	a) un attribut de données doit présenter un type valide de données. b) Un attribut de données doit présenter la plus petite partie de données référencée par le logiciel à mesurer d'une perspective des FURs.
Règles [6]	a) MIS : un élément du dictionnaire de données ou qui apparaît dans le modèle de données conceptuel ou logique. b) Temps-réel : un signal de tension reçu d'une sonde unique.

3 Phase de mesure

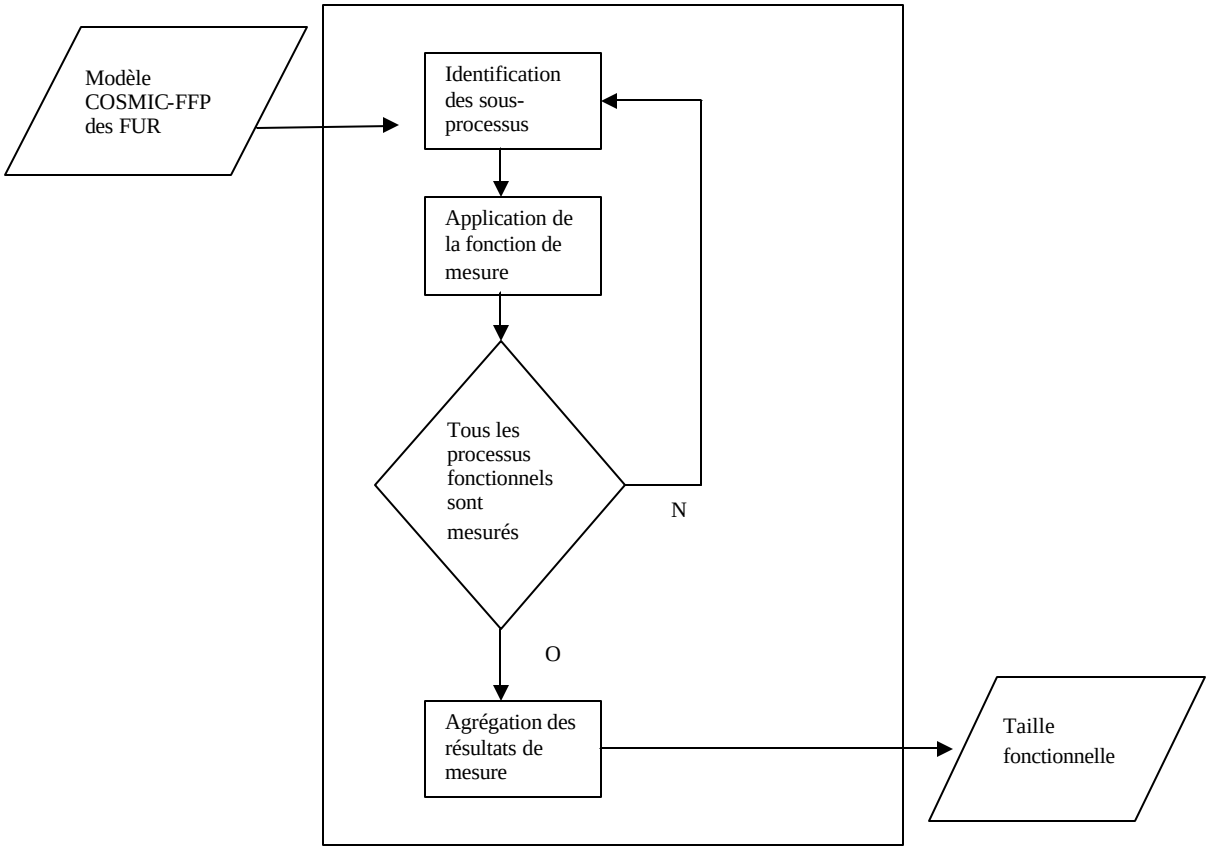

Figure 1.3 : Processus de mesure de COSMIC-FFP [6]

3.1 Identification des Sous-Processus Fonctionnels

Définitions [6]	❑ Un sous-processus de COSMIC-FFP est un mouvement de données qui se produit pendant l'exécution d'un processus fonctionnel. Il existe quatre types de sous-processus : Entrée, Exit, Lecture, Écriture. ❑ Une ENTRÉE (E) est un mouvement des attributs de données d'un même groupe de données du côté de l'utilisateur de la frontière logiciel à l'intérieur de la frontière de logiciel. Une Entrée ne met pas à jour les données qu'elle déplace. ❑ Un EXIT (X) est un mouvement des attributs de données d'un même groupe de données à l'intérieur de la frontière du logiciel au côté de l'utilisateur de la frontière logiciel. Un Exit ne lit pas les données qu'elle déplace. ❑ Une LECTURE (R) fait référence aux attributs de données d'un même groupe de données. Fonctionnellement, une Lecture apporte les données stockées, dans l'extension du processus fonctionnel auquel elle appartient. ❑ Une ÉCRITURE (W) fait référence aux attributs de données d'un même groupe de données. Fonctionnellement, un sous-processus d'Écriture envoie des données se trouvant à l'intérieur du processus fonctionnel auquel il appartient au stockage.
Principes [6]	❑ Sous processus de type **Entrée** : a) Le sous-processus reçoit les attributs de données se trouvant en dehors de la frontière du logiciel, du côté de l'utilisateur. b) Le sous-processus reçoit les données seulement d'un groupe de données. Si les données sont reçues de plus d'un groupe de données, identifiez une ENTRÉE pour chaque groupe. c) Le sous-processus n'est pas un Exit, ni une Lecture et ni une Écriture. ❑ Sous processus de type **Exit** : a) Le sous-processus envoie les attributs de données en dehors de la frontière de logiciel, du côté d'utilisateur b) Le sous-processus envoie les données appartenant à un seul groupe de données. Si les données appartenant à plus d'un groupe de données sont envoyées en dehors de la frontière du logiciel, identifiez un Exit pour chaque groupe de données référencé. c) Le sous-processus n'est pas une Entrée, ni une Lecture et ni une Écriture. ❑ Sous processus de type **Lecture** : a) Le sous-processus récupère les attributs de données d'un groupe de données stocké. b) Le sous-processus fait référence aux attributs de données appartenant à un seul groupe de données. Si plus d'un groupe de données est référencé, identifiez un Exit pour chaque groupe de données. c) Le sous-processus n'est pas une Entrée, ni un Exit et ni une Écriture. ❑ Sous processus de type **Écriture** : a) Le sous-processus stocke les attributs de données dans un groupe de données du côté dédié au stockage pour le logiciel. b) Le sous-processus modifie la valeur des attributs de données appartenant à un seul groupe de données. Si plus qu'un groupe de données est référencé, identifiez une Écriture pour chaque groupe de données. Le sous-processus n'est pas une Entrée, ni un Exit et ni une Lecture.

3.2 Application de la fonction de la mesure

Définitions [6]	❑ La fonction de mesure de COSMIC-FFP est une fonction mathématique qui assigne une valeur à son argument basé sur la norme de mesure de COSMIC-FFP. L'argument de la fonction de mesure de COSMIC-FFP est le sous-processus. ❑ L'unité de mesure de COSMIC-FFP, 1 Cfsu (Cosmic Functional Size Unit), est définie en tant qu'un mouvement de données élémentaire .

3.3 Agrégation des résultats de la mesure

Principes [6]	Dans chaque couche identifiée, les tailles fonctionnelles de différents sous-processus sont agrégées dans une valeur fonctionnelle unique de taille en les ajoutant arithmétiquement ensemble : $\text{Taille}_{\text{Cfsu}}(\text{couche}_i) = \sum \text{taille}(\text{entrées}_i) + \sum \text{taille}(\text{exits}_i) + \sum \text{taille}(\text{lectures}_i) + \sum \text{taille}(\text{Écritures}_i)$.

CHAPITRE 2 : UML ET RATIONAL UNIFIED PROCESS (RUP)

Nous définissons dans ce chapitre tous les concepts d'UML et de RUP.

1. UML

Selon G. Booch et al, 1998 [8], UML est défini comme étant un langage de visualisation, spécification, et documentation des artefacts d'un système de logiciel.

Schématiquement, UML peut être défini sur trois plans [7]:

- Les concepts, dont la sémantique est complètement et formellement définie
- Les diagrammes, utilisés pour spécifier les besoins et les systèmes
- Un certain nombre de mécanismes d'extension, pour intégrer aux processus des stéréotypes spécifiques aux différentes méthodes et/ou contextes applicatifs.

1.1 Les concepts UML

1.1.1 Cas d'utilisation, Acteur et Scénarios

Cas d'utilisation	❑ Un cas d'utilisation est une description du comportement du système, en termes de séquences et d'actions. Il doit produire un résultat observable de valeur à un **Acteur**. Un cas d'utilisation contient tous les flux alternatifs des événements reliés pour produire ce « résultat observable de valeur ». Formellement, un cas d'utilisation définit un ensemble d'**instances de cas d'utilisation** ou **scénarios** [9]. ❑ C'est une spécification d'une séquence d'actions, incluant les variantes, qu'un système (ou autre entité) peut exécuter, interagissant avec les **Acteurs** d'un système [8].
Acteur	❑ Un ensemble cohérent de rôles que les utilisateurs jouent lorsqu'ils interagissent avec les cas d'utilisation. Un Acteur possède un rôle pour chaque cas d'utilisation avec lequel il communique [8].
Instance de cas d'utilisation	❑ Une instance d'un cas d'utilisation est plus spécifique qu'un cas d'utilisation - les acteurs sont remplacés par des personnes spécifiques ou des instances des acteurs, et uniquement un seul chemin qui est pris à travers les flux alternatifs possibles du cas d'utilisation [9].
Scénario	❑ Un scénario est une description d'une instance d'un cas d'utilisation, c'est un sous-ensemble de cas d'utilisation [9]. ❑ Un scénario est une séquence spécifique d'actions qui illustre les comportements. Il peut être utilisé pour illustrer une **interaction** ou une exécution d'une instance d'un cas d'utilisation [8].
Interaction	❑ Une interaction est de spécifier comment les stimuli sont envoyés entre les instances pour exécuter des tâches spécifiques [8].

1.1.2 Objets, Types, Classes et Interfaces

Classe	❑ une classe est une description d'un ensemble d'objets qui partagent les mêmes **attributs**, **opérations**, **méthodes**, relations et sémantiques. Une classe peut utiliser un ensemble d'**interfaces** pour spécifier des collections des opérations qui fournissent à son environnement [8].
Type	❑ un type est une description d'un ensemble d'entités qui partagent des caractéristiques communes de relations, d'attributs et des sémantiques [9].
Interface	❑ Une interface est une collection d'opérations qui est utilisée pour spécifier un service d'une classe ou un composant [9]. ❑ Une interface est un ensemble nommé d'opérations qui caractérise le comportement d'un élément [8].
Instance	❑ une instance est une entité individuelle qui satisfait la description d'une classe ou d'un type [9].
Objet	❑ un objet est une entité ayant des frontières et d'identité bien définies qui encapsule un état et un comportement. L'état est représenté par les attributs et les relations, le comportement est représenté par les opérations, les méthodes et les machines d'état. Un objet est une instance d'une classe [8].
Attribut	❑ un attribut défini par une classe représente une propriété nommée de cette classe ou de ses objets. Un attribut possède un Type qui définit le type de ses instances [9].
Opération	❑ une opération est un service qui peut être demandé d'un objet pour effectuer un comportement [8].
Méthode	❑ une méthode est une implémentation d'une opération. Elle spécifie l'algorithme ou la procédure associée à une opération [8].
Message	❑ un message est une communication d'information d'une instance à une autre. Il peut spécifier un signal ou l'appel d'une opération.

1.1.3 Relations, Associations et Agrégation

Relation	❑ Une relation est une connexion sémantique entre les éléments d'un modèle. Les associations et les généralisations sont des exemples de relation [8].
Association	❑ L'association est une relation sémantique entre deux ou plusieurs classificateurs (classes, types, interfaces, composants) qui spécifient les connections entre ses instances [8].
Agrégation	❑ L'agrégation est une forme spéciale d'association qui spécifie une relation entier-partie (Whole-part) entre l'agrégat (entier) et un composant (partie) [8].

1.1.4 État, Événement et Transitions

État	❑ Un état est une condition ou une situation durant la vie d'un objet pour satisfaire certaines conditions, exécute certaines activités, ou attend certains événements [8].
Machine d'états	❑ Un comportement qui spécifie les séquences des états qu'un objet ou une interaction s'entreprendre à travers son cycle de vie en réponse à un événement [8].
Événement	❑ Un événement est une spécification d'une occurrence significative qui se repère dans le temps et dans l'espace. Dans le contexte du diagramme d'état, un événement est une occurrence qui déclenche une transition [8].
Transition	❑ Une transition est une relation entre deux états indiquant qu'un objet dans le premier état exécute certaines actions spécifiées et entre dans le second état quand un événement se produit et des conditions spécifiées sont satisfaites [8].

1.1.5 Paquetages et Composants

Paquetage	❑ Le paquetage est un mécanisme général qui a pour but d'organiser les éléments dans des groupes [8].
Composant	❑ Un composant est une partie physique du système qui met en paquetage l'implémentation en réalisant un ensemble d'interface. Un composant représente une partie physique d'implémentation du système, incluant le code du logiciel (source, binaire ou exécutable) ou l'équivalent des scripts ou des fichiers de commande.

1.1.6 Les diagrammes UML

La modélisation des systèmes UML s'articule sur deux modes de représentation : le mode statique et le mode dynamique. Le mode statique s'appuie sur 5 diagrammes et le mode dynamique s'appuie sur 4 diagrammes.

Mode statique	**Diagramme de cas d'utilisation**	Le diagramme de cas d'utilisation montre la relation entre les acteurs et les cas d'utilisation dans le système [8].
	Diagramme de classes	Le diagramme de classes montre une collection d'éléments d'un modèle déclaratif (statique) tels que les classes, les types, et leurs contenus et leurs relations [8].
	Diagramme d'objets	Le diagramme d'objets renferme les objets et leurs relations à un instant donné. Il peut être considéré comme un cas spécial d'un diagramme de classe ou d'un diagramme de collaboration [8].
	Diagramme de composants	Le diagramme de composants montre les organisations et les dépendances entre les composants [8].
	Diagramme de déploiement	Le diagramme de déploiement montre la configuration des nœuds à l'exécution des processus et les composants, les processus et les objets en vie pendant cette exécution [8].

Mode dynamique	**Diagramme d'état**	Le diagramme d'états montre la machine d'état [8].
	Diagramme d'activités	Le diagramme d'activités est un cas spécial d'une machine d'état utilisé pour la modélisation des processus impliquant un ou plusieurs classificateurs (Synonyme graphe d'activité) [8].
	Diagramme de séquence	Le diagramme de séquence montre les interactions d'objets arrangées en une séquence de temps. En particulier, il montre les objets qui participent à une interaction et la séquence des messages échangés. à la différence de diagramme de collaboration, un diagramme de séquence inclut les séquences de temps et non la relation entre les objets. Le diagramme de séquence peut exister sous un format générique en décrivant tous les scénarios possibles et sous un format d'instance en décrivant le scénario actuel. Le diagramme de séquence et le diagramme de collaboration expriment la même information, mais ils la montrent de différentes manières [8].
	Diagramme de collaboration	Le diagramme de collaboration montre les interactions organisées autour de la structure d'un modèle, utilisant les classificateurs et les associations ou les instances et les liens [8].

1.2 Les mécanismes d'extension dans UML

L'utilisation d'UML doit être configurée en fonction du contexte méthodologique, applicatif et technique [7]. Cette configuration s'appuie sur des mécanismes d'extension et particulièrement, sur la notion de stéréotype. Un stéréotype est défini comme étant un nouveau type de modélisation d'un élément qui étend la sémantique du méta-modèle. Les stéréotypes doivent être basés sur certains types ou classes existants dans le méta-modèle. Ils doivent étendre les sémantiques et non la structure des types et des classes préexistants. Certains stéréotypes sont prédéfinis dans UML, d'autres peuvent être définis par les utilisateurs [8].

La notion de stéréotype permet [7] :

- D'adapter le langage UML aux contextes applicatifs et méthodologiques en capitalisant les acquis et en enrichissant la démarche.
- De formaliser et de systématiser la transition entre les phases d'analyse (stéréotypes applicatifs) et de conception (design pattern).

2. RATIONAL UNIFIED PROCESS

RUP est un processus de développement logiciel, il fournit une approche disciplinaire d'assignation des tâches et des responsabilités dans une organisation de développement [9]. Son objectif est d'assurer la production d'un logiciel de haute qualité qui répond aux besoins des utilisateurs finaux, dans le temps et le budget prévus. RUP regroupe les activités à mener pour

transformer les besoins d'un utilisateur en système logiciel. RUP capture plusieurs bonnes pratiques de développement moderne de logiciel, il est à base de composants, utilise le langage UML, piloté par les cas d'utilisation, centré sur l'architecture, itératif et incrémental.

Modèle	❑ Un modèle est une abstraction d'un système décrivant le système modélisé d'un certain point de vue et à un certain niveau d'abstraction [11]. ❑ Un modèle est une abstraction sémantiquement fermée d'un système. C'est une vue autonome. en ceci qu'un utilisateur d'un modèle n'a besoin d'aucune autre information (en provenance d'autres modèles, par exemple) pour l'interpréter [10].
Artefact	Un artefact est une information tangible (1) créée, modifiée et utilisée par les travailleurs de leurs activités, (2) représentant un domaine de responsabilité et (3) susceptible d'être placée sous un contrôle de version à part. Un artefact peut être un modèle, un élément de modèle ou un document.
Modèle de cas d'utilisation	Le modèle de cas d'utilisation d'un système contient les acteurs, les cas d'utilisation et les relations qu'ils entretiennent les uns avec les autres. C'est une vue externe du système formulé dans le langage client [10].
Modèle d'analyse	Un model objet, décrivant la réalisation des cas d'utilisation, sert comme une abstraction au modèle de conception.
Modèle de conception	Le modèle de conception est un modèle d'objet décrivant la réalisation physique des cas d'utilisation, en s'attachant aux effets conjugués des exigences fonctionnelles et non fonctionnelles et des autres types de contraintes sur le système en question [10].
Modèle d'implémentation	Le modèle d'implémentation décrit la façon dont les éléments du modèle de conception, sont implémentés par des composants de type fichiers de source, exécutables, etc. il présente également l'agencement des composants en fonction des mécanismes de structuration et de modularisation disponible dans l'environnement d'implémentation, et le ou les langages d'implémentation utilisés, ainsi que les dépendances qui lient les composants les uns aux autres.
Modèle des tests	Le modèle des tests décrit les conditions dans lesquelles les composants exécutables du modèle d'implémentation subissent des tests d'intégration et des tests système. Il peut également indiquer les modalités de tests de certains aspects spécifiques du système [9].
Système	Un système est une collection d'unités qui sont organisées pour accomplir un but spécifique. Un système peut être décrit par un ou plusieurs modèles, peuvent être par différents points de vue [9].
Sous-système	Un sous-système est un groupage des éléments d'un modèle. Le comportement de sous-système est fourni par les éléments qu'il contient [9], [10].
Couche	Une couche est un regroupement des classificateurs et des packages dans le même niveau d'abstraction. Une couche représente un découpage horizontal à travers une architecture, par contre une partition représente un découpage vertical.
Événement	Un événement peut être externe ou interne, les événements externes sont ceux qui se passent entre le système et ses acteurs. Les événements internes sont ceux qui se passent entre les objets en vie du système.

2.1 RUP est piloté par les cas d'utilisation

RUP est centré sur l'utilisateur, car l'objectif principal d'un système logiciel est de rendre service à ses utilisateurs, il faut, par conséquent, bien comprendre les désirs et les besoins des futurs utilisateurs [10]. L'utilisateur représente une personne ou une chose dialoguant avec le système en cours de développement. Ce type d'interaction est appelé cas d'utilisation. Les cas d'utilisation font apparaître les besoins fonctionnels, et leur ensemble constitue le modèle des cas d'utilisation qui décrit les fonctionnalités complètes du système.

Les cas d'utilisation ne sont pas un simple outil de spécification des besoins du système. Ils vont complètement guider le processus de développement à travers l'utilisation de modèles basés sur l'utilisation du langage UML [10] :

- A partir du modèle des cas d'utilisation, les développeurs créent une série de modèles de conception et d'implémentation réalisant les cas d'utilisation.
- Chacun des modèles successifs est ensuite révisé pour en contrôler la conformité par rapport au modèle des cas d'utilisation.
- Enfin, les testeurs testent l'implémentation pour s'assurer que les composants du modèle d'implémentation mettent correctement en oeuvre les cas d'utilisation.

Les cas d'utilisation garantissent la cohérence du processus de développement du système, et ils doivent absolument être développés en paire avec l'architecture du système.

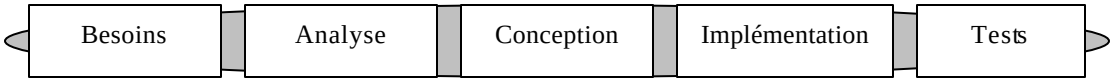

Figure 2.1 : Les principales disciplines sont liées grâce aux cas d'utilisation

2.2 RUP est centré sur l'architecture

Selon IEEE, une architecture est le concept du niveau le plus haut du système dans son environnement. Dans RUP l'architecture d'un système logiciel (à un instant donné) c'est son organisation ou sa structure des composants significatifs interagissant à travers des interfaces. Ces composants sont composés de façon récursive par des petits composants et interfaces [9].

Une architecture de RUP est représentée par 5 vues, appelée « modèle des vues 4+1 » [9] :

Vue de cas d'utilisation : contient les cas d'utilisation et les scénarios qui renferment le comportement significatif architecturel, les classes et les risques techniques. C'est un sous-ensemble du modèle de cas d'utilisation.

Vue logique : contient les classes de conception les plus importantes et leur organisation dans des packages et des sous-systèmes, et l'organisation de ces packages et de ces sous-systèmes en couches. Il contient aussi la réalisation des cas d'utilisation. C'est un sous-ensemble du modèle de conception.

Vue d'implémentation : contient une vue générale du modèle d'implémentation et son organisation en termes de modules dans des packages et des couches. C'est un sous-ensemble du modèle d'implémentation.

Vue de processus : contient la description des tâches (processus et threads), leurs interactions et leurs configurations, et l'allocation des objets et des classes de conception aux tâches. C'est un sous-ensemble du modèle de conception.

Vue de déploiement : contient la description des nœuds physiques pour les configurations des principales plateformes, et l'allocation des tâches (du vue de processus) aux nœuds physiques. C'est un sous-ensemble du modèle de déploiement.

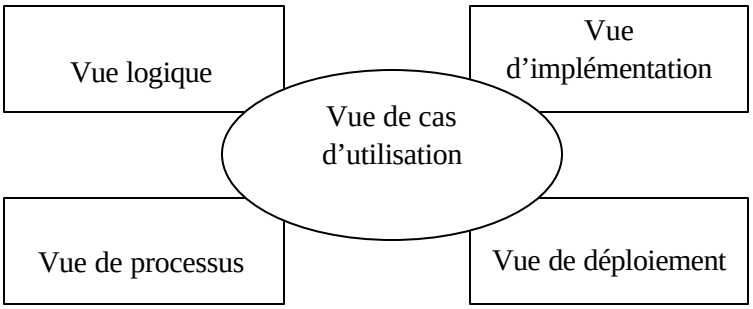

Figure 2.2 : l'architecture Vue « 4 + 1 »

2.3 RUP est itératif et incrémental

Le développement d'un produit logiciel est découpé en plusieurs parties qui sont considérées comme des minis projets. Chacun d'entre eux représente une itération et donne lieu à un incrément.

Une itération désigne la succession des étapes de l'enchaînement d'activités, tandis qu'un incrément correspond à une avancée dans les différents stades de développement sanctionné par un artefact [10].

2.4 Le cycle de vie de RUP

RUP répète un certain nombre de fois une série de cycles. Tout cycle se conclue par la livraison d'une version du produit aux clients et s'articule en 4 phases : création, élaboration, construction et transition, chacune d'entre elles se subdivise à son tour en itérations. Chaque cycle se traduit par une nouvelle version du système. Ce produit se compose d'un corps de code source réparti sur plusieurs composants pouvant être compilés et exécutés et s'accompagne de manuels et de produits associés [10].

Phase de création (Inception) : Traduit une idée en vision de produit fini et présente une étude de rentabilité pour ce produit :

- Que va faire le système pour les utilisateurs ?
- A quoi peut ressembler l'architecture d'un tel système ?
- Quels sont l'organisation et les coûts du développement de ce produit ?

On fait apparaître les principaux cas d'utilisation.

L'architecture est provisoire, identification des risques majeurs et planification de la phase d'élaboration [10].

Phase d'élaboration : Permet de préciser la plupart des cas d'utilisation et de concevoir l'architecture du système. L'architecture doit être exprimée sous forme de vue de chacun des modèles : apparition d'une architecture de référence.

A l'issue de cette phase, le chef de projet doit être en mesure de prévoir les activités et d'estimer les ressources nécessaires à l'achèvement du projet [10].

Phase de construction : Moment où l'on construit le produit. L'architecture de référence se métamorphose en produit complet, elle est maintenant stable. Le produit contient tous les cas d'utilisation que les chefs de projet, en accord avec les utilisateurs ont décidé de mettre au point pour cette version [10]. Celui-ci peut avoir encore des anomalies qui seront résolues lors de la phase de transition [10].

Phase de transition : Le produit est en version bêta. Un groupe d'utilisateurs expérimentés essaie le produit et rend compte des anomalies et des défauts constatés. Cette phase suppose des activités comme la fabrication, la formation des utilisateurs clients, la mise en oeuvre d'un service d'assistance et la correction des anomalies identifiées après la livraison.

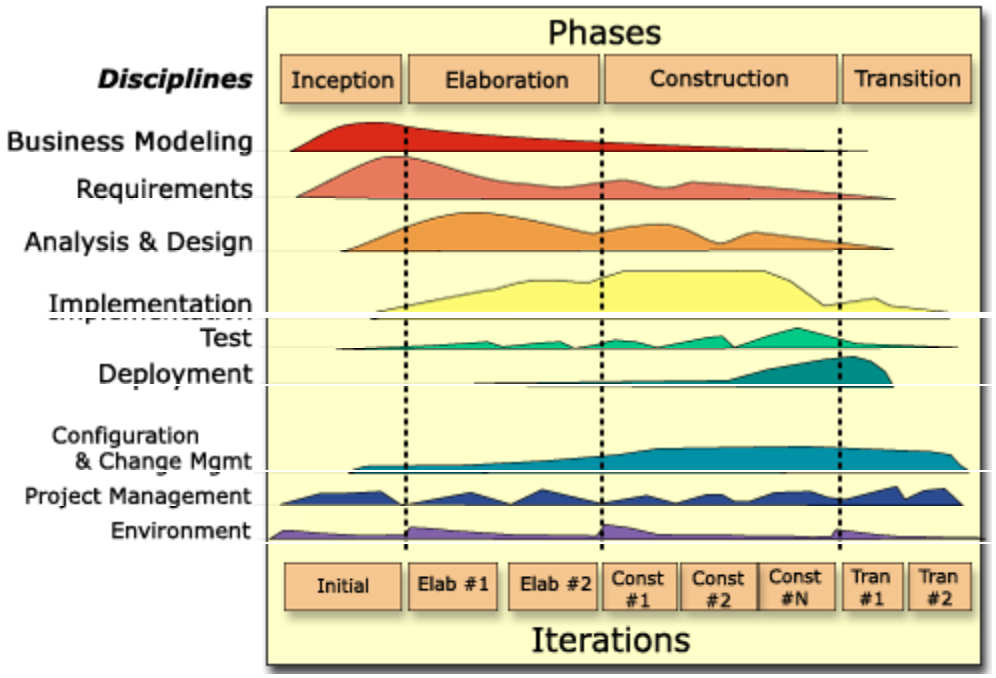

Figure 2.3 : ***les phases et les disciplines dans RUP***

Chaque phase contient une ou plusieurs itérations. Chaque itération est considérée comme un mini projet et contient des disciplines appelées aussi enchaînements d'activités (avec une intensité variable selon les phases Figure 2.3). On distingue deux types de discipline : d'ingénierie (les 5 premiers) et de gestion (les 3 derniers), on s'intéresse dans ce travail seulement à la discipline d'ingénierie, et en particulier aux expressions des besoins (Requirements) et d'analyse et conception Pour chaque discipline, on étudie seulement les principales activités permettant de générer les principaux artefacts.

2.5 Discipline : Expression des besoins (Requirements)

Cette discipline a pour objectif [9] :

- D'établir et maintenir un accord avec les clients et les intervenants sur ce que le système doit faire.
- permettre aux développeurs du système de bien comprendre les exigences
- définir les frontières et les délimitations du système.
- Fournir une base qui permet de planifier les contenus des itérations
- Fournir la base pour l'estimation du coût et le délai de développement du système

- Définir les interfaces utilisateurs du système, en se focalisant sur les besoins et les buts des utilisateurs.

2.5.1 Les principaux artefacts

Artefact : Acteur

Un acteur représente la partie extérieure du système qui collabore avec celui-ci. Il peut être un être humain, une horloge, … l'identification de tous les acteurs permet d'identifier l'environnement externe du système [10].

Figure 2.4** : **artefact Acteur dans RUP

Artefact : Cas d'utilisation

Chaque usage exercé par les utilisateurs est représenté par un cas d'utilisation. Les cas d'utilisation sont les fonctionnalités qu'offre le système afin de produire un résultat satisfaisant pour ces acteurs [10].

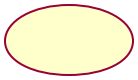

Figure 2.5** : **artefact Cas d'utilisation dans RUP

Artefact : modèle de cas d'utilisation

Le modèle des cas d'utilisation (Voir définition) regroupe tous les cas d'utilisation et les acteurs. C'est un modèle du système projeté vers ses fonctions et son environnement. Il va servir à un contrat entre le client et les développeurs [9]. Il offre un moyen, à la fois systématique et intuitif, de saisir les besoins fonctionnels sous l'angle particulier de l'intérêt qu'ils présentent pour chaque utilisateur [10]. Il est représenté par un ou plusieurs diagrammes de cas d'utilisation définissant la relation entre les acteurs et les cas d'utilisation.

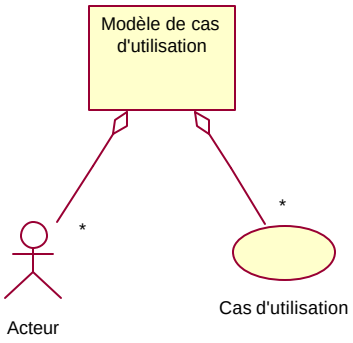

Figure 2.6 : ***artefact Modèle de cas d'utilisation dans RUP***

Artefact: story-board d'un cas d'utilisation

C'est une description logique et conceptuelle sur comment un cas d'utilisation est pourvu par l'interface utilisateur, incluant l'interaction exigée entre les acteurs et le système [9]. Cet artefact est utilisé pour comprendre les raisons des exigences de l'interface utilisateur, incluant les exigences de son utilisabilité [9].

Un story-board d'un cas d'utilisation est décrit en terme de classes de frontières et ses relations statiques et dynamiques telles que les agrégations, les associations et les liens. Chaque classe de frontière est une représentation de haut niveau d'une fenêtre ou un construit similaire dans l'interface utilisateur [9].

Un story-board de cas d'utilisation est réalisé par les classes frontières et l'interaction de leurs objets. Un diagramme de classes doit être créé pour illustrer la participation des classes frontières et leurs relations pour la réalisation du story-board du cas d'utilisation.

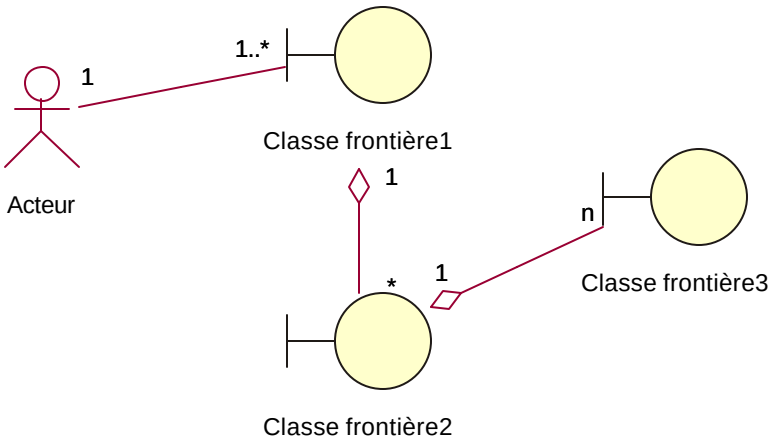

Figure 2.7 : diagramme de classes frontières participantes à un story-board de cas d'utilisation

Pour illustrer les objets frontières participants dans le story-board d'un cas d'utilisation, et leurs interactions avec l'utilisateur, un diagramme de collaboration ou de séquence est utilisé.

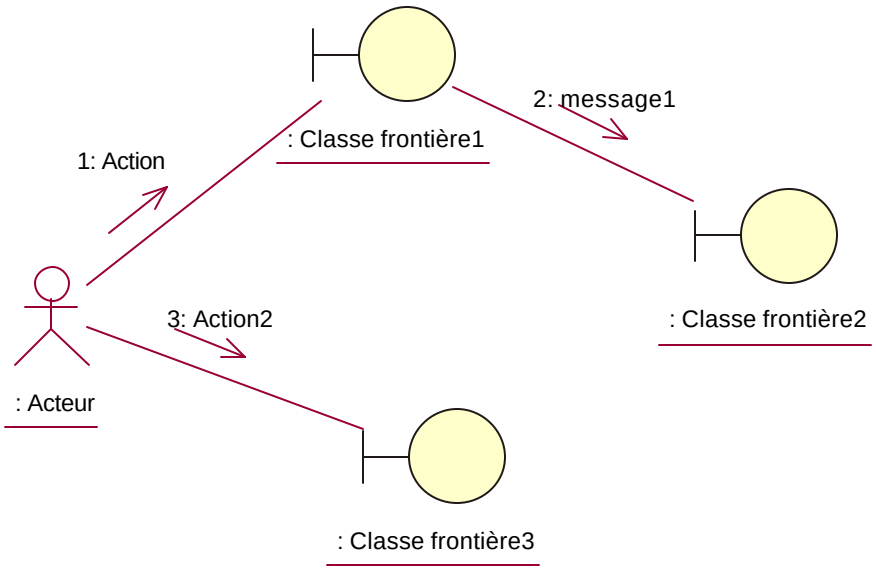

Figure 2.8 : diagramme de collaboration des classes frontières participantes à un story-board de cas d'utilisation

2.6 Discipline : Analyse et Conception

L'objectif de cette discipline est de [9]:

- Transformer les exigences en concevant comment le système doit être.
- Développer une architecture robuste du système.
- Adapter la conception pour joindre l'environnement d'implémentation.

Cette discipline contient deux « sous-disciplines » d'activités : l'analyse et la conception

2.6.1 Analyse

Cette sous-discipline se consacre à l'analyse des besoins décrits dans l'expression des besoins, en les affinant et en les structurant par des classes et des paquetages stéréotypés, donnant une structure à la vue interne [10].

L'analyse des besoins et des exigences sous la forme d'un modèle d'analyse permet de [10] :

- Livrer des besoins plus précis que celle du modèle de cas d'utilisation
- Introduire un plus grand formalisme et servir de base à une réflexion sur les mécanismes internes du système.
- Structurer les besoins et les exigences sous une forme qui en facilite la compréhension, la préparation et la modification.

2.6.1.1 Les principaux artefacts

Artefact : Modèle d'analyse

Le modèle d'analyse est représenté par un ou plusieurs paquetages d'analyse de niveau supérieur du modèle. L'utilisation des **paquetages d'analyse** permet d'aménager le modèle en parties plus gérables, représentant des abstractions de sous-système, voire des **couches** complètes de la conception du système. C'est une vue interne du système formulé dans le langage du développeur [10]. Dans le cas où il existe un seul paquetage d'analyse, c'est le paquetage du système lui-même. Chaque paquetage d'analyse est composé de façon récursive par des paquetages d'analyse, des **classes d'analyse** qui constituent des abstractions de classes et éventuellement, des sous-systèmes de la conception, et de **réalisation d'analyse des cas d'utilisation** qui représente la réalisation des cas d'utilisation par des classes d'analyse et les objets qu'elles comportent [10] (*figure 2.9*).

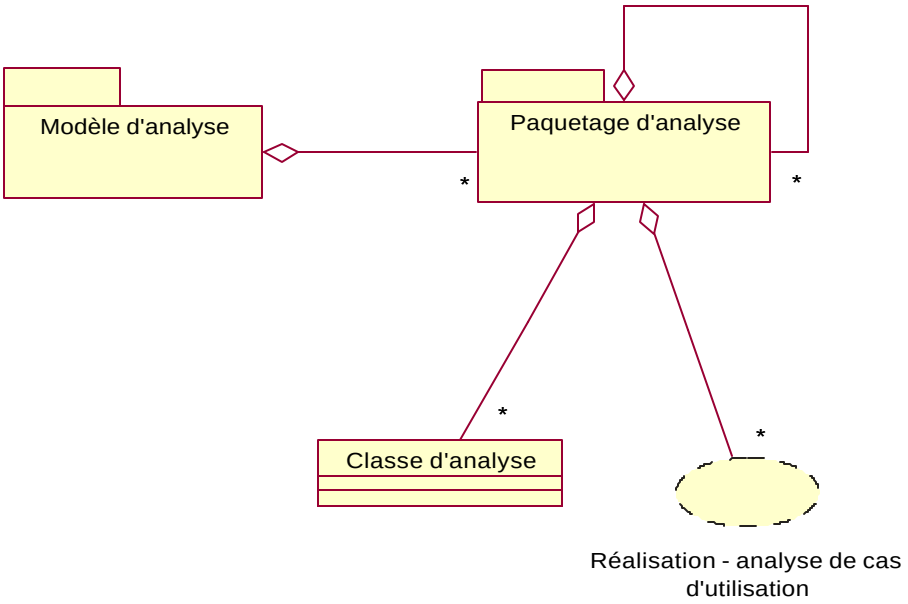

Figure 2.9 : Modèle d'analyse de RUP

Artefact : classe d'analyse

Une classe d'analyse possède les caractéristiques suivantes [10]:

- Se concentre sur les besoins fonctionnels et renvoie la prise en compte des exigences non fonctionnelles aux activités postérieures de conception et d'implémentation. Cette séparation rend les classes d'analyses plus évidentes dans le contexte du domaine du problème.
- Son comportement est défini par des responsabilités à un niveau plus élevé et moins formel.
- Elle définit des attributs de type conceptuel et reconnaissable à partir du domaine du problème, tandis que ceux des classes de conception et d'implémentation présentent souvent des types issus des langages de programmation.
- Elle est représenté par l'un des trois stéréotypes de base : « frontière », « contrôle » et « entité » (*figure 2.10*).

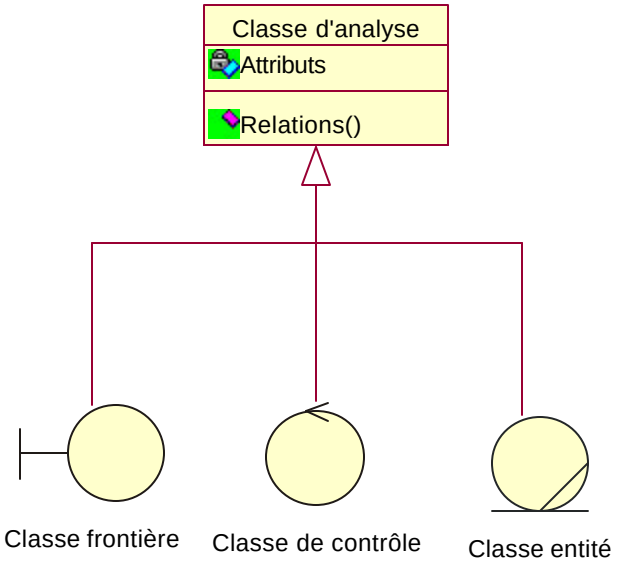

Figure 2.10 : artefact Classe d'analyse

Classe frontière : est utilisée pour modéliser la communication entre l'environnement du système et ses fonctionnements internes [9], Plus précisément, c'est l'interaction entre le système et ses acteurs. Cette interaction implique la réception et la présentation d'informations et des requêtes de la part et en direction des utilisateurs et des systèmes externes [10]. Les classes frontières modélisent les parties du système qui dépendent de ses acteurs, ce qui implique qu'elles clarifient et recueillent les exigences pesant sur les frontières du système [10].

Classe entité : sert à modéliser les informations de longue durée et souvent de nature persistante. Ce type de classe modélise les informations et le comportement associé d'un phénomène ou d'un concept tel qu'un individu, un objet ou un événement du monde réel [10]. Les classes entités fournissent une représentation des informations utile au développement, surtout pour la prise en charge de la persistance.

Classe de contrôle : sert à modéliser le comportement spécifique d'un ou plusieurs cas d'utilisation [9]. Elle représente la coordination, le séquencement, les transactions et le contrôle d'autres objets. Les classes de contrôle servent à encapsuler le contrôle associé à un cas d'utilisation spécifique. Elles permettent aussi de représenter des dérivations et des calculs complexes, tels que la logique métier, ne pouvant être liés à aucune information spécifique et durable stockée par le système [10].

Artefact : réalisation – analyse de cas d'utilisation

Une réalisation - analyse de cas d'utilisation est une collaboration au sein du modèle d'analyse décrivant la façon dont un cas d'utilisation donné est réalisé et exécuté en terme de classes d'analyse et d'interactions entre les objets qu'elles contiennent [10].

Il existe une traçabilité directe entre une réalisation - analyse de cas d'utilisation du modèle d'analyse et un cas d'utilisation du modèle des cas d'utilisation (*figure 2.11)*.

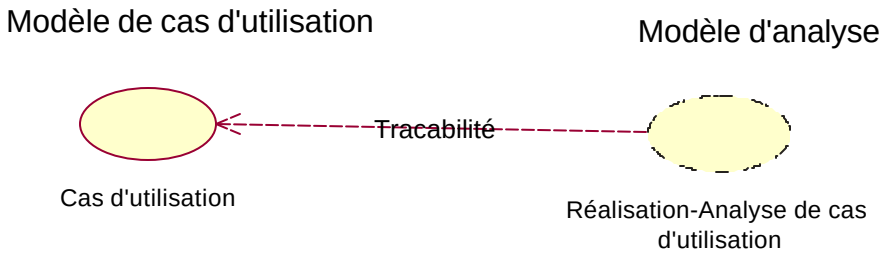

Figure 2.11 :Traçabilité directe entre une réalisation - analyse de cas d'utilisation du modèle d'analyse et un cas d'utilisation du modèle des cas d'utilisation [10]

Une réalisation de cas d'utilisation s'intéresse seulement aux exigences fonctionnelles, car elle est décrite seulement par des classes et des objets d'analyse [10].

Elle présente une description textuelle des **flots d'événements**, des **diagrammes de classes** qui décrivent ses classes d'analyse, et des **diagrammes d'interaction** illustrant la réalisation d'un flot ou d'un scénario particulier du cas d'utilisation en termes d'interactions entre les objets d'analyse [10].

Artefact : Diagramme de classes d'analyse:

Le diagramme de classes d'analyse renferme une collection des classes d'analyse et leurs relations. Une classe d'analyse participe à plusieurs réalisations de cas d'utilisation, Il est important de coordonner toutes les exigences que sont susceptibles d'avoir différentes réalisations de cas d'utilisation sur une classe et ses objets [10].

Artefact : Diagramme d'interaction:

La séquence d'actions d'un cas d'utilisation débute lorsqu'un acteur invoque ce cas d'utilisation en envoyant un message à un objet frontière du système. L'objet frontière envoie à son tour un message à un autre objet, de sorte que les objets concernés dialoguent pour réaliser le cas d'utilisation [10]. Les diagrammes de collaboration sont les mieux placés pour répondre à ce

besoin car, contrairement aux diagrammes de séquence (voir définitions), l'intérêt est d'identifier les exigences et les responsabilités des objets et non les séquences chronologiques des interactions [10].

Artefact : Paquetage d'analyse

Un paquetage d'analyse est composé de classes d'analyse, de réalisation de cas d'utilisation et d'autres paquetages d'analyse de façon récursive (*figure 2.12*).

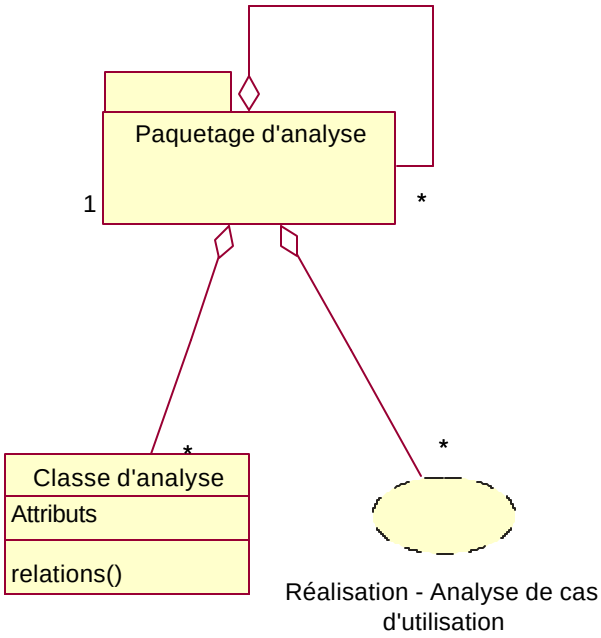

***Figure 2.12 : Paquetage d'analyse** [10]*

Les paquetages d'analyse offrent un moyen d'organiser les artefacts du modèle d'analyse en parties gérables présentant les caractéristiques suivantes [10] :

- Les paquetages d'analyses doivent provenir des besoins fonctionnels et du domaine du problème. Ils ne doivent pas s'appuyer sur les exigences non fonctionnelles, ni sur le domaine de la solution.
- Les paquetages d'analyse peuvent devenir ou seront repartis entre des sous-systèmes des deux couches supérieures d'applications du modèle de conception. Dans certains cas, un paquetage d'analyse pourra même refléter toute une couche de niveau supérieur du modèle de conception.

2.6.2 Conception

L'activité de conception consiste à façonner le système en lui donnant une forme et une architecture répondant à tous les besoins et les exigences : fonctionnelles, non fonctionnelles et toutes les autres contraintes formulées à son endroit [10].

Les principaux objectifs de cette activité sont [10] :

- Comprendre toutes les exigences non fonctionnelles et les contraintes liées aux langages de programmation, à la réutilisation des composants, aux systèmes d'exploitation, aux technologies de distribution, de concurrence, de bases de données, d'interfaces utilisateurs, de gestion de transactions, etc.
- Constituer une base aux activités d'implémentation en formulant les exigences pesant sur chaque sous-système individuel, sur les interfaces et sur les classes.
- Créer une abstraction transparente de l'implémentation du système.

2.6.2.1 Principaux artefacts

Artefact : Modèle de conception

Le modèle de conception (voir définition) est représenté par un ou plusieurs sous-systèmes lui offrant la possibilité d'être découpé en parties plus gérables, des classes de conception, de réalisation -conception de cas d'utilisation et des interfaces (*figure 2.13*). Dans le cas où il existe un seul sous-système, c'est le système lui-même appelé sous-système de haut niveau.

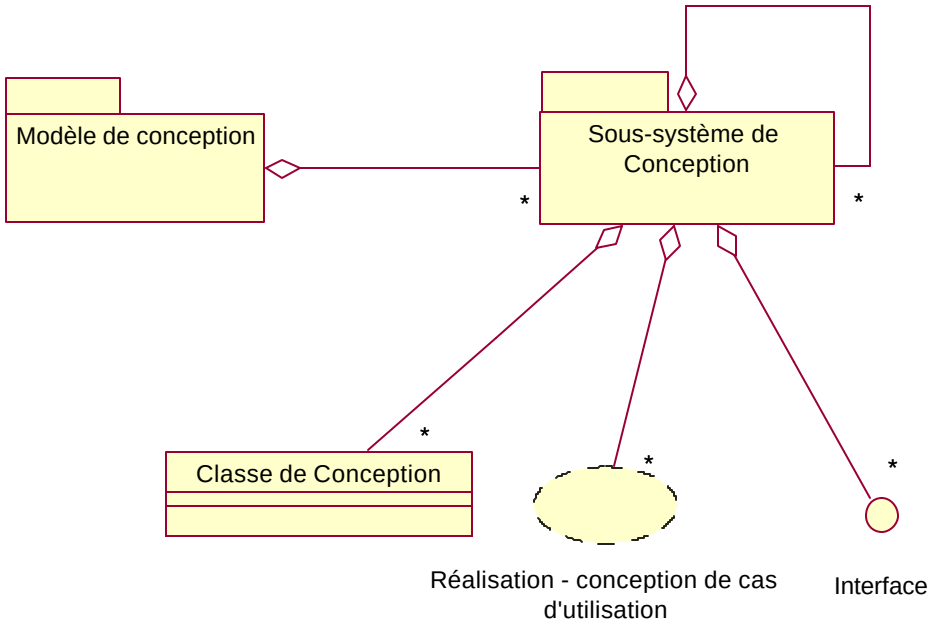

***Figure 2.13 : Modèle de conception** [10]*

Artefact : classe de conception

Une classe de conception est une abstraction transparente d'une classe ou d'une construction équivalente de l'implémentation du système, car [10]:

- Sa spécification (opérations, attributs, paramètres, types, …) doit utiliser le même langage que le langage de programmation.
- Ses relations avec d'autres classes doivent trouver une signification directe lors de son implémentation.
- Elle est stéréotypée selon le langage de programmation.

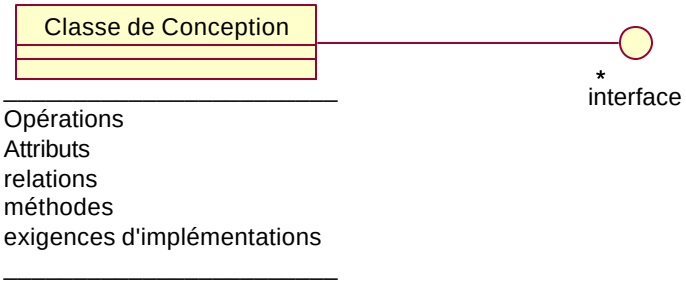

***Figure 2.14 : Classe de conception** [10]*

Artefact : réalisation – conception de cas d'utilisation

Une réalisation - conception de cas d'utilisation décrit la réalisation et l'exécution d'un cas d'utilisation par des classes de conception et les objets qui les composent. Elle offre une traçabilité directe avec une réalisation – analyse de cas d'utilisation du modèle d'analyse [10]. Elle comporte une description textuelle de flot d'événement, des diagrammes de classes de conception y participant, et des diagrammes d'interaction décrivant la réalisation d'un flot ou d'un scénario précis du cas d'utilisation en termes d'interactions entre les objets de conception (*figure 2.15*).

Une réalisation – conception de cas d'utilisation fournit une réalisation physique de la réalisation – analyse de cas d'utilisation avec laquelle elle est liée par une relation de traçabilité, et prend en compte la plupart des exigences non fonctionnelles et celles qui sont liées aux activités d'implémentation.

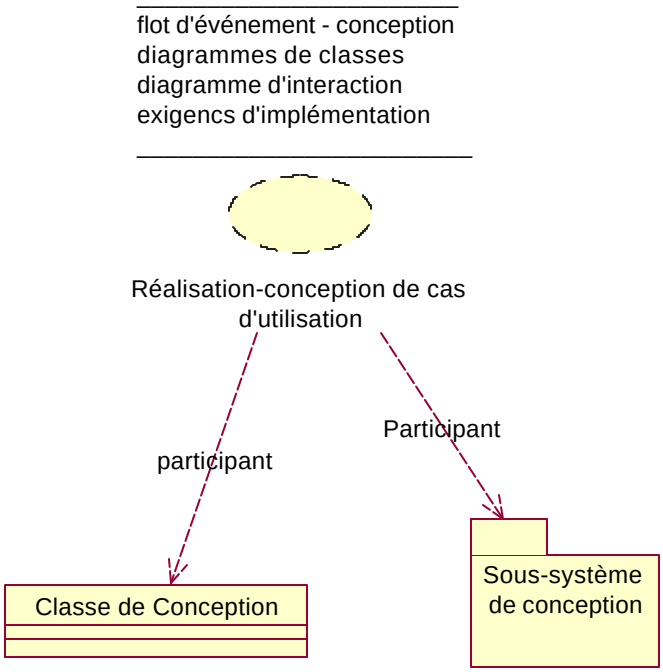

Figure 2.15 : Réalisation – conception de cas d'utilisation *[10]*

Artefact : Diagramme de classes

Les diagrammes de classes associées à une réalisation de cas d'utilisation montrent les classes et les sous-systèmes de conception qui y participent, et leurs relations [10].

Artefact : Diagramme d'interaction

La séquence d'actions d'un cas d'utilisation débute lorsqu'un acteur invoque celui-ci en adressant un message à un objet de conception du système. Cet objet appellera un autre objet, afin que les objets de conception impliqués dialoguent entre eux et exécutent le cas d'utilisation. En conception, cette interaction est tracée à l'aide du diagramme de séquence, puisqu'on veut détecter les séquences d'interaction détaillées et chronologiques [10].

Artefact : Sous-système de conception

Un sous-système de conception se compose de classes de conception, de réalisations de cas d'utilisation, d'interfaces et d'autres sous-systèmes de façon récursive ; et il peut fournir aussi des interfaces représentant les fonctions qu'exportent celles-ci en termes d'opérations [10].

Les deux couches supérieures de l'application et leurs sous-systèmes dans le modèle de conception ont souvent une traçabilité directe avec des paquetages d'analyse et/ou des classes d'analyse [10].

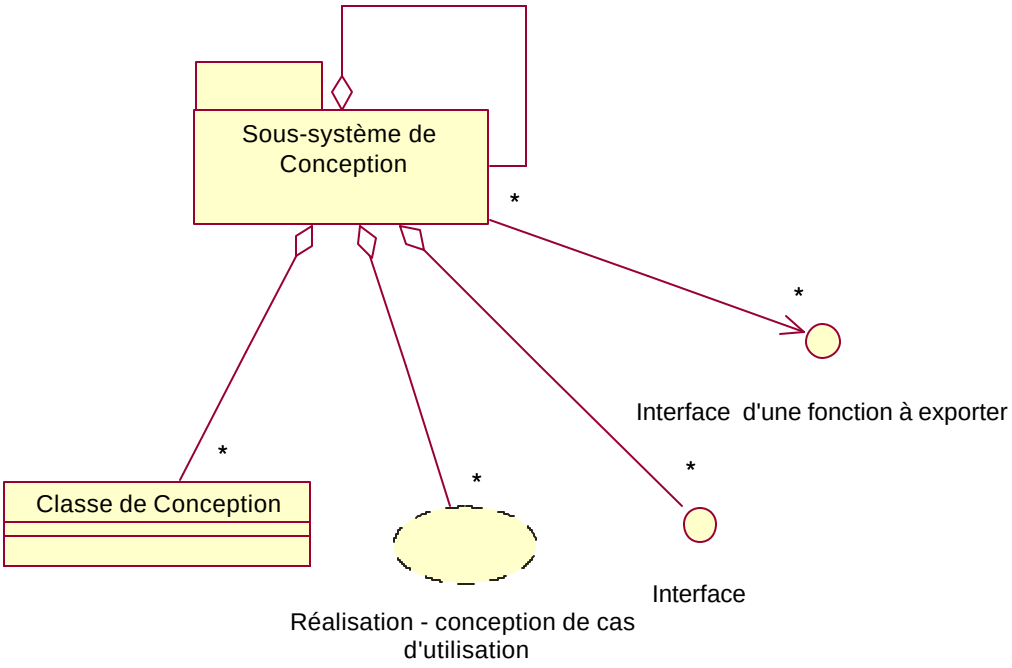

Figure 2.16 : Sous-système de conception [10]

2.6.3 Comparaison entre le modèle d'analyse et le modèle de conception

Modèle d' analyse	Modèle de conception
Modèle conceptuel, car c'est une abstraction du système qui évite les questions d'implémentation.	Modèle physique, puisque c'est un plan d'élaboration et de construction de l'implémentation
Générique à la conception	Spécifique à une implémentation
Peu de couches	Nombreuses couches
S'attache peu à la séquence	S'intéresse beaucoup à la séquence
Risque de ne pas être conservé tout au long de cycle de vie de logiciel	Doit être conservé tout au long de cycle de vie de logiciel
Définit une structure constituant une base essentielle pour la création du système.	Façonne le système tout en s'efforçant de préserver autant que possible la structure définie par le modèle d'analyse.

2.7 Les couches dans RUP

Le développement orienté objet offre une possibilité d'architecturer les applications selon un modèle à plusieurs couches. Chaque couche offre un ou plusieurs services logiciels, à travers des packages à une autre couche de plus haut niveau. Chaque package encapsule un ensemble de traitements et offre son service à travers une interface, de telle façon qu'une couche de niveau supérieur utilise ce service sans savoir comment il est implémenté [10].

2.7.1 Le concept de multicouches

Le concept de multicouches dans RUP représente un groupage ordonné de fonctionnalités. Les fonctions spécifiques à l'application sont localisées dans les couches supérieures, les fonctionnalités du domaine d'application sont dans les couches du milieu et les fonctionnalités spécifiques à l'environnement de développement sont dans les couches les plus basses [9].
Le nombre et la décomposition en couches dépendent de la complexité du domaine du problème et de la solution proposée [9] :

- généralement, il existe une seule couche spécifique à l'application
- les domaines dans lesquels les systèmes prévus sont construits ou dans lesquelles les grands systèmes sont décomposés en sous-systèmes interopérables pour un grand besoin de partager l'information entre les équipes de développement. Comme résultat, la couche spécifique au domaine d'affaire doit partiellement exister ou elle peut être structurée en plusieurs couches pour plus de clarté.
- La solution qui est bien supportée par les produits du middleware, dans lequel le logiciel du système complexe joue un grand rôle, doit avoir des couches basses bien développées, avec peut être plusieurs couches de middleware et du logiciel système.

Tous les sous-systèmes peuvent être organisés en couches. Les sous-systèmes, spécifiques à l'application, trouvés dans les couches supérieures de l'architecture, les sous-systèmes spécifiques aux matériels et au système d'opération dans les couches basses de l'architecture, et les services dit universels pour des besoins généraux occupent les couches de middleware [9].
On distingue généralement quatre couches décrites ci-dessous et ordonnées de haut en bas [9], [10] :

Couche spécifique à l'application : partie d'un système (composée de paquetages ou de sous-systèmes) offre des services spécifiques à l'application et non partagés par d'autres parties (sous-systèmes). Cette couche utilise la couche spécifique au domaine d'affaire.

Couche spécifique au domaine d'affaire : partie d'un système (composée de paquetages ou de sous-systèmes) réutilisable dans le cadre d'un métier ou d'un domaine et offrant des composants spécifique au domaine d'affaire. Cette couche est utilisée par la couche spécifique à l'application.

Couche de middleware : couche fournissant des briques de bases réutilisables (paquetages ou sous-systèmes) pour des infrastructures préfabriquées d'utilitaires et des services indépendants de toute plate-forme, par exemple pour le calcul et l'interopérabilité des objets distribués dans des environnements hétérogènes. Il s'agit, entre autres, des ORB, des infrastructures préfabriquées indépendantes de toute plate-forme, permettant la création d'interface utilisateur graphique ou, en général, des produits mettant des mécanismes génériques de conception.

Couche de logiciel système : couche contenant les logiciels de calcul et de mise en réseau de l'infrastructure, comme les systèmes d'opération, les systèmes de gestion de bases de données, les interfaces avec des matériels spécifiques, etc. c'est la couche située au bas de la hiérarchie des couches.

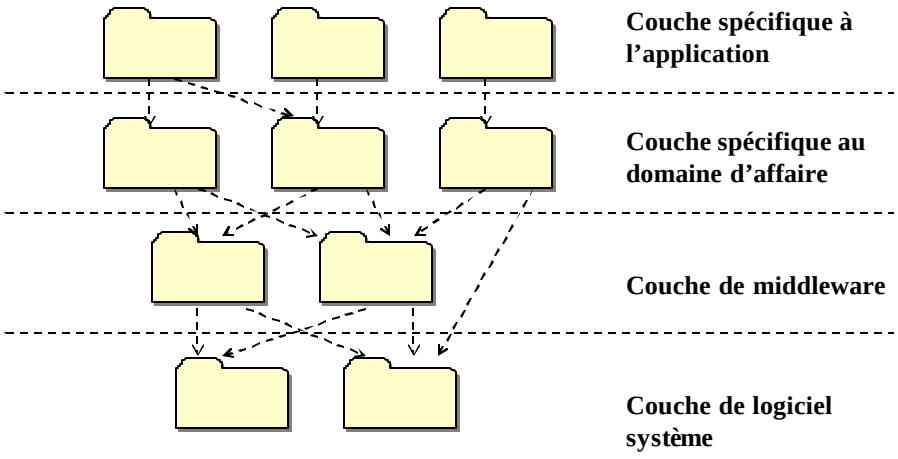

***Figure 2.17 : l'architecture de 4 couches en RUP** [10]*

L'architecture en couche est d'une importance déterminante, car il simplifie la compréhension et l'organisation du développement de systèmes complexes. Il réduit les dépendances en faisant en sorte que les couches inférieures ignorent les détails ou les interfaces des couches supérieures. Il facilite aussi l'identification de ce qui peut être réutilisé, et fournit une structure d'aide à la prise de décision sur ce qui doit être acheté ou construit [10].

Les systèmes à architecture en couches disposent, en leur sommet, des sous-systèmes d'applications individuels, élaborés à partir de sous-systèmes des couches inférieures, tels que des

frameworks ou des bibliothèques de classes. La couche générale aux applications contient des sous-systèmes qui ne sont pas spécifiques à une seule application, mais peuvent être réutilisés pour diverses applications du même domaine ou du même métier. L'architecture des deux couches inférieures peut être élaborée sans prendre en compte les détails des cas d'utilisation, car ces couches ne sont pas spécifiques au métier, alors que l'architecture des deux couches supérieures est, quant à elle, créée à partir des cas d'utilisation pertinents sur le plan architectural (spécifiques au métier) [10].

Une couche est un ensemble de sous-systèmes partageant le même degré de généralité et de volatilité des interfaces : les couches inférieures sont générales à plusieurs applications et doivent bénéficier d'interfaces plus stables, contrairement aux couches supérieures, plus spécifiques aux applications et pouvant, par conséquent, disposer d'interfaces plus mouvantes. Parce qu'elles ont des interfaces moins changeantes, les couches inférieures peuvent servir de base à l'élaboration des couches supérieures. Les sous-systèmes des différentes couches peuvent réutiliser des cas d'utilisation, d'autres sous-systèmes de niveau inférieur, des classes, interfaces, collaborations et composants issus de couches inférieures [10].

2.7.2 Les règles de groupage des composants (sous-systèmes ou paquetages) dans des couches.

Le concept de couches fournit une partition logique des composants dans des ensembles, en définissant certaines règles qui doivent être formées entre les couches. Il permet de réduire la dépendance entre les composants pour que le système devienne faiblement couplé et facilement maintenable. Le groupage des composants suit les règles suivantes [9] :

La visibilité : les composants doivent dépendre seulement des composants d'une même couche ou de la couche suivante du niveau le plus bas.

La volatilité :

- Dans les couches les plus hautes, mettre les éléments qui varient quand les exigences utilisateurs changent.
- Dans les couches les plus basses, mettre les éléments qui varient quand la plateforme d'implémentation change (matériel, langage, système d'opération, base de données, etc.)
- Intercalé au milieu, mettre les éléments applicables à une large gamme de systèmes et les environnements d'implémentation.
- Ajouter des couches quand des partitions additionnelles dans ces larges catégories aident à organiser le modèle.

Généralité : les éléments du modèle abstrait tendent à être placés plus bas dans le modèle. S'ils ne sont pas spécifiques à l'implémentation, ils tendent à graviter dans les couches du milieu.

Nombre de couches : pour un petit système, trois couches sont suffisantes. Pour un système complexe, 5-7 couches sont d'habitude suffisantes.

- 0 à 10 classes : il n'existe pas un besoin de couches
- 10 à 50 classes : 2 couches
- 25 à 150 classes : 3 couches
- 100 à 1000 classes : 4 couches

2.7.3 Utilisation des couches dans le modèle d'analyse : identification des paquetages d'analyse

L'objectif est la mise au point des paquetages relativement indépendants et faiblement couplés, mais ayant une forte cohésion interne. Il est donc recommandé de tenter de réduire le nombre de relations entre classes de différents paquetages afin de limiter les dépendances entre les paquetages.

Pour clarifier les dépendances, il peut être utile de découper le modèle d'analyse en couches, en plaçant les paquetages spécifiques à l'application dans une couche de haut niveau, et les paquetages spécifique au domaine d'affaire dans une couche inférieure. La distinction entre les fonctionnalités spécifiques et les fonctionnalités générales devient plus évidente [10].

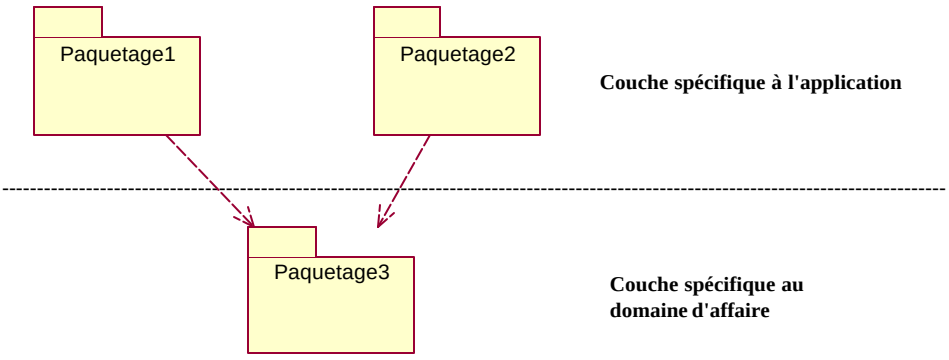

Figure 2.18 : affectation des paquetages d'analyse dans des couches

Les premiers rôles identifiés dans l'activité d'analyse de cas d'utilisation sont d'exprimer le comportement des couches les plus hautes du système – les comportements spécifiques à l'application et au domaine d'affaire. Les classes frontières et de contrôle évoluent typiquement dans le design des éléments de la couche application, pendant que les classes entités évoluent dans le design de la couche spécifique du domaine d'affaire [9].

2.7.4 Utilisation des couches dans le modèle de conception : identification des sous-système d'applications

On identifie les sous-systèmes de la couche spécifique à l'application et ceux de la couche spécifique au domaine d'affaire (les deux couches supérieures). Si l'analyse autorise une décomposition satisfaisante en paquetages d'analyse, on utilisera ces derniers autant que possible et on identifiera les sous-systèmes correspondants dans le modèle de conception [10].

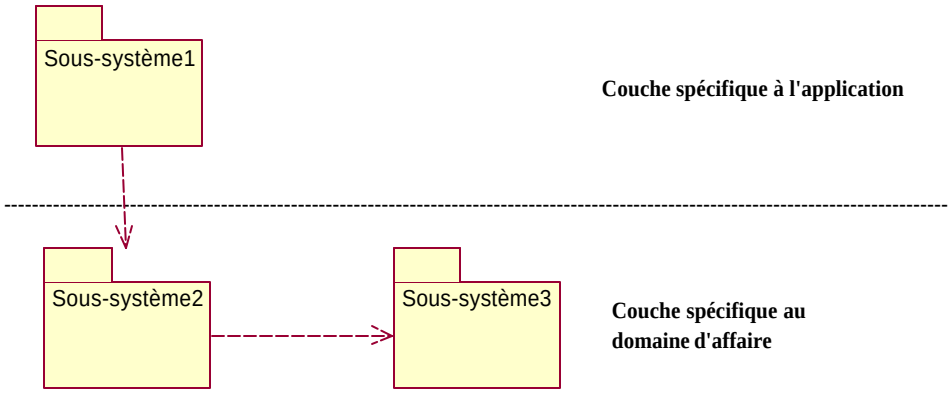

Figure 2.19 : affectation des sous-systèmes d'applications dans des couches

La conception permet d'identifier des sous-systèmes, qui fourniront un service général susceptible d'être employé par plusieurs réalisations de cas d'utilisation [10].

2.7.5 Utilisation des couches dans le modèle de déploiement : identification des sous-système de middleware et de logiciel système.

Le middleware et le logiciel système constituent le fondement d'un système, car toutes les fonctions s'appuient sur des logiciels tels que les systèmes d'exploitation, les systèmes de gestion de bases de données, les logiciels de communication, les technologies de distribution des objets et les technologies de gestion de transactions.

Les middleware Java, par exemple, fournissent des interfaces utilisateur graphiques indépendantes de toute plate-forme (java.awt) et assurent la mise en œuvre de la distribution des objets (java.rmi).

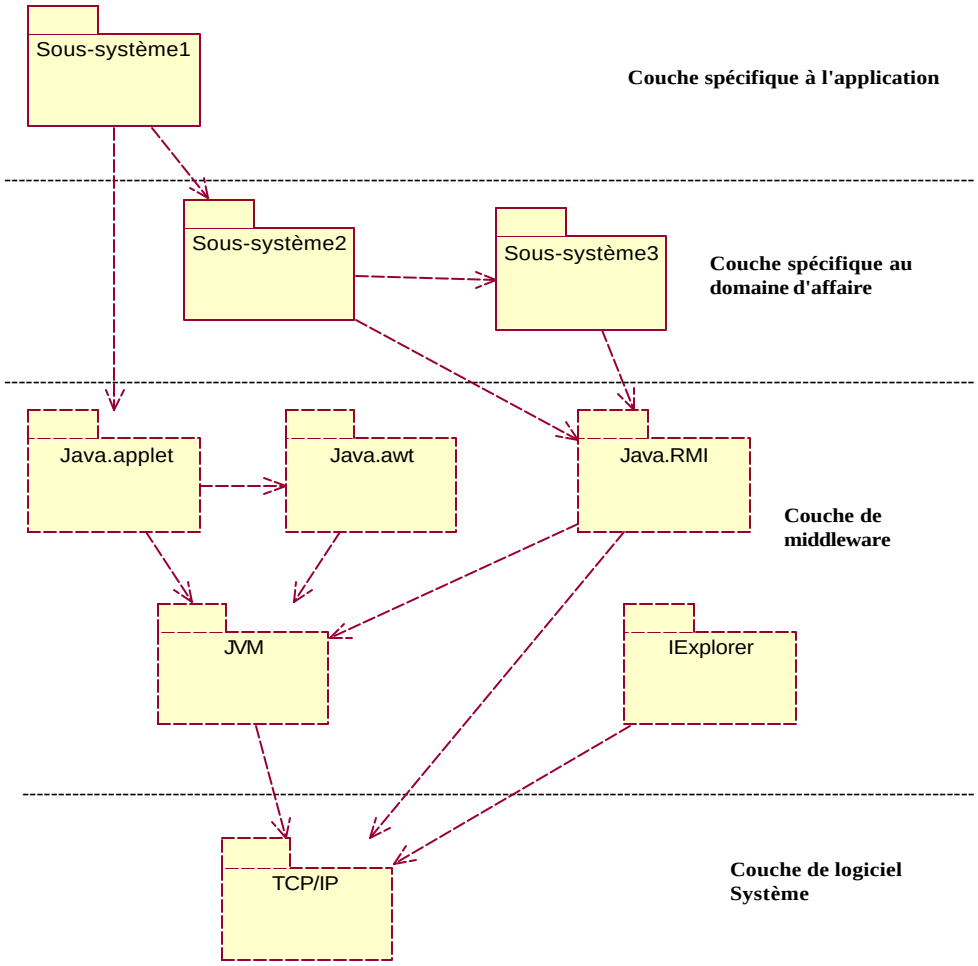

Figure 2.20 : Répartition des sous-systèmes dans des couches dans le modèle de déploiement

CHAPITRE 3 : RAPPROCHEMENT COSMIC-FFP ET RUP

Supposons que nous sommes dans une itération de la phase de création ou d'élaboration du processus RUP : on aurait certainement exécuté les activités des disciplines « expressions des besoins » et « analyse et conception » tout en générant les artefacts correspondant à chaque activité.

La procédure de rapprochement consiste à exécuter la phase de mappage (*figure 1.2*) et la phase de mesure (*figure 1.3*) du processus de mesure de COSMIC-FFP (*figure 1.1*) tout en appliquant les règles et les principes de COSMIC-FFP définis dans le chapitre 1 sur les artefacts RUP générés dans ces activités et décrites dans le chapitre 2. L'objectif est d'identifier les différents éléments du modèle COSMIF-FFP, tout en respectant ses principes.

1 Discipline : Expression des besoins

Les principaux artefacts générés dans les activités de cette discipline sont : les acteurs, les cas d'utilisation et le modèle de cas d'utilisation qui est composé par un ou plusieurs diagrammes de cas d'utilisation, décrivant la relation entre les acteurs et les cas d'utilisation (voir chapitre 2).

1.1 Processus de mappage COSMIC-FFP

1.1.1 Identification des couches

	Rapprochement	Élément COSMIC-FFP	Artefact RUP	Références
Couches	Le modèle de cas d'utilisation ne permet pas d'identifier plusieurs couches COSMIC-FFP. L'interaction entre les acteurs (environnement utilisateurs) et les cas d'utilisation (le système) permet d'identifier une seule couche logicielle du système entier.	une seule couche logicielle du système entier est identifiée.	Modèle de cas d'utilisation.	❑ le principe a) d'identification des couches de COSMIC-FFP ❑ les définitions acteur, cas d'utilisation, diagramme de cas d'utilisation de UML ❑ la définition du modèle de cas d'utilisation de RUP.

1.1.2 Identification des frontières

	Rapprochement	Eléments COSMIC-FFP	Artefact RUP	Références
Frontières	Si on considère que le système représenté par les cas d'utilisation est une pièce de logiciel, donc l'environnement de système où il opère, qui est représenté par les acteurs, est le même environnement pour la pièce de logiciel. C'est-à-dire que la frontière, selon COSMIC-FFP, est celle qui sépare les acteurs et les cas d'utilisation dans les diagrammes de cas d'utilisation qui forment le modèle de cas d'utilisation de RUP. En plus, l'identification d'une seule couche dans la section précédente permet d'identifier une seule frontière, celle qui sépare la couche (système entier) et l'environnement des acteurs.	Une seule frontière COSMIC-FFP est identifiée qui sépare la couche logicielle du système entier et l'environnement des utilisateurs.	Modèle de cas d'utilisation	- Définition des frontières de COSMIC-FFP - définition diagramme de cas d'utilisation de UML - définition du modèle de cas d'utilisation de RUP
Utilisateurs	Les acteurs du modèle de cas d'utilisation qui interagissent avec les cas d'utilisation du système sont des Utilisateurs dans COSMIC-FFP qui utilisent la pièce de logiciel (le système représenté par les cas d'utilisation).	Tous les utilisateurs de l'unique couche logicielle sont identifiés.	- Acteur - Modèle de cas d'utilisation	- définition Utilisateur de COSMIC-FFP - définition Acteur en UML - définition Modèle de cas d'utilisation de RUP

Identification des processus fonctionnels

	Rapprochement	Élément COSMIC-FFP	Artefact RUP	Références
Processus fonctionnels	Les cas d'utilisation sont déduits directement à partir des exigences fonctionnelles utilisateurs et ils définissent le comportement du système par des séquences d'actions pour produire un résultat observable aux utilisateurs. Dans le modèle de cas d'utilisation, les valeurs sont produites par les cas d'utilisations lorsqu'ils interagissent avec les acteurs. Ce qui implique que dans ce modèle un cas d'utilisation correspond à un processus fonctionnel et que les valeurs peuvent être produites suites aux mouvements des données.	les Processus fonctionnels de la couche logicielle du système entier sont identifiés.	- cas d'utilisation - modèle de cas d'utilisation	- définition et principes des processus fonctionnel de COSMIC-FFP - définition UML de cas d'utilisation - définition RUP du modèle de cas d'utilisation
Événements	Les séquences d'actions d'un cas d'utilisation dans le modèle de cas d'utilisation sont déclenchées par un acteur qui se trouve en dehors de la frontière de la couche logicielle (voir identification des frontières). Cette interaction (acteur – cas d'utilisation) dans le modèle de cas d'utilisation est appelée événement externe dans RUP (voir définition) et considérée un événement dans COSMIC-FFP lorsque sa source est l'acteur et sa destination est le cas d'utilisation (sens : acteur – cas d'utilisation). Un événement est matérialisé par la relation entre un acteur et un cas d'utilisation dans le diagramme de cas d'utilisation.	Les événements déclencheurs des processus fonctionnels de la couche logicielle du système entier sont identifiés.	- cas d'utilisation - acteur - modèle de cas d'utilisation.	- définition d'un événement déclencheur en COSMIC-FFP - définition UML de cas d'utilisation - définition RUP du modèle de cas d'utilisation et événement.

1.1.3 Identification des attributs de données

Les classes frontières de l'artefact story-board des cas d'utilisation servent à concevoir un prototype d'interface utilisateur dans cette première discipline du projet. Ces interfaces permettent de valider avec les utilisateurs, à cette étape de développement, les informations utiles pour prendre en charge ses exigences fonctionnelles. On essaie de collecter le maximum d'informations qui sont directement liées à ces exigences et de les présenter sous forme, par exemple, de prototypes de fenêtres ou de rapports aux utilisateurs pour les valider. On distingue deux types d'information dans ces prototypes d'interface utilisateur :

- Information liée directement aux besoins fonctionnels et représente le monde réel, et qui est considérée un attribut de données, selon COSMIC-FFP.
- Information qui est calculée à partir des informations du premier type selon des règles de gestion liées aux besoins fonctionnels, et qui ne sont pas des attributs des données.

Par conséquent, la majorité des attributs des données du système sont identifiés à ce niveau et qui résident dans les classes frontières.

1.1.4 Identification des groupes de données

On distingue deux types de classe frontière dans l'artefact story-board de cas d'utilisation :

- Une classe frontière qui contient seulement le premier type d'information identifié dans la section précédente (attribut de données) et qui apparaît dans le diagramme d'interaction en relation directe avec les acteurs (le message est dans le sens acteur – classe frontière). Cette classe contient seulement les attributs de données introduits par les utilisateurs au système.
- Une classe frontière qui contient les deux types d'information (attributs et non attributs des données). Cette classe n'a pas de relation avec les acteurs et elle est invoquée par les classes du premier type pour afficher ou imprimer un résultat à l'utilisateur.

Par conséquent, la première classe est considérée comme un groupe de données car :

- Elle est persistante (persistance courte).
- Elle contient des attributs de données distincts, non vides, non ordonnés et non redondants.
- Elle respecte les 3 principes des groupes de données.

1.2 Processus de mesure de COSMIC-FFP

1.2.1 Identification des sous-processus

Chaque cas d'utilisation identifié dans l'artefact du modèle de cas d'utilisation sera réalisé par les classes frontières dans l'artefact story-board de cas d'utilisation. Si la couche logicielle du système entier identifié dans la section 1.1 est représentée par les cas d'utilisation dans le modèle de cas d'utilisation, elle sera représentée par les classes frontières dans l'artefact story-board de cas d'utilisation, et les interactions des classes frontières de tous les cas d'utilisation dans les diagrammes d'interaction représentent (dans la discipline « expression des besoins ») le système entier.

L'artefact de story-board nous permet d'identifier, à travers ses diagrammes d'interaction (collaboration ou séquence), deux types de sous-processus :

- Un sous-processus Entrée (E) qui déplace les attributs de données du groupe de données (classe frontière du premier type) du côté de l'utilisateur de la frontière logicielle à l'intérieur de la frontière de logicielle. il est matérialisé par un message de l'acteur vers la classe frontière groupe de données.
- Un sous-processus Exit (X) qui déplace les attributs de données de la classe frontière groupe de données de l'intérieur de la frontière logicielle à l'utilisateur. Il est matérialisé par un message de la classe frontière groupe de données vers une autre classe frontière. Cette dernière, si elle est invoquée à travers un message, c'est pour la raison de faire sortir (afficher, imprimer,…) des informations à l'utilisateur, car c'est une classe frontière. Dans un diagramme d'interaction de l'artefact story-board de cas d'utilisation on ne trouve pas des messages direction classe frontière – acteur car, si on veut sortir des informations à l'utilisateur, on invoque tout simplement une classe frontière.

1.2.2 Application de la fonction de mesure

- Identifier les cas d'utilisation (un cas d'utilisation est un processus fonctionnel),
- Identifier tous les diagrammes d'interaction (collaboration ou séquence, et non pas les deux à la fois) de l'artefact story-board correspondant à chaque cas d'utilisation identifié précédemment (un cas d'utilisation est réalisé par un ou plusieurs diagrammes d'interaction).
- Pour chaque diagramme d'interaction, les messages direction Acteur – Classe frontière sont des sous-processus de type Entrée (E), et les messages direction Classe frontière – Classe frontière sont de type Exit (X). Chacun d'eux correspond à 1 Cfsu.

1.2.3 Agrégation du résultat de mesure

Comme il existe une seule couche pour le système, on a

$$\text{Taille Partielle}_{\text{Cfsu}}(\text{couche}) = \Sigma\ \text{taille(entrées)} + \Sigma\ \text{taille (exits)}$$

2 Discipline : Analyse et Conception

Dans RUP la traçabilité entre les artefacts des différentes activités des disciplines est garantie par les cas d'utilisation. Dans cette deuxième discipline, on détaille les cas d'utilisation décrits dans les activités de l'expression des besoins, en ajoutant d'autres artefacts. Cette discipline est composée en deux grandes sous-disciplines : analyse et conception.

2.1 Analyse

Le modèle d'analyse est composé par un ou plusieurs paquetages d'analyse, chaque paquetage d'analyse est composé par un ou plusieurs paquetages d'analyse (de façon récursive), des classes d'analyse (frontière, contrôle et entité) et des réalisations - analyse de cas d'utilisation. Les paquetages d'analyse sont distribués dans les deux couches supérieures :

- La couche spécifique à l'application qui contient généralement les paquetages des classes frontières et la réalisation – analyse des cas d'utilisation avec ces classes.
- La couche spécifique au domaine d'affaire qui contient généralement les paquetages des classes entités, des classes de contrôle et la réalisation – analyse des cas d'utilisation avec ces classes.

La réalisation – analyse cas d'utilisation contient un diagramme de classes d'analyse (frontière ou contrôle et entité) qui collaborent pour réaliser le cas d'utilisation et un ou plusieurs diagrammes d'interaction (collaboration ou séquence) entre les classes d'analyses et les acteurs.

2.1.1 Processus de mappage de COSMIC-FFP

Comme dans RUP, notre processus de mappage détaille ce qui était rapproché dans la discipline précédente, en se basant sur les cas d'utilisation qui sont considérés maintenant des processus fonctionnels.

2.1.1.1 Identification des couches

Une couche dans COSMIC-FFP est composée par des processus fonctionnels. Les pièces de logiciel dans les couches échangent des données à travers ces processus fonctionnels.

Si on considère qu'un processus fonctionnel est un cas d'utilisation, on résume qu'une couche COSMIC-FFP est composée par des cas d'utilisation. C'est-à-dire qu'on distribue nos cas d'utilisation dans des couches, ce qui n'est pas le cas pour RUP. RUP considère, dans le modèle d'analyse, qu'un cas d'utilisation se réalise, à l'aide des classes d'analyses, dans plusieurs paquetages inclus dans des couches. C'est-à-dire les cas d'utilisation ne sont pas distribués dans des couches, mais leurs réalisations s'effectuent dans des couches.

À part cette nuance, les concepts des couches dans COSMIC-FFP sont les mêmes que dans RUP (voir définitions, principes et concepts des couches dans RUP et COSMIC-FFP).

Ce problème peut être résolu de deux façons différentes :

1. En se basant sur la règle b) d'identification des couches de COSMIC-FFP, qui stipule que si un logiciel est conçu en utilisant un paradigme architectural connu, on utilise ce paradigme pour identifier les couches. Dans ce cas, on a qu'à utiliser le paradigme architectural de RUP.

2. On suppose qu'un processus fonctionnel n'est pas un cas d'utilisation (comme il est décrit dans la discipline précédente), mais c'est **une réalisation d'un cas d'utilisation dans une couche** (et non dans un paquetage d'une couche). Comme la réalisation des cas d'utilisation s'effectue dans plusieurs couches (dans des packages), donc les processus fonctionnels sont distribués dans plusieurs couches; et dans ce cas tous les principes COSMIC-FFP du processus fonctionnel seront respectés.

La deuxième solution sera adoptée dans ce travail, donc les couches RUP sont les mêmes que COSMIC-FFP.

	Rapprochement	**Élément COSMIC-FFP**	**Artefact RUP**	**Références**
Couches	❑ Selon la solution décrite précédemment une couche RUP correspond bien à une couche COSMIC-FFP, donc toutes les couches identifiées dans le modèle d'analyse sont des couches COSMIC-FFP ❑ Généralement, on distingue deux couches : la couche spécifique à l'application et la couche spécifique au domaine d'affaire. ❑ En absence de couches dans le modèle d'analyse de RUP, la couche logicielle du système entier identifiée dans l'expression des besoins les remplace.	Un, Deux ou plusieurs couches logicielles sont identifiées. Dans le cas d'une seule couche c'est celle du système entier.	Modèle d'analyse et modèle de cas d'utilisation	❑ Définition et principes d'identification des couches de COSMIC-FFP ❑ La définition, les concepts et les principes des couches dans RUP.

2.1.1.2 Identification des frontières

	Rapprochement	Élément COSMIC-FFP	Artefact RUP	Références
frontières	Les frontières sont identifiées selon le nombre de couches identifiées précédemment : ❑ une couche logicielle du système entier : une seule frontière, celle qui sépare les acteurs et le modèle de cas d'utilisation identifié dans l'expression des besoins. ❑ Deux ou plusieurs couches logicielles : la première frontière est celle qui sépare l'environnement des utilisateurs et la première couche. La deuxième frontière est entre la première couche et la deuxième couche. Le même raisonnement est adopté pour les couches suivantes.	Une, deux ou plusieurs frontières sont identifiées	Modèle d'analyse	principe a) des frontières dans COSMIC-FFP.
Utilisateurs	Deux types d'utilisateurs selon le nombre de couches identifiées: ❑ S'il existe une seule couche, les utilisateurs de la pièce de logiciel dans la première couche ou du système entier sont les acteurs de tout le système identifiés dans l'expression des besoins. ❑ utilisateurs de la pièce logicielle dans la deuxième couche sont les paquetages ou les classes (en absence de paquetage) de la première couche qui interagissent avec la pièce logicielle de la deuxième couche. Le même raisonnement est adopté pour les autres couches.	Tous les utilisateurs de toutes les couches sont identifiés	Modèle de cas d'utilisation Les diagrammes d'interaction dans réalisation - analyse cas d'utilisation dans les paquetages d'analyse.	- définition Utilisateur de COSMIC-FFP - définition Acteur en UML - définition diagrammes d'interaction dans UML.

2.1.1.3 Identification des processus fonctionnels

	Rapprochement	Élément COSMIC-FFP	Artefact RUP	Références
Processus fonctionnels	Il existe une traçabilité directe (relation 1:1) entre un cas d'utilisation du modèle de cas d'utilisation et une réalisation – analyse de cas d'utilisation du modèle d'analyse (*figure 2.11*). en plus, Une réalisation - analyse de cas d'utilisation, par définition, est une collaboration au sein du modèle d'analyse décrivant la façon dont un cas d'utilisation donné est réalisé et exécuté en terme de classes d'analyse et d'interactions entre les objets qu'elles contiennent [10]. Donc, si un cas d'utilisation est un processus fonctionnel, sa réalisation est aussi un processus fonctionnel dans un environnement d'exécution et de mouvement des données avec des classes d'analyse qui correspondent à des groupes de données.	Tous les processus fonctionnels sont identifiés.	Réalisation – analyse cas d'utilisation.	- définition processus fonctionnel dans COSMIC-FFP - définition réalisation – analyse cas d'utilisation dans RUP.
Événements	Deux types d'événements selon le nombre de couches identifiées: ❑ Événements se produisent dans l'environnement des utilisateurs en dehors de la frontière de la première couche (cas de plusieurs couches) ou du système entier (cas d'une seule couche), et initient respectivement les processus fonctionnels de la première couche ou du système entier. ❑ Événements se produisent dans la première couche (cas de plusieurs couches) et initient les processus fonctionnels dans la deuxième couche. Le même raisonnement est adopté pour les autres couches. Un événement RUP est matérialisé par l'interaction Acteur – Cas d'utilisation (voir identification événement dans expression de besoins), or les cas d'utilisation ici sont représentés par les classes d'analyses qui les réalisent. Donc, les événements sont matérialisés par l'interaction entre un acteur et une classe d'analyse dans les diagrammes de classes participantes à la réalisation - analyse cas d'utilisation dans les paquetages d'analyse.	Tous les événements qui déclenchent les processus fonctionnels dans toutes les couches sont identifiés.	Les diagrammes de classes participantes à la réalisation - analyse cas d'utilisation dans les paquetages d'analyse.	- définition événement de COSMIC-FFP - définition diagrammes d'interaction dans UML.

2.1.1.4 Identification des groupes de données

	Rapprochement	Élément COSMIC -FFP	Artefact RUP	Références
Groupe de données	❑ Une classe d'analyse se concentre sur les besoins fonctionnels et du contexte du domaine du problème. Son comportement est défini par des responsabilités à un niveau plus élevé et moins formel. elle définit des attributs de type conceptuel et reconnaissable à partir du domaine du problème. elle est représentée par l'un des trois stéréotypes de base : « frontière », « contrôle » et « entité ». ❑ Une classe entité sert à modéliser les informations de longue durée et de nature persistante. Elle modélise les informations et le comportement associé d'un phénomène ou d'un concept tel qu'un individu, un objet ou un événement du monde réel. Elle fournit une représentation des informations utiles au développement, surtout pour la prise en charge de la persistance. ❑ Une classe de contrôle sert à modéliser le comportement spécifique d'un ou de plusieurs cas d'utilisation, elle ne peut être lié à aucune information spécifique et durable stockée par le système. ❑ Une classe frontière modélise les parties du système qui dépendent de ses acteurs, ce qui implique qu'elles clarifient et recueillent les exigences pesant sur les frontières du système. Il est clair que selon ces définitions qu'une classe de contrôle ne peut pas être un groupe de données, et qu'une classe entité est un groupe de données. Deux types de classe frontière (voir identification de groupe de données dans l'expression des besoins) un type qui est en relation directe avec les utilisateurs et qui constitue un groupe de données car il contient des attributs de données, et le deuxième type n'est pas un groupe de données.	Tous les groupes de données du système sont identifiés.	Classe d'analyse du modèle d'analyse	Définition groupe de données de COSMIC-FFP Définition classe d'analyse de RUP.

2.1.1.5 Identification des attributs de données

	Rapprochement	Élément COSMIC-FFP	Artefact RUP	Références
attributs de données	Tous les attributs des classes d'analyse de type entité sont des attributs de données COSMIC-FFP, car ces classes sont des groupes de données.	Tous les attributs de données sont identifiés	Classe d'analyse du type entité du modèle d'analyse	Définition attributs de données de COSMIC-FFP Définition classe d'analyse de type entité de RUP.

2.1.2 Processus de mesure de COSMIC-FFP

2.1.2.1 Identification des sous-processus

Un processus fonctionnel est une réalisation – analyse cas d'utilisation. Un cas d'utilisation se réalise dans les paquetages d'analyse dans les différentes couches du modèle avec des classes d'analyse. Les diagrammes de collaboration ou de séquence (ou exclusif) dans les paquetages dans les couches de l'artefact modèle permet d'identifier les sous-processus fonctionnels.

L'événement se produit dans l'environnement utilisateur lorsque l'acteur interagit la première fois avec une classe frontière, le processus fonctionnel se déclenche en mouvementant des attributs de données ; c'est les sous-processus fonctionnels.

Les diagrammes d'interaction du modèle d'analyse ne contiennent pas des messages de direction Classe d'analyse – Acteur, car lorsqu'on veut sortir une information à l'utilisateur on invoque une classe frontière.

On rappelle que les classes de type entité sont des groupes de données et contiennent des attributs de données, et qu'il existe deux types de classes frontières : celles qui sont en relation directe avec l'utilisateur sont des groupes de données, et les autres ne sont pas des groupes de données, car elles contiennent des attributs de données redondants et des informations calculées à partir des attributs de données.

a) Les sous-processus fonctionnels de la première couche : la pièce de logiciel de la première couche interagit avec les acteurs du système identifié à partir du modèle de cas d'utilisation, dans le cas où on n'identifie pas des couches, la première couche est la couche du système entier identifié à partir du modèle de cas d'utilisation.

- Cas où il existe plus qu'une couche : Les cas d'utilisation de cette couche sont réalisés généralement par l'interaction des acteurs avec les classes frontières. Cette couche est appelée couche spécifique à l'application dans RUP ou couche de présentation dans d'autre processus. On identifie dans ce cas deux types de sous-processus :
  - Un sous-processus Entrée (E) qui déplace les attributs de données (de la classe frontière) du côté de l'utilisateur de la frontière logicielle à l'intérieur de la frontière de logiciel. il est matérialisé par un message de l'acteur vers la classe frontière.

- Un sous-processus Exit (X) qui déplace les attributs de données (de la classe frontière) de l'intérieur de la frontière logicielle à l'utilisateur. Il est matérialisé par un message de la classe frontière vers une autre classe frontière (si on veut sortir des informations à l'utilisateur, on invoque une classe frontière)

- Cas où il existe une seule couche : Les cas d'utilisation de cette couche sont réalisés par l'interaction des acteurs avec les trois classes d'analyse de frontière, de contrôle et d'entité. Dans ce cas on identifie les deux sous-processus fonctionnels de E et X de la même façon que le cas précédent, plus un troisième type :

  - Un sous-processus fonctionnel de type Lecture (R) ou Écriture (W). Il est matérialisé par un message arrivant à une classe analyse de type entité.

b) les sous-processus fonctionnels de la deuxième couche : la pièce de logiciel dans cette couche interagit avec les acteurs de la première couche qui sont des paquetages ou des classes d'analyse. Cette couche est appelée couche spécifique au domaine d'affaire et qui contient généralement les classes de type entité et de contrôle. Les cas d'utilisation sont réalisés dans cette couche par les interactions des acteurs de la première couche avec les classes de la couche. On identifie 3 types de sous-processus fonctionnels :

- Un sous-processus Entrée (E) qui déplace les attributs de données (de la classe entité) du côté de l'utilisateur de la frontière logicielle à l'intérieur de la frontière de logiciel. Il est matérialisé par un message de l'acteur vers une classe de la couche.

- Un sous-processus Exit (X) qui déplace les attributs de données (de la classe entité) de l'intérieur de la frontière logicielle à l'utilisateur. Il est matérialisé par un message d'une classe de la couche vers l'utilisateur.

- Un sous-processus fonctionnel de type Lecture (R) ou Écriture (W). Il est matérialisé par un message arrivant à une classe analyse de type entité et qui n'est pas un message de type Entrée (E). c'est-à-dire l'émetteur de message ne doit pas être un Acteur.

c) les sous-processus fonctionnels des autres couches : s'il existe d'autres couches, ils sont généralement semblables à la deuxième couche.

Les artefacts du modèle d'analyse ne nous permettent pas de différencier entre les sous-processus de lecture (R) et d'écriture (W). L'unique solution consiste à créer deux nouveaux stéréotypes pour les messages : Read pour la lecture et Write pour l'écriture.

À part ce problème, qui n'a aucune répercussion sur le processus de mesure, car il n'affecte pas la taille de logiciel à mesurer, tous les sous-processus fonctionnels sont identifiés.

2.1.2.2 Application de la fonction de mesure

Dans le modèle d'analyse :

- Identifier les couches,
- Pour chaque couche, identifier les paquetages d'analyse
- Pour chaque paquetage d'analyse, identifier les réalisations – analyse cas d'utilisation (une réalisation – analyse cas d'utilisation est un processus fonctionnel),
- pour chaque réalisation – analyse cas d'utilisation, identifier tous les diagrammes d'interaction (collaboration ou séquence, et non pas les deux à la fois).
- Pour chaque diagramme d'interaction, identifier les sous-processus fonctionnels. Un sous-processus fonctionnel correspond à 1 Cfsu.

2.1.2.3 Agrégation du résultat de mesure

$\text{Taille}_{\text{Cfsu}}(\text{couche}_i) = \sum \text{taille}(\text{entrées}_i) + \sum \text{taille}(\text{exits}_i) + \sum \text{taille}(\text{lectures}_i \text{ ou écritures}_i)$.

2.2 Conception

La mesure de la taille fonctionnelle de COSMIC-FFP est basée seulement sur les exigences fonctionnelles des utilisateurs, en excluant les exigences de qualité et techniques. En plus, la mesure fonctionnelle, de façon générale, doit être indépendante du développement technique et les décisions d'implémentation, car elle peut être utilisée pour comparer la production de différentes techniques et technologies d'implémentation [6].

Le modèle de conception est plus détaillé par rapport au modèle d'analyse. Ces artefacts peuvent être très intéressants pour la génération des éléments du modèle de COSMIC-FFP, Mais, malheureusement ce n'est pas le cas pour les raisons suivantes :

- Le modèle de conception est un modèle objet décrivant la réalisation physique des cas d'utilisation, en s'attachant aux effets conjugués des exigences fonctionnelles et non fonctionnelles et des autres types de contraintes sur le système en question [10].
- Une réalisation – conception de cas d'utilisation fournit une réalisation physique de la réalisation – analyse de cas d'utilisation à laquelle est liée par une relation de traçabilité, et prend en compte la plupart des exigences non fonctionnelles et celles qui sont liées aux activités d'implémentation [10].

- Les classes d'UML qui facilitent généralement l'identification des groupes et des attributs de données ne peuvent pas être exploitées dans le modèle de conception. Une classe de conception est une abstraction transparente d'une classe ou d'une construction équivalente de l'implémentation du système, car elle est spécifiée (spécification des attributs qui sont rapprochés par les attributs de données en COSMIC-FFP) en utilisant le même langage que le langage de programmation et sa relation avec les autres classes doit trouver une signification directe lors de son implémentation [10].
- Les messages dans les diagrammes de séquences ne peuvent pas être exploités pour identifier les sous-processus fonctionnels car, ils deviennent des opérations techniques de l'objet invoqué ou d'une interface réalisée par l'objet, et ils sont liés au langage d'implémentation [10].

Les principaux artefacts nécessaires pour le rapprochement tels que les classes et leurs attributs, les diagrammes d'interaction et leurs messages et les réalisations des cas d'utilisation sont liés directement à l'implémentation physique du système et décrits avec le langage de programmation, ce qui est contre les principes même de la taille fonctionnelle. On résume qu'on ne peut pas rapprocher ces artefacts avec le modèle COSMIC-FFP.

3. Conclusion

Seul, le modèle d'analyse présente des artefacts excellents pour les phases de mappage et de mesure de COSMIC-FFP. Mais, comme le modèle de cas d'utilisation et le modèle d'analyse sont liés à travers les cas d'utilisation et leur réalisation, il est préférable d'utiliser les deux modèles car ils permettent tout d'abord de vérifier la cohérence entre eux, ensuite faciliter l'identification des acteurs et les réalisations – analyse cas d'utilisation, surtout lorsqu'il existe une seule couche du système entier. On remarque aussi que la taille du logiciel identifié à partir du modèle de cas d'utilisation est égale à la taille de la première couche (dans le cas où le système possède plusieurs couches), car ils ont les mêmes processus fonctionnels et les mêmes classes frontières identifiées qui participent pour réaliser ces cas d'utilisation dans la première couche (appelée aussi couche de présentation). Le tableau suivant résume les différents éléments COSMIC-FFP identifiés dans les phases de mappage et de mesure de COSMIC-FFP en utilisant les artefacts de deux modèles RUP de cas d'utilisation et d'analyse.

		Expression des besoins	Analyse
Phase de mappage	**couches**	une seule couche logicielle du système entier est identifiée.	Un, deux ou plusieurs couches logicielles sont identifiées. Dans le cas d'une seule couche, c'est celle du système entier.
	frontières	Une seule frontière est identifiée qui sépare la couche logicielle du système entier et l'environnement des utilisateurs. Tous les utilisateurs de l'unique couche logicielle du système entier sont identifiés.	Une, Deux ou plusieurs frontières sont identifiées selon le nombre de couches. Tous les utilisateurs de toutes les couches sont identifiés.
	Processus fonctionnels	les Processus fonctionnels de la couche logicielle du système entier sont identifiés. Les événements déclencheurs des processus fonctionnels de la couche logiciel du système entier sont identifiés.	Tous les processus fonctionnels de toutes les couches sont identifiés. Tous les événements qui déclenchent les processus fonctionnels dans toutes les couches sont identifiés.
	Groupe de données	Quelques groupes de données sont identifiés.	Tous les groupes de données du système sont identifiés
	Attributs de données	Quelques attributs de données sont identifiés.	Tous les attributs de données sont identifiés.
Phase de mesure	**Sous-processus fonctionnels**	Seulement les sous-processus fonctionnels de la couche logicielle du système entier de type Entrée et eXit sont identifiés	3 types de sous-processus fonctionnels sont identifiés : Entrée, eXit et Lecture (R) ou Écriture (W). On n'arrive pas à distinguer entre R et W.
	Fonction de mesure	Les diagrammes d'interaction de chaque réalisation d'un cas d'utilisation dans la couche logicielle du système entier avec les classes frontières fournissent les mouvements de données : ❑ Un message direction Acteur – Classe frontière est un mouvement de données de type E, est égale à 1 Cfsu. ❑ Un message direction Classe frontière – Classe frontière est un mouvement de données de type X est égale à 1 Cfsu.	Les diagrammes d'interaction de chaque réalisation - analyse de cas d'utilisation dans chaque paquetage d'analyse et pour chaque couche fournissent les mouvements de données. Les messages spécifiques à chaque classe d'analyse dans une couche permettent d'identifier les types de mouvement de données. Chaque mouvement de données correspond à un Cfsu.
	agrégation	Taille Partielle $_{Cfsu}$ (couche du système entier) = Σ taille(entrées) + Σ taille (exits).	Taille$_{Cfsu}$ (couche$_i$) = Σ taille(entrées$_i$) + Σ taille (exits$_i$) + Σ taille (lectures$_i$ ou écritures$_i$).

CHAPITRE 4 : ÉTUDE DE CAS – GESTION GLOSSAIRE

L'objectif de ce chapitre est d'étudier un cas qui fait apparaître tous les aspects de rapprochement entre COSMIC-FFP et RUP du chapitre précédent et en particulier la notion de couches. On a choisi un cas simple d'une application Web extrait du livre Building Web Applications with UML [12] qui présente l'avantage d'être simple, développé selon le processus RUP et implémenté avec la technologie Microsoft (VB, ASP, SQL Server). L'architecture de l'application a été modifiée pour faire apparaître la notion de couche.

L'objectif de cette étude de cas ne réside pas dans le développement de l'application, ni dans sa mesure, mais de vérifier, seulement, les règles de rapprochement. Le choix d'un cas simple présente l'avantage de minimiser la polémique qui se crée, généralement, selon le niveau de compréhension des exigences utilisateurs, entre les développeurs sur la façon d'architecturer la même application et entre les mesureurs sur la taille du même logiciel à estimer. Notre cas a beaucoup de chance d'être architecturé et mesuré de la même façon par des personnes différentes.

L'architecture d'une application Web présente généralement plusieurs couches. On distingue aux moins quatre couches dans son modèle de déploiement : une couche de présentation, une couche du domaine d'affaire, une couche de middleware et une couche du logiciel du système d'opération.

1. Énoncé du cas

L'application Glossaire propose une version en ligne d'un glossaire de projet de développement logiciel. Cette application met à la disposition des membres de l'équipe de développement, par l'intermédiaire d'un navigateur Web, une base de données de termes d'un glossaire propre à un projet. Les membres de l'équipe peuvent également modifier, ajouter et supprimer des entrées de la base à l'aide de la même interface Web.

Cette application profite de l'Internet existant, auquel tous les membres de l'équipe ont accès. Tout membre de l'équipe peut se promener, effectuer des recherches et des modifications d'entrées dans le glossaire.

L'application Glossaire se veut une application simple bâtie avec ASP.

2. Mesure directe de l'application Glossaire à partir des FURs

Selon l'énoncé de l'étude de cas, on peut identifier les FURs suivantes :

- **FUR1** : L'application doit permettre aux utilisateurs de consulter tous les termes (mot et définition) du glossaire par la première lettre du mot.
- **FUR-2** : L'application doit permettre aux utilisateurs de consulter tous les termes (mot et définition) du glossaire par un mot présent soit dans le mot, soit dans la définition.
- **FUR-3** : L'application doit permettre aux utilisateurs de modifier, ajouter ou supprimer des entrées dans le glossaire.
- **FUR-4** : Toutes les fonctionnalités des exigences précédentes doivent interagir avec une base de données pour la manipulation des données persistantes à travers un navigateur Web.

Phases		**Identification**
Phase de mappage	**couches**	D'habitude les mesureurs de COSMIC-FFP identifient une seule couche pour une petite application pareille. C'est la couche du système entier.
	frontières	Une seule frontière qui sépare l'environnement des utilisateurs et la couche du système entier. Deux utilisateurs peuvent être identifiés : un lecteur, selon FUR1 et FUR2, qui consulte les termes du glossaire et un éditeur, selon FUR3, qui ajoute, modifie et supprime les termes du glossaire.
	Processus fonctionnels	Trois processus fonctionnels peuvent être identifiés selon les FURs : ❑ PF-1 : Consulter un terme par première lettre de mot : il est identifié à partir de FUR-1 et FUR-4 et permet d'afficher tous les termes (mot et définition) ayant un mot qui commence par une lettre donnée. ❑ PF-2 : Consulter des termes par un mot : il est identifié à partir de FUR-2 et FUR-4 et permet d'afficher tous les termes (mot et définition) ayant un mot donné. ❑ PF-3 : Éditer un terme dans le glossaire : il est identifié à partir du FUR-3 et FUR-4 et permet d'ajouter, de modifier ou de supprimer un terme (mot et définition) dans le glossaire. Trois événements se déclenchent dans l'environnement utilisateur en dehors de la frontière de la couche unique du système entier. Chacun deux initie un processus fonctionnel de la couche. Les événements sont matérialisés par l'introduction de l'utilisateur d'une information (correspondant à un attribut de données) puis la sélection d'un bouton envoi.
	Groupe de données	Un seul groupe de données qui existe appelé Glossaire qui contient le terme du glossaire décrit par deux attributs de données : mot et définition.
	Attributs de données	Deux attributs de données : Mot et Définition

Phases		Identification
Phase de mesure	**Sous-processus fonctionnels**	Neuf sous-processus fonctionnels sont identifiés (trois par processus fonctionnel): PF-1 : contient 3 sous-processus : ❑ Entrée (E) : l'introduction d'une lettre par l'utilisateur Lecteur qui correspond à l'attribut de donnée Mot. ❑ Lecture (R) : une lecture dans le groupe de données Glossaire pour la recherche des termes ayant un mot qui commence par la lettre introduit. ❑ Exit (X) : affichage d'une liste de termes correspondant au critère de recherche (la liste peut être vide indiquant la non satisfaction du critère). PF-2 : contient trois sous-processus : ❑ Entrée (E) : l'introduction d'un mot par l'utilisateur Lecteur qui correspond aux attributs de données Mot ou Définition. ❑ Lecture (R) : une lecture dans le groupe de données Glossaire pour la recherche des termes (mot et définition) possédant le mot introduit. ❑ Exit (X) : affichage d'une liste de termes correspondant au critère de recherche (la liste peut être vide indiquant la non satisfaction du critère). PF-3 : contient trois sous-processus : Au départ l'utilisateur Éditeur joue le rôle du Lecteur, en exécutant les processus PF-1 et PF-2 en faisant la recherche dans le Glossaire par une lettre ou par un mot. Selon le résultat, il décide d'ajouter, de supprimer ou de modifier un terme. ❑ Entrée (E) : l'Éditeur introduit le mot et sa définition dans le cas d'un ajout, modifie le mot et sa définition dans le cas de modification ou il sélectionne le mot en cas de suppression. Toutes ces informations correspondent aux attributs de données Mot et Définition. ❑ Écriture (W) : une écriture dans le groupe de données Glossaire pour ajouter, supprimer ou modifier un terme. ❑ Exit (X) : affichage du résultat de l'action demandée (réussie ou non).
	Fonction de mesure	Chaque sous-processus identifié précédemment correspond à 1Cfsu.
	agrégation	$Taille_{Cfsu}$ (couche unique du système entier) = Σ taille(entrées) + Σ taille (exits) + Σ taille (lectures) + Σ taille (Écritures) = 9.

3. Mesure de l'application à partir des artefacts RUP

3.1 Discipline : Expression des besoins

3.1.1 Production des artefacts RUP

Artefact : Acteur

On identifie deux acteurs : Lecteur et Éditeur

- **Lecteur** : il représente tout utilisateur qui parcourt et lit les termes (mot et définition) du Glossaire [12].
- **Éditeur** : Un éditeur est un Lecteur qui ajoute, édite ou supprime les termes du glossaire [12].

Artefact : Cas d'utilisation

On identifie 3 cas d'utilisation :

- **Éditer terme Glossaire** : Ajouter, éditer ou supprimer un terme du glossaire (le mot et sa définition) [12].
- **Lire Glossaire** : Parcourir les termes du Glossaire (les mots et leur définition), à partir de la première lettre du terme [12].
- **Rechercher dans Glossaire** : Parcourir les termes du glossaire (les mots et leur définition), à partir de la recherche d'un mot clé présent soit dans le mot, soit dans la définition [12].

Artefact : modèle de cas d'utilisation

Il est constitué par un seul diagramme de cas d'utilisation (*figure 4.1*):

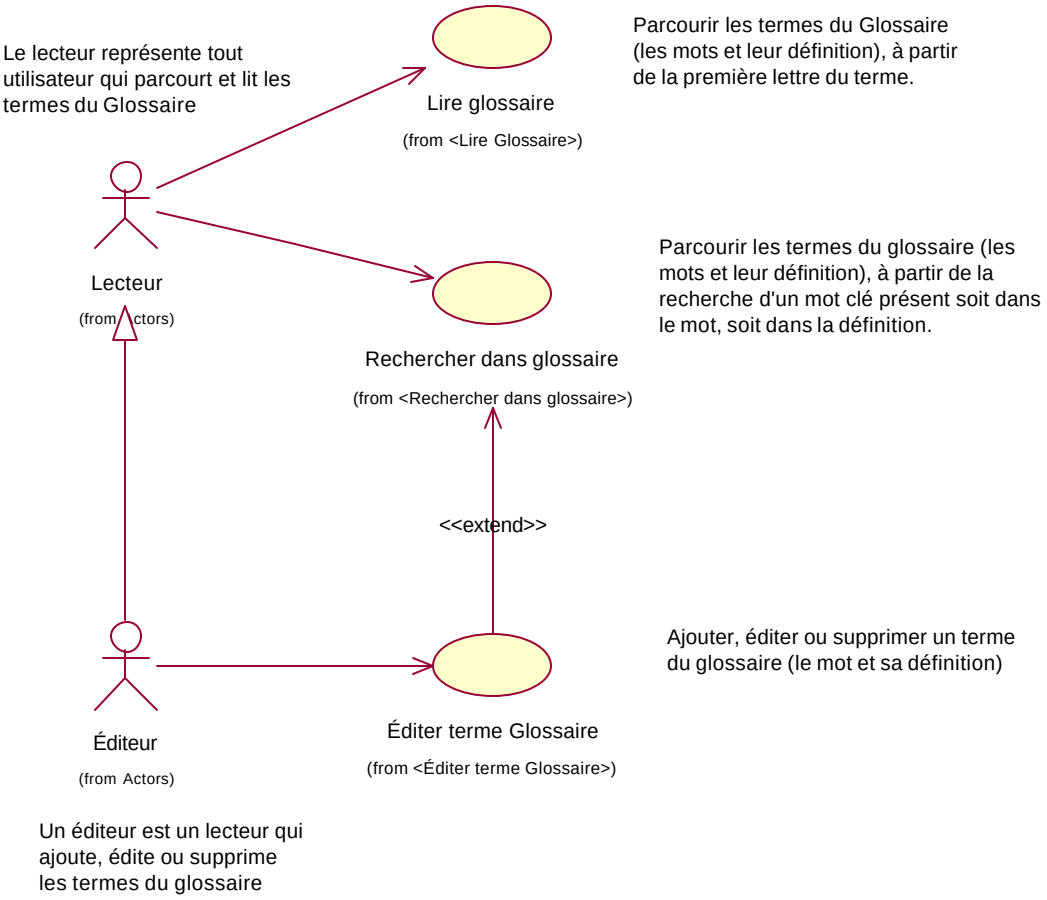

Figure 4.1 : Diagramme de cas d'utilisation de l'application Glossaire

Artefact : story-board de cas d'utilisation

Classes frontières :

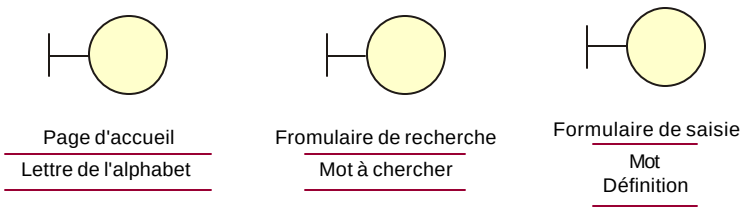

Figure 4.2 : Classes frontières

Cas d'utilisation : lire glossaire

Deux classes frontières participent au story-board cas d'utilisation lire glossaire: Page d'accueil et Formulaire de saisie.

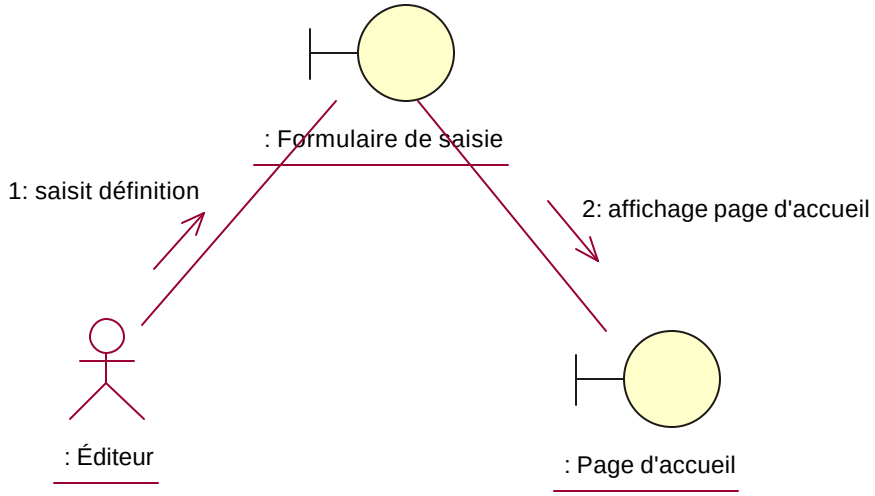

Figure 4.3 : diagramme de collaboration story-board cas d'utilisation: lire glossaire

Cas d'utilisation : Rechercher dans glossaire

Deux classes frontières participent au story-board cas d'utilisation rechercher dans glossaire : Formulaire de recherche et Résultat de recherche.

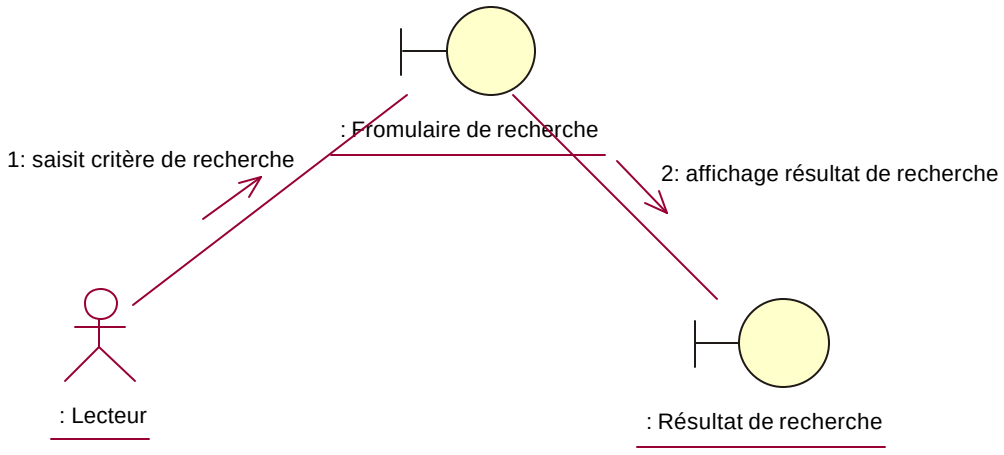

Figure 4.4 : diagramme de collaboration story-board cas d'utilisation: Rechercher dans glossaire

Cas d'utilisation : Éditer glossaire

Deux classes frontières participent au story-board cas d'utilisation éditer glossaire : Formulaire de saisie et Page d'accueil.

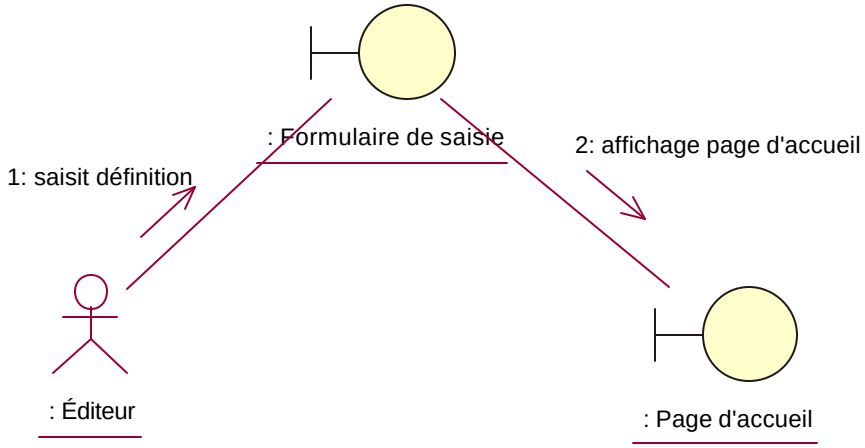

Figure 4.5 : diagramme de collaboration story-board cas d'utilisation: Éditer glossaire

3.1.2 Processus de mesure utilisant les artefacts

		Identification	Artefact RUP
Phase de mappage	**couches**	une seule couche logicielle du système entier est identifiée contenant tous les processus fonctionnels.	Diagramme de cas d'utilisation dans le modèle de cas d'utilisation.
	frontières	Une seule frontière est identifiée qui sépare la couche logicielle du système entier et l'environnement des utilisateurs. Deux utilisateurs sont identifiés : Lecteur et Éditeur.	Diagramme de cas d'utilisation dans le modèle de cas d'utilisation. Artefact Acteur.
	Processus fonctionnels	Trois processus fonctionnels : Lire glossaire, Rechercher dans glossaire et Éditer terme glossaire. Trois événements sont identifiés matérialisés par les relations entre les acteurs et les cas d'utilisation dans le diagramme de cas d'utilisation. Les noms de ces relations ne sont pas spécifiés dans notre cas, donc les événements n'auront pas de noms.	Artefact cas d'utilisation. Artefact diagramme de cas d'utilisation du modèle de cas d'utilisation.

		Identification	Artefact RUP
	Groupe de données	Trois groupes de données sont identifiés : ❑ Page d'accueil qui contient l'attribut de données Lettre de l'alphabet. ❑ Formulaire de recherche qui contient l'attribut de données Mot à chercher. ❑ Formulaire de saisie qui contient les attributs de données Mot et Définition.	Artefact classes frontières : Page d'accueil, Formulaire de recherche et Formulaire de saisie.
	Attributs de données	Lettre de l'alphabet, Mot à chercher, Mot et Définition.	Artefact classes frontières : Page d'accueil, Formulaire de recherche et Formulaire de saisie.
Phase de mesure	**Sous-processus fonctionnels**	Six sous-processus fonctionnels sont identifiés : Diagramme de collaboration Story-board de cas d'utilisation Lire glossaire : ❑ Un sous-processus de type Entrée (E) : Sélection lettre. ❑ Un sous-processus de type Exit (X) : Affichage résultat. Diagramme de collaboration Story-board de cas d'utilisation Rechercher dans glossaire : ❑ Un sous-processus de type Entrée (E) : Saisit critère de recherche. ❑ Un sous-processus de type Exit (X) : Affichage résultat de recherche Diagramme de collaboration Story-board de cas d'utilisation Éditer glossaire : ❑ Un sous-processus de type Entrée (E) : Saisit définition. ❑ Un sous-processus de type Exit (X) : Affichage page d'accueil.	Diagrammes de collaboration de l'artefact story-board des cas d'utilisation.
	Fonction de mesure	Chaque sous-processus fonctionnel identifié correspond à 1 Cfsu.	
	Agrégation	$Taille_{Cfsu}$ (couche du système entier) = Σ taille(entrées) + Σ taille (exits) = 3+3 = 6	

3.2 Analyse

On étudie dans la sous-discipline Analyse deux cas différents : cas où l'architecte ne découvre pas de couche, donc l'unique couche est celle du système entier, et l'autre cas où il découvre deux

couches : couche de présentation (couche spécifique à l'application) et une couche de modèle d'affaire (couche du domaine d'affaire).

3.2.1 Cas d'une seule couche

Pour simplifier encore le cas, on suppose qu'il existe un seul paquetage d'analyse appelé Gestion de glossaire et tous les cas d'utilisation se réalisent dans ce paquetage.

3.2.1.1 Production des artefacts

Les principaux artefacts générés dans cette sous-discipline sont les classes d'analyse, les réalisations – analyse de cas d'utilisation qui contient un diagramme de classes d'analyse participantes à la réalisation et des diagrammes de collaboration.

Artefact : Réalisation - Analyse de cas d'utilisation : Lire glossaire

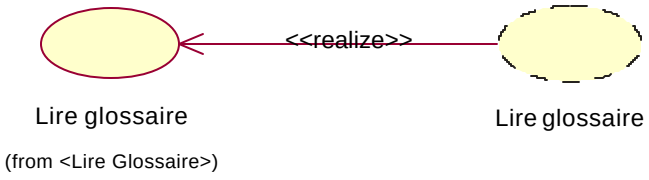

Figure 4.6 : Lire glossaire du modèle d'analyse réalise Lire Glossaire du modèle de cas d'utilisation

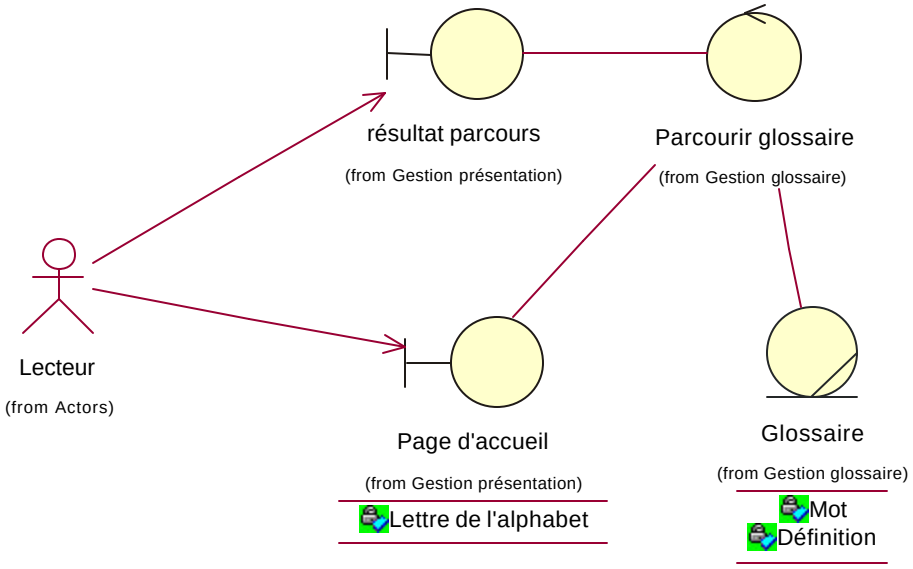

Figure 4.7 : Classes participantes à la réalisation – Analyse du cas d'utilisation : Lire Glossaire

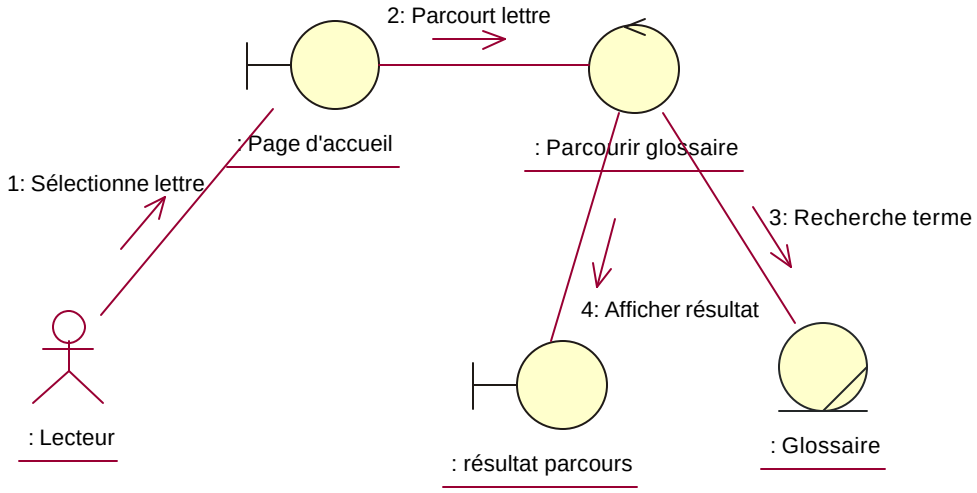

Figure 4.8 : Diagramme de collaboration à la réalisation – analyse cas d'utilisation : Lire Glossaire

Artefact : Réalisation - Analyse de cas d'utilisation : Rechercher dans glossaire

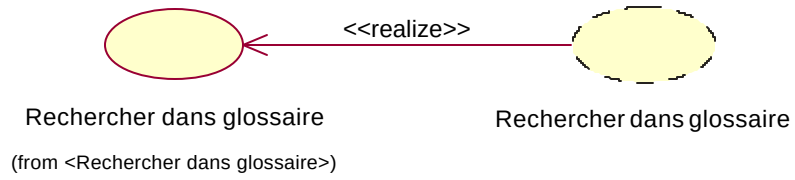

Figure 4.9 : Rechercher dans glossaire du modèle d'analyse réalise Rechercher dans glossaire du modèle de cas d'utilisation

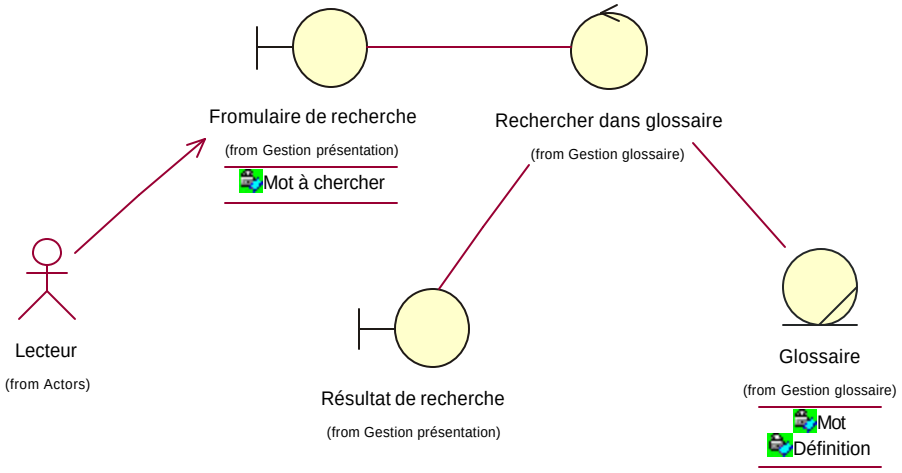

Figure 4.10 : Classes participantes à la réalisation – Analyse du cas d'utilisation : Rechercher dans glossaire

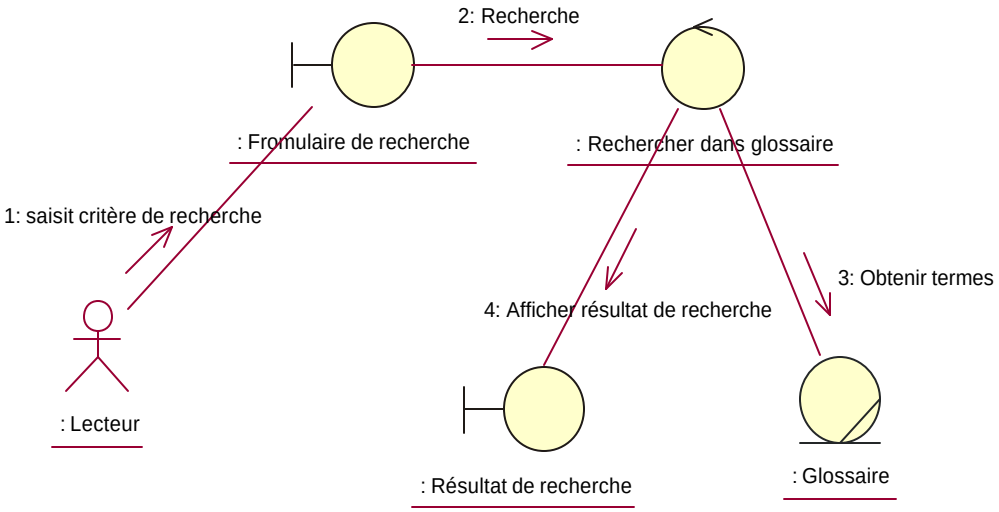

Figure 4.11 : Diagramme de collaboration à la réalisation – analyse cas d'utilisation : Rechercher dans glossaire

Artefact : Réalisation - Analyse de cas d'utilisation : Éditer terme Glossaire

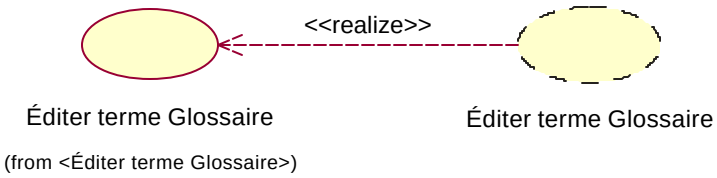

Figure 4.12 : Éditer terme Glossaire du modèle d'analyse réalise Éditer terme Glossaire du modèle de cas d'utilisation

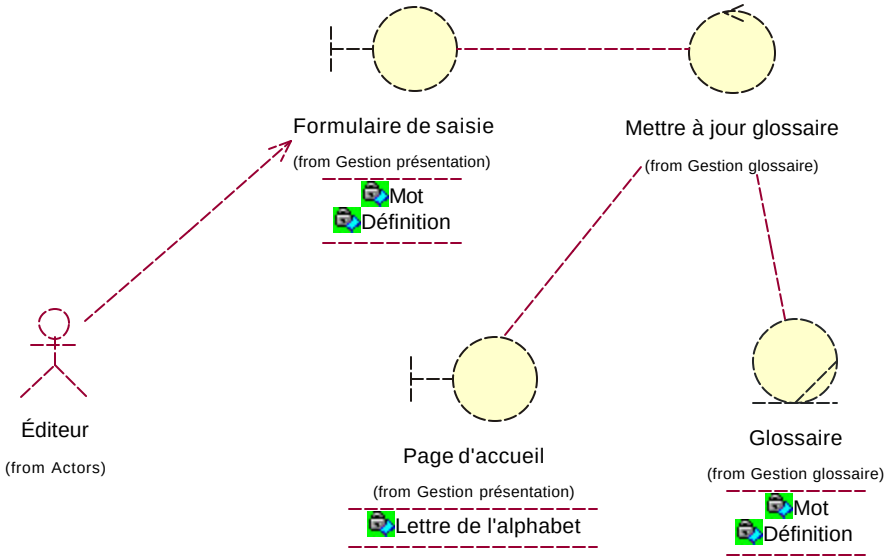

Figure 4.13 : Classes participantes à la réalisation – Analyse du cas d'utilisation : Éditer terme Glossaire

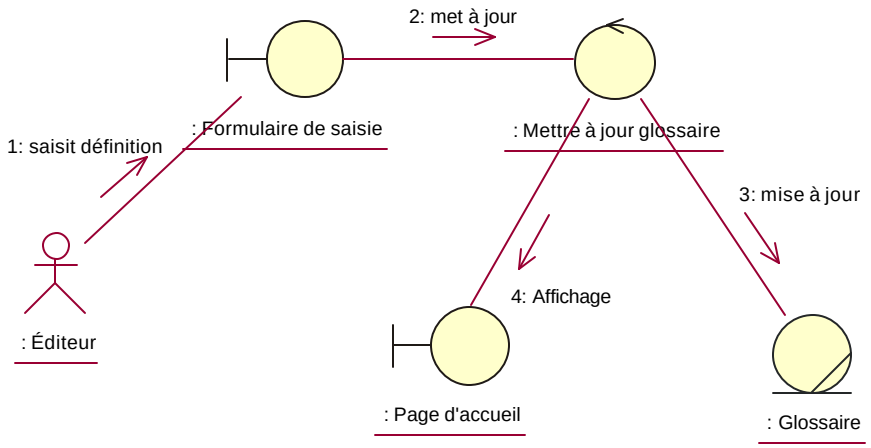

Figure 4.14 : Diagramme de collaboration à la réalisation – analyse cas d'utilisation : Éditer terme Glossaire

3.2.1.2 Processus de mesure utilisant les artefacts

		Identification	Artefact RUP
Phase de mappage	**couches**	En absence de couches dans le modèle d'analyse, la seule couche logicielle du système est celle du système entier identifiée à partir du modèle de cas d'utilisation.	Diagramme de cas d'utilisation dans le modèle de cas d'utilisation.
	frontières	Une seule frontière est identifiée qui sépare la couche logicielle du système entier et l'environnement des utilisateurs. Deux utilisateurs sont identifiés : Lecteur et Éditeur.	Diagramme de cas d'utilisation dans le modèle de cas d'utilisation. Artefact Acteur.
	Processus fonctionnels	Selon les règles de rapprochement, un processus fonctionnel dans une couche est la réalisation d'un cas d'utilisation dans cette couche, donc nous avons trois processus fonctionnels correspondant à la réalisation des cas d'utilisation: Lire glossaire, Rechercher dans glossaire et Éditer terme glossaire, et qui portent le même nom Trois événements sont identifiés matérialisés par les relations entre les acteurs et les classes d'analyses dans le diagramme de classes participantes à la réalisation de chaque cas d'utilisation.	Diagramme de cas d'utilisation montrant la traçabilité entre le cas d'utilisation et sa réalisation. Artefact diagramme de classes participantes.

		Identification	Artefact RUP
	Groupe de données	Un seul groupe de données est identifié qui correspond à la classe entité Glossaire contenant tous les attributs de données. Les classes frontières ne sont pas des groupes de données car elles contiennent des attributs redondants par rapport au groupe Glossaire,	Artefact classes d'analyses participantes à la réalisation – analyse de cas d'utilisation.
	Attributs de données	Mot et Définition du groupe de données Glossaire.	Artefact classes entité Glossaire.
Phase de mesure	**Sous-processus fonctionnels**	neuf sous-processus fonctionnels sont identifiés : Diagramme de collaboration réalisation - analyse de cas d'utilisation Lire glossaire : ❑ Un sous-processus de type Entrée (E) : Sélectionne lettre. ❑ Un sous-processus de type Lecture (R) ou Écriture (W) : Recherche terme. ❑ Un sous-processus de type Exit (X) : Afficher résultat. Diagramme de collaboration réalisation - analyse de cas d'utilisation Rechercher dans glossaire : ❑ Un sous-processus de type Entrée (E) : Saisit critère de recherche. ❑ Un sous-processus de type Lecture (R) ou Écriture (W) : Obtenir terme. ❑ Un sous-processus de type Exit (X) : Afficher résultat de recherche Diagramme de collaboration réalisation - analyse de cas d'utilisation Éditer glossaire : ❑ Un sous-processus de type Entrée (E) : saisit définition. ❑ Un sous-processus de type Lecture (R) ou Écriture (W) : Mise à jour. ❑ Un sous-processus de type Exit (X) : Affichage.	Diagrammes de collaboration de l'artefact réalisation – analyse des cas d'utilisation.
	Fonction de mesure	Chaque sous-processus fonctionnel identifié correspond à 1 Cfsu.	
	Agrégation	$\text{Taille}_{\text{Cfsu}}$ (couche du système entier) = Σ taille(entrées) + Σ taille (exits) + Σ taille (lectures_i ou écritures_i) = 3+3+3 = 9.	

3.2.2 Cas de plusieurs couches

La première couche s'appelle la Couche de présentation, et la deuxième couche est celle du domaine d'affaire. Le processus de mesure s'effectue pour chaque couche.

Pour simplifier l'artefact, on considère que chaque couche contient un seul paquetage d'analyse.

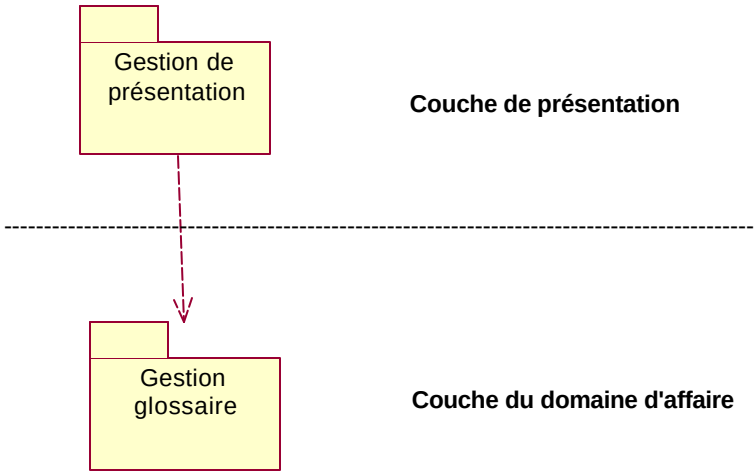

Figure 4.15 : Architecture en couches de l'application Glossaire

3.2.2.1 La couche de présentation

3.2.2.1.1 Production des artefacts

Les principaux artefacts générés dans cette sous-discipline sont les classes d'analyse, les réalisations – analyse de cas d'utilisation qui contient un diagramme de classes d'analyse participantes à la réalisation, et des diagrammes de collaboration.

Artefact : Réalisation - Analyse de cas d'utilisation : Lire glossaire

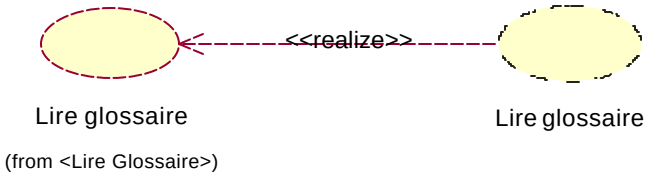

Figure 4.16 : Lire glossaire du modèle d'analyse réalise dans la couche présentation Lire Glossaire du modèle de cas d'utilisation

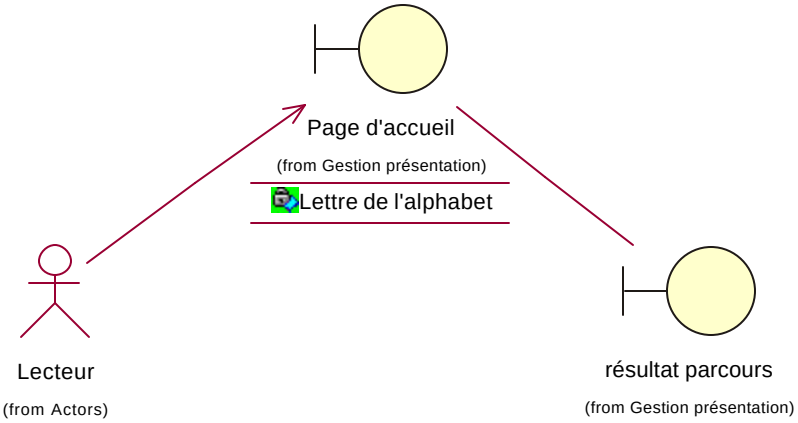

Figure 4.17 : Classes participantes dans la couche présentation à la réalisation – Analyse du cas d'utilisation : Lire Glossaire

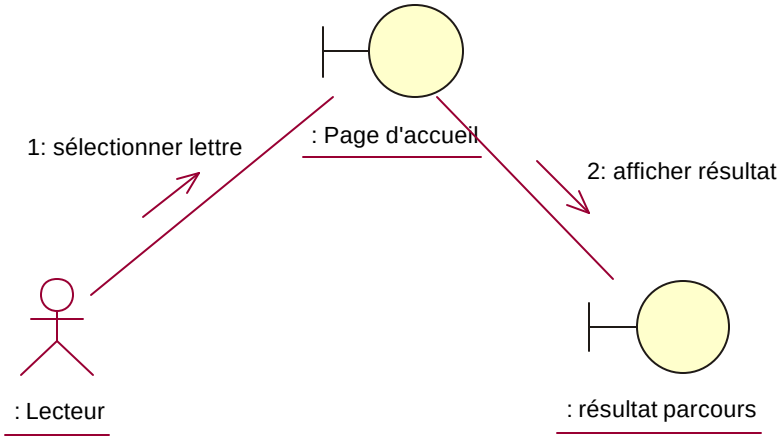

Figure 4.18 : Diagramme de collaboration dans la couche présentation à la réalisation – analyse cas d'utilisation : Lire Glossaire

Artefact : Réalisation - Analyse de cas d'utilisation : Rechercher dans glossaire

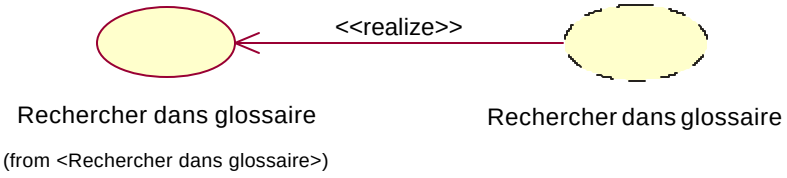

Figure 4.19 : Lire glossaire du modèle d'analyse réalise dans la couche présentation Rechercher dans glossaire du modèle de cas d'utilisation

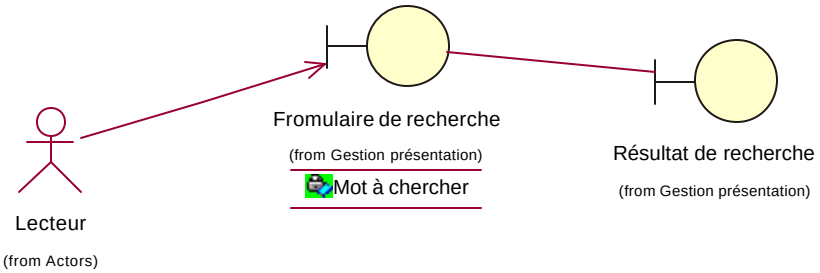

Figure 4.20 : Classes participantes dans la couche présentation à la réalisation – Analyse du cas d'utilisation : Rechercher dans glossaire

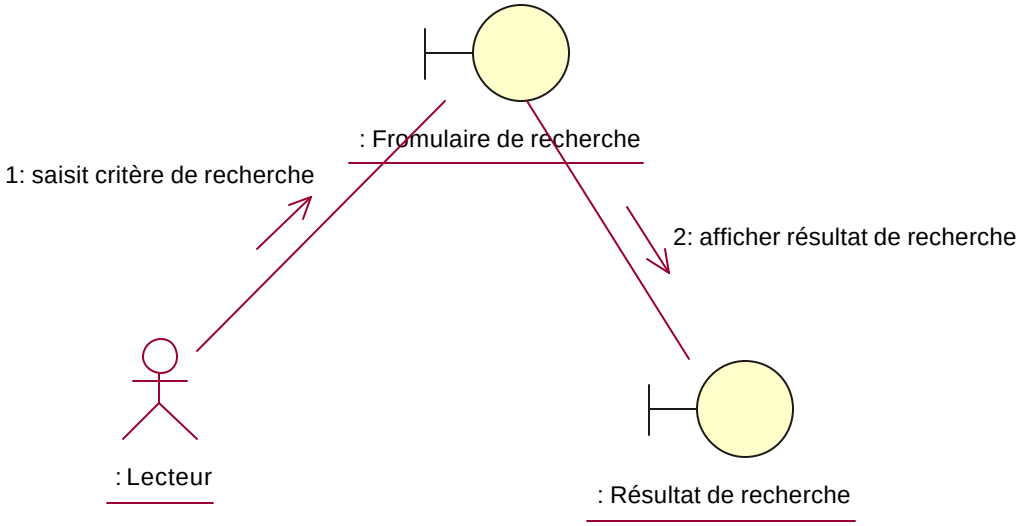

Figure 4.21 : Diagramme de collaboration dans la couche présentation à la réalisation – analyse cas d'utilisation : Rechercher dans glossaire

Artefact : Réalisation - Analyse de cas d'utilisation : Éditer terme glossaire

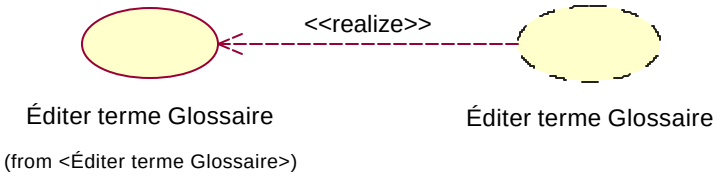

Figure 4.22 : Lire glossaire du modèle d'analyse réalise dans la couche présentation Éditer terme Glossaire du modèle de cas d'utilisation

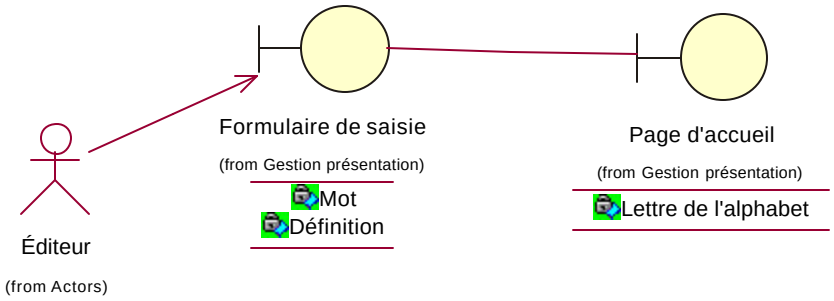

Figure 4.23 : Classes participantes dans la couche présentation à la réalisation – Analyse du cas d'utilisation : Éditer terme Glossaire

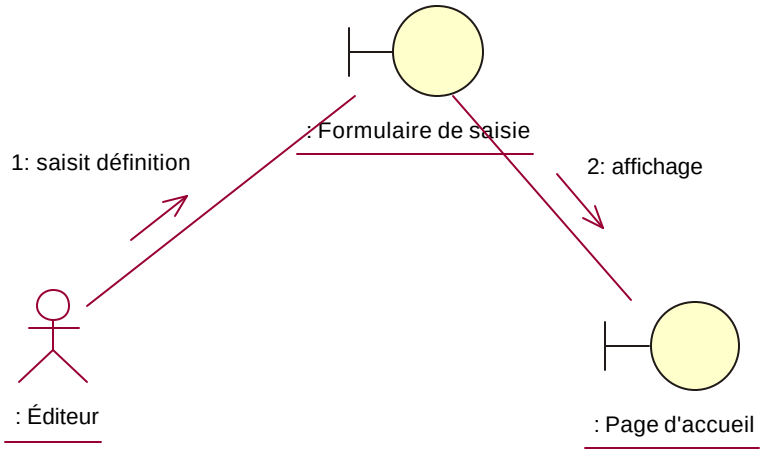

Figure 4.24 : Diagramme de collaboration dans la couche présentation à la réalisation – analyse cas d'utilisation : Éditer terme Glossaire

3.2.2.1.2 Processus de la mesure de la couche présentation

		Identification	Artefact RUP
Phase de mappage	**couches**	La couche RUP correspond à une couche COSMIC-FFP, donc on identifie une première couche appelée aussi couche de présentation.	Diagramme correspondant à l'architecture de l'application.
	frontières	Selon les règles de rapprochement, la frontière et les utilisateurs de la première couche sont ceux de la couche unique du système entier, identifiés à partir du modèle de cas d'utilisation : ❑ La frontière identifiée est celle qui sépare la première couche et l'environnement des utilisateurs. ❑ Deux utilisateurs sont identifiés : Lecteur et Éditeur.	Diagramme de collaboration ou de classes participantes.
	Processus fonctionnels	Un processus fonctionnel dans une couche est la réalisation d'un cas d'utilisation dans cette couche. Nous avons trois processus fonctionnels correspondant à la réalisation des cas d'utilisation dans la première couche à savoir : Lire glossaire, Rechercher dans glossaire et Éditer terme glossaire et qui portent le même nom. Trois événements sont identifiés matérialisés par les relations entre les acteurs et les classes d'analyses dans le diagramme de classes participantes à la réalisation de chaque cas d'utilisation.	Diagramme de cas d'utilisation montrant la traçabilité entre le cas d'utilisation et sa réalisation. Artefact diagramme de classes participantes.
	Groupe de données	Trois groupes de données sont identifiés : ❑ Page d'accueil qui contient l'attribut de données Lettre de l'alphabet. ❑ Formulaire de recherche qui contient l'attribut de données Mot à chercher. ❑ Formulaire de saisie qui contient les attributs de données Mot et Définition à ajouter.	Artefact classes frontières : Page d'accueil, Formulaire de recherche et Formulaire de saisie.
	Attributs de données	Lettre de l'alphabet, Mot à chercher, Mot et Définition.	Artefact classes frontières : Page d'accueil, Formulaire de recherche et Formulaire de saisie.

		Identification	Artefact RUP
Phase de mesure	**Sous-processus fonctionnels**	Six sous-processus fonctionnels sont identifiés : Diagramme de collaboration réalis ation - analyse cas d'utilisation dans la couche de présentation : Lire glossaire : ❑ Un sous-processus de type Entrée (E) : Sélectionner lettre. ❑ Un sous-processus de type Exit (X) : Afficher résultat. Diagramme de collaboration réalisation - analyse cas d'utilisation dans la couche de présentation : Rechercher dans glossaire : ❑ Un sous-processus de type Entrée (E) : Saisie critère de recherche. ❑ Un sous-processus de type Exit (X) : Affichage résultat de recherche Diagramme de collaboration réalisation - analyse cas d'utilisation dans la couche de présentation : Éditer terme glossaire : ❑ Un sous-processus de type Entrée (E) : Saisie définition. ❑ Un sous-processus de type Exit (X) : Affichage.	Diagrammes de collaboration de l'artefact réalisation - analyse cas d'utilisation dans la couche de présentation.
	Fonction de mesure	Chaque sous-processus fonctionnel identifié correspond à 1 Cfsu.	
	Agrégation	$Taille_{Cfsu}$ (couche de présentation) = Σ taille(entrées) + Σ taille (exits) = 3+3 = 6	

3.2.2.2 Couche du domaine d'affaire

3.2.2.2.1 Production des artefacts

Les principaux artefacts générés dans cette sous-discipline sont les classes d'analyse, les réalisations – analyse de cas d'utilisation qui contient un diagramme de classes d'analyse participantes à la réalisation et des diagrammes de collaboration.

Artefact : Réalisation - Analyse de cas d'utilisation : Lire glossaire

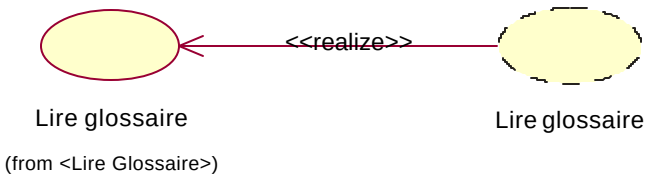

Figure 4.25 : Lire glossaire du modèle d'analyse réalise dans la couche du domaine d'affaire Lire Glossaire du modèle de cas d'utilisation

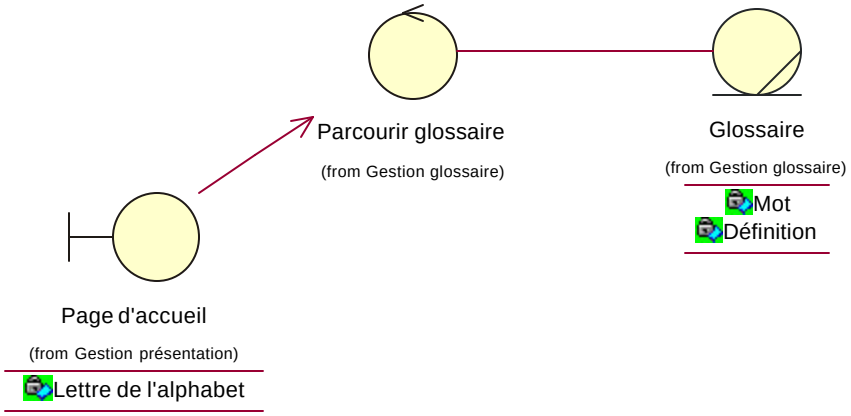

Figure 4.26 : Classes participantes dans la couche du domaine d'affaire à la réalisation – Analyse du cas d'utilisation : Lire Glossaire

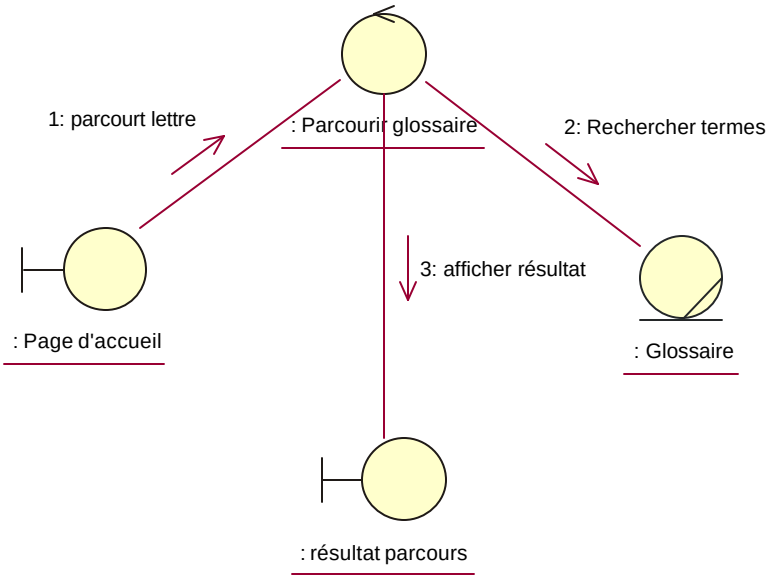

Figure 4.27 : Diagramme de collaboration dans la couche du domaine d'affaire à la réalisation – analyse cas d'utilisation : Lire Glossaire

Artefact : Réalisation - Analyse de cas d'utilisation : Rechercher dans glossaire

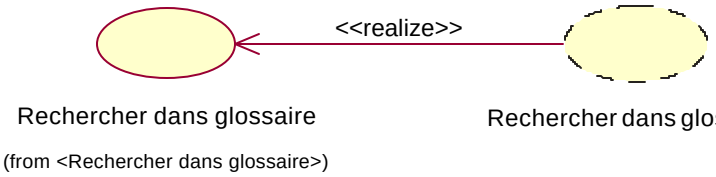

Figure 4.28 : Rechercher dans glossaire du modèle d'analyse réalise dans la couche du domaine d'affaire Rechercher dans glossaire du modèle de cas d'utilisation

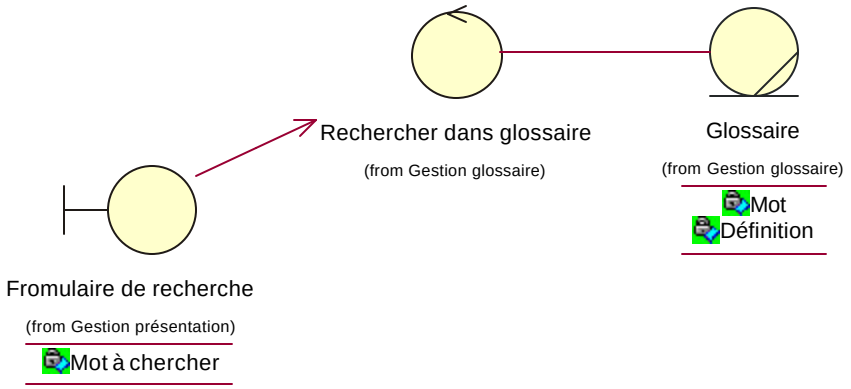

Figure 4.29 : Classes participantes dans la couche du domaine d'affaire à la réalisation – Analyse du cas d'utilisation : Rechercher dans glossaire

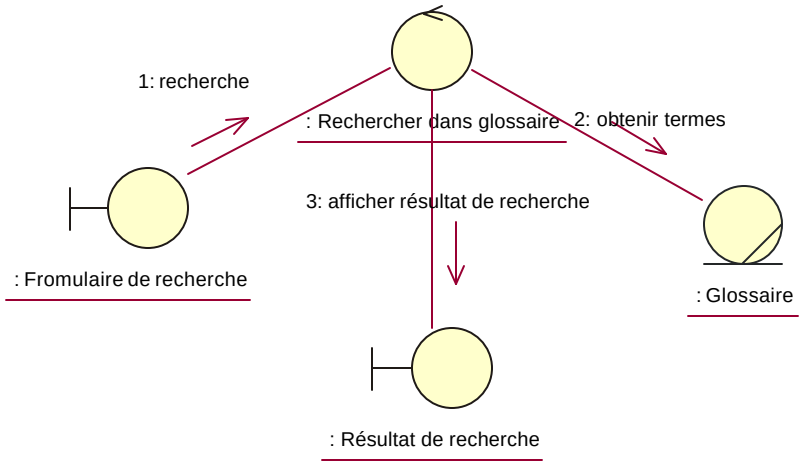

Figure 4.30 : Diagramme de collaboration dans la couche du domaine d'affaire à la réalisation – analyse cas d'utilisation : Rechercher dans glossaire

Artefact : Réalisation - Analyse de cas d'utilisation : Éditer terme Glossaire

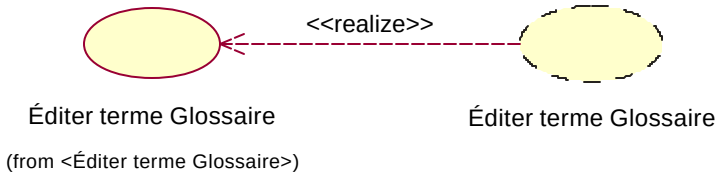

Figure 4.31 : Éditer terme Glossaire du modèle d'analyse réalise dans la couche du domaine d'affaire Éditer terme Glossaire du modèle de cas d'utilisation

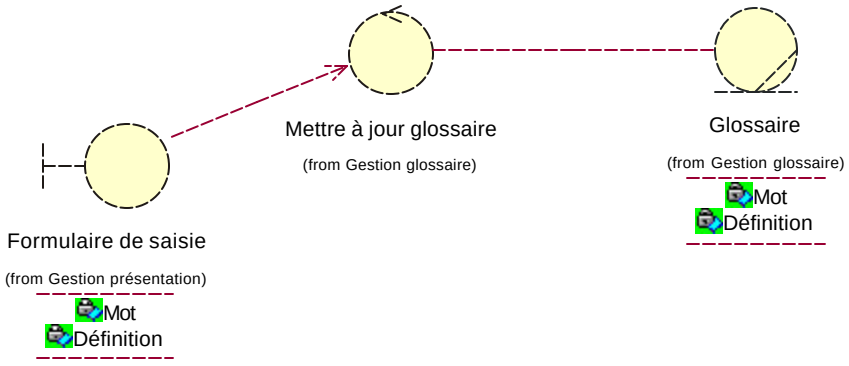

Figure 4.32 : Classes participantes dans la couche du domaine d'affaire à la réalisation – Analyse du cas d'utilisation : Éditer terme Glossaire

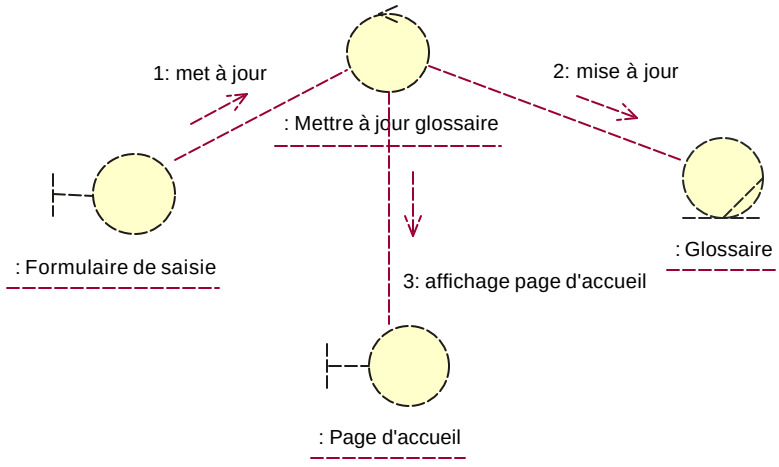

Figure 4.33 : Diagramme de collaboration dans la couche du domaine d'affaire à la réalisation – analyse cas d'utilisation : Éditer terme Glossaire

3.2.2.2.2 Processus de mesure utilisant les artefacts

		Identification	Artefact RUP
Phase de mappage	**couches**	Selon les règles de rapprochement une couche RUP correspond à une couche COSMIC-FFP. On identifie une deuxième couche appelée couche du domaine d'affaire.	Architecture de l'application
	frontières	La frontière identifiée sépare la première et la deuxième couche. Trois utilisateurs sont identifiés, se sont des classes d'analyse appartenant à la première couche : Page d'accueil, Formulaire de recherche et Formulaire de saisie.	Diagramme de collaboration réalisation – analyse cas d'utilisation dans la couche du domaine d'affaire.
	Processus fonctionnels	Selon les règles de rapprochement, un processus fonctionnel dans une couche est la réalisation d'un cas d'utilisation dans cette couche, donc nous avons trois processus fonctionnels correspondant à la réalisation des cas d'utilisation dans la couche du domaine d'affaire : Lire glossaire, Rechercher dans glossaire et Éditer terme glossaire et qui portent le même nom Trois événements sont identifiés matérialisés par les relations entre les acteurs et les classes d'analyses dans le diagramme de classes participantes à la réalisation de cas d'utilisation dans cette couche.	Diagrammes de collaboration et de classes réalisation – analyse cas d'utilisation dans la couche du domaine d'affaire.
	Groupe de données	Un seul groupe de données est identifié qui correspond à la classe entité Glossaire contenant tous les attributs de données.	Artefact classes d'analyses participantes à la réalisation – analyse de cas d'utilisation dans cette couche.
	Attributs de données	Mot et Définition du groupe de données Glossaire.	Artefact classes entité Glossaire.

		Identification	Artefact RUP
Phase de mesure	**Sous-processus fonctionnels**	neuf sous-processus fonctionnels sont identifiés : Diagramme de collaboration réalisation - analyse de cas d'utilisation Lire glossaire : ❑ Un sous-processus de type Entrée (E) : Parcourt lettre. ❑ Un sous-processus de type Lecture (R) ou Écriture (W) : Rechercher terme. ❑ Un sous-processus de type Exit (X) : Afficher résultat. Diagramme de collaboration réalisation - analyse de cas d'utilisation Rechercher dans glossaire : ❑ Un sous-processus de type Entrée (E) : Recherche. ❑ Un sous-processus de type Lecture (R) ou Écriture (W) : Obtenir terme. ❑ Un sous-processus de type Exit (X) : Afficher résultat de recherche Diagramme de collaboration réalisation - analyse de cas d'utilisation Éditer glossaire : ❑ Un sous-processus de type Entrée (E) : Met à jour. ❑ Un sous-processus de type Lecture (R) ou Écriture (W) : Mise à jour. ❑ Un sous-processus de type Exit (X) : Affichage page d'accueil.	Diagrammes de collaboration de l'artefact réalisation – analyse des cas d'utilisation dans la couche du domaine d'affaire.
	Fonction de mesure	Chaque sous-processus fonctionnel identifié correspond à 1 Cfsu.	
	Agrégation	$Taille_{Cfsu}$ (couche du système entier) = Σ taille(entrées) + Σ taille (exits) + Σ taille ($lectures_i$ ou $écritures_i$) = 3+3+3 = 9.	

4 Tableau récapitulatif

	Mesure Manuelle sans tenir compte des couches	**Mesure selon les artefacts RUP**			
		Discipline Expression des besoins	**Sous-discipline Analyse**		
			Sans couche	**Avec couches**	
				1ère couche	**2ème couche**
Couches	La couche du système entier.	La couche du système entier	La couche du système entier	Couche de présentation	Couche du domaine d'affaire
Frontières	Entre l'environnement des utilisateurs et la couche du système entier. 2 utilisateurs : Lecteur et Éditeur	Entre l'environnement des utilisateurs et la couche du système entier. 2 utilisateurs : Lecteur et Éditeur	Entre l'environnement des utilisateurs et la couche du système entier. 2 utilisateurs : Lecteur et Éditeur	Entre l'environnement des utilisateurs et la première couche. 2 utilisateurs : Lecteur et Éditeur	Une frontière séparant la première et la deuxième couche. 3 utilisateurs : Page d'accueil, Formulaire de recherche et Formulaire de saisie
Processus fonctionnels	3 PFs : Consulter terme par 1ère lettre du mot, Consulter termes par un mot, Éditer terme dans le glossaire. 3 événements déclenchent PFs	3 PFs : Lire glossaire, Rechercher dans glossaire et Éditer terme glossaire. 3 événements déclenchent les PFs	3 PFs : Lire glossaire, Rechercher dans glossaire et Éditer terme glossaire. 3 événements déclenchent les PFs	3 PFs : Lire glossaire, Rechercher dans glossaire et Éditer terme glossaire. 3 événements déclenchent les PFs	3 PFs : Lire glossaire, Rechercher dans glossaire et Éditer terme glossaire. 3 événements déclenchent les PFs
Groupes de données	Un GP : Glossaire	3 GPs : Page d'accueil, Formulaire de recherche, Formulaire de saisie.	Un GP : Glossaire	3 GPs : Page d'accueil, Formulaire de recherche, Formulaire de saisie.	Un GP : Glossaire
Attributs de données	2 attributs de données : Mot et Définition	4 attributs : Lettre de l'alphabet, Mot à chercher, Mot et Définition.	2 attributs de données : Mot et Définition	4 attributs : Lettre de l'alphabet, Mot à chercher, Mot et Définition.	2 attributs de données : Mot et Définition
Sous-processus	9 S-PFs : 3 Entrées, 3 Exits et 2 Lectures et 1 Écritures	6 S-PFs : 3 Entrées et 3 Exits.	9 S-PFs : 3 Entrées, 3 Exits et 3 Lectures ou Écritures.	6 S-PFs : 3 Entrées et 3 Exits.	9 S-PFs : 3 Entrées, 3 Exits et 3 Lectures ou Écritures
taille	Taille (Couche) = 9	Taille (Couche) = 6	Taille (Couche) = 9	Taille (Couche) = 6	Taille (Couche) = 9

5 Conclusion

Plusieurs remarques s'imposent de cette étude de cas dont les résultats sont résumés dans le tableau récapitulatif précédent :

- Les artefacts RUP sont bels et bien exploitables pour mesurer la taille fonctionnelle selon COSMIC-FFP en donnant un résultat plus ou moins satisfaisant.
- Les artefacts RUP générés par les activités de la discipline « Expression des besoins » ne sont pas suffisants pour trouver la taille fonctionnelle réelle du logiciel.
- Les artefacts RUP générés dans les activités de la sous-discipline Analyse de la discipline Analyse et conception sont les mieux placés pour répondre à nos besoins.
- Si on ne tient pas compte des couches, les artefacts d'analyse permettent de trouver le même modèle que celui trouvé manuellement : même processus fonctionnel, même groupe de données, même taille, …
- Si on tient compte des couches, les artefacts d'analyse donnent aussi un résultat intéressant pour deux raisons :
    1. Ils permettent d'aborder la notion de couches dans COSMIC-FFP, qui reste jusqu'à maintenant l'élément le plus délicat à identifier dans COSMIC-FFP.
    2. Ils identifient six Cfsu de plus que la taille identifiée sans tenir compte de couche (les mouvements de données entre les couches). Ces unités de taille fonctionnelle ne sont pas négligeables dans l'effort de développement (si on considère que la taille fonctionnelle est mesurée pour estimer l'effort de développement), au contraire, elles sont les plus coûteuses pour des applications présentant une architecture multicouches client/serveurs ou Web. Par exemple, pour développer une application Web utilisant la technologie Java, ces Cfsu sont développés à l'aide de Servlets par des spécialistes de ce langage, par contre les unités de taille fonctionnelle de lecture et d'écriture dans la deuxième couche sont généralement fournis par le conteneur EJB dans laquelle on déploie notre application.
- Suite à la règle de rapprochement de couches entre RUP et COSMIC-FFP, on constate que le nombre de processus fonctionnels est doublé, dans le cas où on identifie deux couches. Ceci est normal, car même la taille de l'application n'est plus la même.

CHAPITRE 5 : ANALYSE ET EXPLOITATION DES RÉSULTATS

Dans ce travail, on a commencé par présenter les principes et les règles de mesure de COSMIC-FFP selon son manuel de mesure [6]. Puis, on a présenté les concepts clés d'UML vus par les créateurs de ce langage, et on a présenté RUP selon les références [8], [9] et [10] et le site Web de RUP, en étudiant ses deux premières disciplines d'ingénierie à savoir « Expression de besoins » et « Analyse et Conception » pour déterminer les principaux artefacts qu'on a jugé utiles pour générer le modèle COSMIC-FFP. Cette présentation de COSMIC-FFP, d'UML et de RUP selon des sources originales est nécessaire pour notre processus de rapprochement, qui permet d'identifier les éléments de COSMIC-FFP à partir des artefacts RUP définis avec la notation UML, car une mauvaise interprétation d'un concept ou d'une définition faussera le résultat obtenu.

Dans cette section on débutera par redéfinir le processus de mesure COSMIC-FFP en se basant sur les travaux effectués dans les chapitres précédents, puis on présentera une première ébauche d'une redéfinition du processus de Mesure COSMIC-FFP à l'aide du langage UML, après on proposera quelques remarques d'ordre général permettant de bien exploiter les règles de rapprochement, et enfin on présentera les faiblesses de l'étude ainsi qu'une idée sur les futures recherches.

1 Processus de mesure

Les activités de la phase de mappage seront basées sur un rapprochement entre les éléments du modèle COSMIC-FFP et les modèles RUP en considérant que l'entrée n'est pas les FURs mais les artefacts RUP. Le modèle produit sera utilisé par la phase de mesure pour mesurer la taille fonctionnelle.

On considère que les modèles RUP suivants doivent exister **simultanément** pour pouvoir exécuter ce processus et produire un résultat satisfaisant :

- **Modèle de cas d'utilisation** des activités de la discipline Expression de besoins contenant au moins les artefacts suivants:
  - Acteur
  - Cas d'utilisation
  - Digramme de cas d'utilisation

- **Modèle d'analyse** des activités de la discipline Analyse et Conception contenant aux moins les artefact suivants :
  - Couche : Si l'architecture de l'application contient des couches, chaque couche doit être sous forme d'un paquetage d'analyse avec le stéréotype «Layer », elle doit renfermer un ou plusieurs paquetages d'analyse.
  - paquetage d'analyse : un paquetage d'analyse renferme d'autres paquetages d'analyse (de façon récursive), des classes d'analyses et des réalisations – analyse cas d'utilisation.
  - Classe d'analyse : 3 types de classes d'analyse Frontière, Contrôle et Entité. Les classes frontières et entités doivent contenir des attributs de données.
  - Réalisation – analyse cas d'utilisation : chaque cas d'utilisation identifié dans le modèle de cas d'utilisation doit avoir un correspondant dans le modèle d'analyse appelé réalisation – analyse cas d'utilisation avec le stéréotype « use-case realization ». La réalisation – analyse cas d'utilisation doit contenir un diagramme de classes participantes à la réalisation en incluant les acteurs et un ou plusieurs diagrammes d'interaction (Collaboration ou Séquence, et non les deux à la fois) de la réalisation.

Les artefacts de ce modèle seront utilisés dans des entrées pour les activités du processus.

1.1 Phase de mappage

1.1.1 Activité : identification des couches

	Activité : Identification des couches
Entrée	Le modèle d'analyse RUP
Étapes	Une couche dans RUP est un paquetage UML ayant le stéréotype « Layer ». Le nombre de couches est égal au nombre de paquetages UML dans le modèle d'analyse ayant ce stéréotype. S'il existe zéro ou un paquetage, on identifie seulement la couche du système entier.
Résultat	Une ou plusieurs couches COSMIC-FFP.

1.1.2 Activité : identification des frontières

	Activité : Identification des frontières
Entrée	❑ Résultat de l'activité précédente (une ou plusieurs couches), car les frontières ne sont pas matérialisées dans les artefacts RUP. ❑ Diagrammes de cas d'utilisation du modèle de cas d'utilisation ❑ Artefact diagramme d'interaction (collaboration ou de séquence, ou est exclusif) de réalisation – analyse cas d'utilisation.
Étapes	Deux cas se présentent, selon le nombre de couches : ❑ Une seule couche COSMIC-FFP existe : o Une seule frontière existe, celle qui sépare l'environnement des utilisateurs avec celui du système entier. o Les utilisateurs sont les acteurs dans le modèle de cas d'utilisation ❑ Plusieurs couches existent : o La première frontière est celle qui sépare l'environnement des utilisateurs et la première couche. o Les utilisateurs de la première couche sont les acteurs dans le modèle de cas d'utilisation. o La nième frontière est celle qui sépare la nième et (n -1) ième couches o Les utilisateurs de la nième couche sont les acteurs (acteurs de l'environnement utilisateur, classes ou paquetages) qui interagissent avec les classes ou les paquetages de la nième couche dans les diagrammes d'interactions de l'artefact réalisation – analyse des cas d'utilisation de tous les paquetages de la nième couche.
Résultat	Les frontières des couches ainsi que ses utilisateurs.

1.1.3 Activité : identification des processus fonctionnels

	Activité : Identification des processus fonctionnels
Artefact	❑ Réalisation – analyse cas d'utilisation ❑ Diagramme de classes participantes à réalisation des cas d'utilisation.
Étapes	Un processus fonctionnel est une réalisation – analyse de cas d'utilisation dans une couche. Dans RUP, une réalisation prend le stéréotype « use-case realization ». Donc, on identifie par couche tous les cas d'utilisation ayant ce stéréotype, ce sont des processus fonctionnels. Comme dans RUP, un cas d'utilisation se réalise dans plusieurs paquetages de la même couche, on risque de trouver des doubles qu'il faut supprimer. Un événement est matérialisé par la relation entre un acteur et une classe d'analyse dans les diagrammes de classes participantes à la réalisation du cas d'utilisation. Pour la même raison mentionnée ci-dessous, on doit supprimer les doublons.
Résultat	Les processus fonctionnels et leurs événements déclencheurs.

1.1.4 Activité : identification des groupes de données

	Activité : Identification des groupes de données
Artefact	❑ Diagrammes de classes participantes à la réalisation des cas d'utilisation
Étapes	❑ Toutes les classes d'analyse de type entité sont des groupes de données ❑ Les classes frontières qui sont en relation directe avec un acteur dans les diagrammes de classes participantes à la réalisation des cas d'utilisation sont des groupes de données, en absence des classes de type entité dans la même couche. C'est-à-dire, si dans une couche des classes de type entité existent, ce sont des groupes de données et il ne faut pas aller chercher dans les classes frontières, car ces dernières vont contenir certainement des attributs de données en double par rapport aux classes frontières. Cette règle favorise les classes entités par rapport aux classes frontières, car les premières présentent une persistance indéfinie (généralement, elles sont mappées directement à des tables dans la base de données), par contre les deuxièmes ont une persistance courte qui ne survit pas après que le logiciel cesse d'opérer.
Résultat	Les groupes de données

1.1.5 Activité : identification des attributs de données

	Activité : Identification des attributs de données
Artefact	❑ Diagrammes de classes participantes à la réalisation des cas d'utilisation
Étapes	❑ Les attributs des classes entités sont des attributs de données. ❑ Les attributs des classes frontières considérés comme groupes de données (voir section précédente) sont des attributs de données.
Résultat	Les attributs de données

1.2 Activité : Phase de mesure

1.2.1 Activité : identification des sous-processus fonctionnels

	Activité : Identification des sous-processus fonctionnels
Artefact	❑ Diagrammes d'interactions à la réalisation des cas d'utilisation
Étapes	Cette identification est variable selon le niveau de la couche : **a) La première couche** : Dans le cas où on n'identifie pas des couches, la première couche est la couche unique du système entier : ❑ Cas où il existe plus qu'une couche : Les cas d'utilisation de cette couche sont réalisés généralement par l'interaction des acteurs avec les classes frontières. On identifie dans ce cas deux types de sous-processus : o Un sous-processus Entrée (E) qui déplace les attributs de données (de la classe frontière) du côté de l'utilisateur de la frontière logicielle à l'intérieur de la frontière de logiciel. il est matérialisé par un message de l'acteur vers la classe frontière. o Un sous-processus Exit (X) qui déplace les attributs de données (de la classe frontière) de l'intérieur de la frontière logicielle à l'utilisateur. Il est matérialisé par un message de la classe frontière vers une autre classe frontière (si on veut sortir des informations à l'utilisateur, on invoque une classe frontière) ❑ Cas où il existe une seule couche : Les cas d'utilisation de cette couche sont réalisés par l'interaction des acteurs avec les trois classes d'analyse de frontière, de contrôle et d'entité. Dans ce cas, on identifie les deux sous-processus fonctionnels E et X de la même façon que le cas précédent, plus un troisième type : o Un sous-processus fonctionnel de type Lecture (R) ou Écriture (W). Il est matérialisé par un message arrivant à une classe analyse de type entité. **b) les sous-processus fonctionnels de la deuxième couche :** la pièce du logiciel dans cette couche interagit avec les acteurs des couches de plus haut niveau qui sont des acteurs, des paquetages ou des classes d'analyse. Cette couche est appelé couche spécifique au domaine d'affaire et qui contient généralement les classes de type entité et de contrôle. Les cas d'utilisation sont réalisés dans cette couche par les interactions des acteurs avec les classes de la couche. On identifie 3 types de sous-processus fonctionnels : o Un sous-processus Entrée (E) matérialisé par un message de l'acteur vers une classe de la couche. o Un sous-processus Exit (X) matérialisé par un message d'une classe de la couche vers l'utilisateur. o Un sous-processus fonctionnel de type Lecture (R) ou Écriture (W). Il est matérialisé par un message arrivant à une classe analyse de type entité et qui n'est pas un message de type Entrée (E). c'est-à-dire l'émetteur de message ne doit pas être un Acteur. **c) les sous-processus fonctionnels des autres couches :** s'il existe d'autres couches, elles sont généralement semblables à la deuxième couche. Les artefacts du modèle d'analyse ne nous permettent pas de différencier entre les sous-processus de lecture (R) et d'écriture (W). L'unique solution consiste à créer deux nouveaux stéréotypes pour les messages : Read pour la lecture et Write pour l'écriture.
Résultat	Les sous-processus fonctionnels

1.2.2 Activité : Application de la fonction de mesure

Un sous-processus identifié dans la section précédente correspond à 1 Cfsu.

1.2.3 Activité : agrégation des résultats de mesure

Puisqu'on ne distingue pas entre les sous-processus fonctionnels Lecture (R) et Écriture (W), la fonction d'agrégation devient :

$$\text{Taille}_{\text{Cfsu}}(\text{couche}_i) = \Sigma\ \text{taille}(\text{entrées}_i) + \Sigma\ \text{taille}(\text{exits}_i) + \Sigma\ \text{taille}(\text{Lectures}_i \text{ ou } \text{Écritures}_i).$$

2 Processus de mesure COSMIC-FFP basé sur UML

Le processus de mesure actuel de COSMIC-FFP est présenté de façon informelle. La procédure de rapprochement entre les artefacts de RUP et le modèle de COSMIC-FFP m'a inspiré une idée de formalisation de ce processus à l'aide d'UML, ceci à pour avantage de présenter des règles de mesure formelles à l'aide d'un langage qui s'impose comme un standard de modélisation. Ci-dessous une première ébauche d'un travail qui nécessite plus de réflexion.

2.1 Hypothèses

Ces hypothèses peuvent être remplacées par des définitions dans une étude plus détaillée de ce processus. Elles consistent à établir des correspondances entre des artefacts RUP développés avec la notation UML et les éléments du modèle COSMIC-FFP.

- un processus fonctionnel est un cas d'utilisation
- un groupe de données est une classe d'analyse de type entité
- un attribut de données est un attribut dans la classe d'analyse de type entité
- un utilisateur est un acteur UML
- Une couche contient un ou plusieurs cas d'utilisation

- Un cas d'utilisation se réalise par des classes de type frontière et entité, et d'interaction entre elles et les acteurs. Des diagrammes de séquence ou de collaboration représentent ces interactions.

2.2 Identification des couches

On utilise le même patron de couches utilisées dans RUP, la seule différence consiste en ce qu'une couche contient un ou plusieurs cas d'utilisation, et non pas un cas d'utilisation se réalise dans une ou plusieurs couches.

On doit produire un diagramme de cas d'utilisation pour chaque couche. Les acteurs sont ceux de l'environnement utilisateurs ou bien les cas d'utilisation, c'est-à-dire si un cas d'utilisation de la première couche utilise un cas d'utilisation de la deuxième couche, la première est considérée un acteur, donc un utilisateur COSMIC-FFP de la deuxième couche. Ce qui signifie qu'un processus fonctionnel d'une couche est aussi un utilisateur d'une autre couche. Ceci correspond bien à la définition des couches et des processus fonctionnels dans RUP.

2.3 Identification des frontières

Il est difficile de trouver l'équivalence de l'aspect frontière de COSMIC-FFP dans UML. Ce concept peut ne pas être matérialisé, mais il est facilement distinguable avec les couches et les utilisateurs.

Les utilisateurs sont les acteurs dans les diagrammes de cas d'utilisation (il existe un diagramme par couche).

2.4 Identification des processus fonctionnels

Un cas d'utilisation est un processus fonctionnel.

Une association entre un acteur et un cas d'utilisation dans les diagrammes de cas d'utilisation est un événement. On crée un stéréotype « Event » pour ces associations.

2.5 Identification des groupes et des attributs de données

Une classe d'analyse de type entité est un groupe de données, leurs attributs sont des attributs de données.

2.6 Identification des sous-processus fonctionnels

Chaque cas d'utilisation doit être représenté par un ou plusieurs diagrammes d'interaction. Les messages sont des sous-processus fonctionnels. On crée quatre stéréotypes de messages :

- « Entry » pour un sous-processus fonctionnel de type Entrée.
- « Exit » pour un sous-processus fonctionnel de type Exit.
- « Write » pour un sous-processus fonctionnel de type Écriture.
- « Read » pour un sous-processus fonctionnel de type Lecture.

Le sous-processus fonctionnel est identifié par le stéréotype dans le message.

3 Remarques

UML seul ne permet pas de retrouver le modèle COSMIC-FFP

On a commencé au départ dans la présentation du travail (Section 1.4.2) par dire que UML seul ne permet pas de définir un processus de rapprochement complet, permettant d'identifier tous les éléments du modèle COSMIC-FFP. Cette idée apparaît claire maintenant pour plusieurs raisons :

- Les couches ne peuvent pas être identifiées sans le recours à la notion de couches dans RUP et non dans UML. Mais on utilise UML pour les représenter à l'aide de paquetage et de stéréotype.
- Une classe UML peut être une classe d'analyse ou de conception. Or, on a remarqué qu'une classe de conception ne peut être équivalente à un groupe de données, car elle renferme des attributs qui sont spécifiques à l'implémentation, et qui ne sont pas des attributs de données. Une classe d'analyse de type entité ne contient que des attributs de données, car elle est encore liée au domaine fonctionnel des utilisateurs. Cette classe est identifiée grâce à RUP avec la notation de UML.
- Les réalisations des cas d'utilisation sont des processus fonctionnels et non pas les cas d'utilisation décrits dans le diagramme de cas d'utilisation. Ces réalisations sont dans RUP avec un nouveau stéréotype de UML.
- UML, sans le processus RUP, peut être une source pour le modèle COSMIC-FFP, mais on doit imposer plusieurs contraintes sur ses éléments (classes, attributs, diagrammes, …) ou sur le processus de rapprochement, ce qui complique énormément la tâche d'arrimage en influençant la qualité du résultat obtenu.

RUP ou un autre processus générant ses artefacts à l'aide d'UML

Ce travail est basé essentiellement sur les artefacts générés par le processus RUP selon la notation UML. Mais, il peut être n'importe quel autre processus à condition qu'il génère les mêmes artefacts. Maintenant RUP peut être reconfiguré avec des plug-in pour s'adapter à un environnement d'une entreprise ou à une technologie spécifique. Cette reconfiguration ne peut pas causer de problèmes pour notre processus de rapprochement, car généralement, les deux modèles de cas d'utilisation et d'analyse sont nécessaires pour toutes les activités de développement.

Avantages du modèle d'analyse

Bien que RUP propose des guidelines pour la production des artefacts, la qualité et le niveau de détail de ces artefacts dépendent toujours de la vision faite par l'équipe de développement à l'application à développer. Deux équipes différentes ne produisent pas nécessairement des artefacts semblables. Il existe toujours des nuances, mais le résultat (le produit à développer) fournit les mêmes fonctionnalités. Cette distinction s'accentue à partir du modèle de conception, car chaque équipe propose une solution à un modèle d'analyse qui est généralement unique. Les équipes, dans les activités d'expression des besoins et d'analyse, sont encore près des utilisateurs, et elles sont influencées par leurs besoins en produisant des artefacts plus ou moins semblables. Ceci représente un avantage aux activités de rapprochement, car le premier objectif de l'automatisation du processus de mesure est de réduire la dépendance entre le résultat de mesure et le mesureur.

Les meilleurs contenus des artefacts pour le rapprochement

Un meilleur contenu d'un artefact est celui produit par un développeur qui connaît les principes et les règles de mesure de COSMIC-FFP. Les artefacts de cas d'utilisation et du diagramme d'interactions doivent être taillés selon les besoins de COSMIC-FFP. Ceci ne change rien pour les modèles de cas d'utilisation et d'analyse de RUP, mais il apporte beaucoup d'avantages aux activités de rapprochement. Le développeur doit connaître au moins qu'une réalisation d'un cas d'utilisation correspond à un processus fonctionnel et qu'un message dans un diagramme de collaboration correspond à un sous-processus fonctionnel.

Bénéfices pour COSMIC-FFP

Pratiquement les experts de mesure COSMIC-FFP n'utilisent pas seulement les FURs comme elles sont formulées par les utilisateurs pour construire le modèle de COSMIC-FFP, mais ils font

recours aux artefacts de développement si l'application à mesurer est déjà développée. Ceci est intéressant car les aspects de COSMIC-FFP apparaissent plus clairs avec ces artefacts. Cette étude a renforcé cette démarche, surtout lorsqu'on associe la construction du modèle COSMIC-FFP aux artefacts d'un processus donné. La notion de couches devient plus claire si on l'associe à celui de RUP.

Bénéfices pour RUP

RUP ne possède pas actuellement une méthode d'estimation de la taille fonctionnelle du logiciel à développer, pourtant elle est importante, car elle sera une source fiable pour la discipline de la gestion du projet qui se focalise sur la planification et l'estimation de l'effort de développement. Le résultat de cette étude permet d'intégrer une activité de mesure dans les différentes disciplines de RUP et ceci est facilement réalisable car, et comme toute activité de RUP, elle prend en entrée les résultats des autres activités et crée des artefacts qui seront exploités par d'autres activités.

4 Les faiblesses de cette étude

- La non identification des couches de middleware et de système logiciels, car ces derniers dépendent du modèle de déploiement de l'activité de conception qu'on a trouvé très liés à l'implémentation de l'application.
- Ce travail se limite au logiciel de type MIS (management information system), les systèmes embarqués ou temps-réel ne suivent peut-être pas les mêmes règles de rapprochement.
- Limitation de l'étude à un processus de développement bien particulier, malgré qu'il est une référence dans le domaine de génie logiciel et que tous les autres processus s'inspirent de lui.
- La première discipline de RUP Modélisation d'affaire (Business Modeling) n'est pas explorée dans cette étude. Elle pourrait, peut-être, donner un résultat satisfaisant.

5 Futures recherches

- Effectuer plusieurs études de cas pour vérifier les règles de rapprochement définies précédemment.
- Généraliser les règles de rapprochement pour n'importe quel processus de développement.

- Généraliser les règles de rapprochement pour n'importe quel type de système (MIS, Temps-réel, …)
- Approfondir l'étude du processus de mesure à l'aide du langage UML décrit dans la section 2 de ce chapitre.
- Faire le même processus de rapprochement en générant des éléments UML de COSMIC-FFP et en utilisant le résultat du processus précédent.
- Définir une activité de mesure de la taille fonctionnelle avec COSMIC-FFP dans les disciplines de RUP qui prend comme en entrée les artefacts des activités courantes et sortie le modèle COSMIC-FFP, en se basant sur les règles de rapprochement définies dans cette étude.

6 Conclusion

Il faut mettre en application cette étude théorique par le développement d'un outil qui prend en charge les règles définies dans le rapprochement pour automatiser la génération automatique du modèle COSMIC-FFP, c'est le sujet de la deuxième partie de ce travail.

CHAPITRE 6 : DÉVELOPPEMENT DE L'OUTIL CFFP-GAUGE

1. Énoncé de positionnement du produit

Pour	Les spécialistes de mesure avec COSMIC-FFP et les développeurs selon un processus donné.
Qui	Visualisent et mesurent la taille fonctionnelle d'un logiciel
Le produit	CFFP-GAUGE permet de visualiser un modèle COSMIC-FFP à travers une interface utilisateur et générer automatiquement ce modèle en exploitant les artefacts produits dans les différentes activités d'un processus de développement.
Qui	❑ Gère le modèle COSMIC-FFP par sa visualisation à travers des interfaces utilisateurs. ❑ Exécute automatiquement les deux phases COSMIC-FFP, de mappage pour créer le modèle, et de mesure pour calculer la taille, en utilisant les artefacts UML produits dans un processus de développement par des outils de visualisation tel que Rose ou Together.
Comparativement à	l'exécution manuelle de ces tâches.
Notre produit	est plus rapide, plus précis et moins coûteux et applique de façon standard les règles et les principes de COSMIC-FFP à tous les logiciels à mesurer.

2. Besoins clés des utilisateurs

Besoins	**Priorité**	**Préoccupations**	**Solution actuelle**	**Solution proposée**
Créer une interface graphique utilisateur qui prend en charge toutes les étapes de mesure de COSMIC-FFP.	Élevée	Aider un mesureur de visualiser sur un écran le modèle de COSMIC-FFP en affichant les éléments nécessaires pour effectuer la mesure.	Consultation du manuel du modèle de mesure.	Créer une interface GUI qui affiche le modèle et permet à l'utilisateur de le modifier en ajoutant et en supprimant ses éléments tout en respectant les principes COSMIC-FFP.
Possibilité de sauvegarder le modèle COSMIC-FFP créé pour chaque mesure dans un format persistant.	Élevée	Créer une base de tous les modèles de mesure pour des fins ultérieures.	Reporter manuellement les résultats de mesure.	Sauvegarder, à la demande, chaque modèle de mesure dans une base des données ou sous un format XML.
Automatiser la mesure de la taille fonctionnelle d'un logiciel en utilisant les artefacts élaborés lors d'un processus de développement.	Élevée	Standardiser l'application des règles de mesure en créant un outil qui les utilise de façon unique à tous les types de logiciel, car actuellement l'application manuelle de ces règles est au mesureur.	Recours au service d'un expert ayant de l'expérience dans le type de logiciel concerné.	exécuter automatiquement les deux phases de mappage et de mesure COSMIC-FFP, pour générer respectivement le modèle COSMCI-FFP et la taille fonctionnelle, en utilisant comme entrée les artefacts UML élaborés dans les activités d'un processus de développement.

Besoins	Priorité	Préoccupations	Solution actuelle	Solution proposée
Effectuer plusieurs mesures automatiques pour un même logiciel au fur et à mesure de l'avancement du processus de développement	Élevé	Permettre au mesureur de réajuster la taille du logiciel au fur à mesure que les développeurs avancent dans le projet.	Mise à jour manuelle de la taille calculée initialement en se référant aux documents de changement des exigences.	L'outil à développer doit générer, à la demande, plusieurs versions du modèle COSMIC_FFP d'un même logiciel selon les phases, les activités et les itérations du processus de développement.

3. Résumé des capacités

Le tableau suivant dresse les bénéfices tirés par les utilisateurs et les caractéristiques supportées par CFFP-GAUGE.

Bénéfices des utilisateurs	Caractéristiques supportées
Le mesureur trouvera un outil d'aide qui lui permet de créer et de visualiser les modèles COSMIC-FFP pour mesurer la taille fonctionnelle d'un logiciel.	À la demande de l'utilisateur l'outil affichera, à l'aide d'une interface graphique, un modèle pré-contsruit standard et l'utilisateur n'a qu'à modifier le modèle pour l'ajuster en fonction du logiciel à mesurer. Le calcul de la taille s'effectuera automatiquement en fonction du nombre des mouvements des données dans le modèle.
Le mesureur formera une base des modèles COSMIC-FFP construits pour tous les logiciels mesurés, cette base servira ultérieurement pour des études empiriques.	Chaque modèle construit doit être conservé sous un format persistant (base de données ou XML) et il peut être retrouvé à la demande.
Pour les logiciels à mesurer possédant des artefacts UML élaborés par un processus de développement, la création du modèle COSMIC-FFP s'effectue automatiquement. Cette procédure possède les avantages suivants : • Une application standardisée des règles et des concepts de la méthode de mesure. • Plus de précision, car généralement les artefacts de développement contiennent plus de détails par rapport aux documents destinés à la procédure de mesure. • Moins d'effort humain, ce qui engendre moins de coût supplémentaire destiné à l'effort de mesure.	L'outil utilisera les artefacts UML élaborés dans un processus de développement en utilisant des outils de visualisation tels que Rational Rose, pour exécuter les deux phases de mappage et de mesure permettant de construire automatiquement le modèle COSMIC-FFP du logiciel à mesurer, et par conséquent calculer sa taille fonctionnelle.
Une estimation de la taille d'un logiciel dès la première phase de développement est bénéfique pour le gestionnaire du projet car elle permet d'évaluer, dès le départ, l'effort de développement et par conséquent une meilleure gestion des ressources du projet.	L'outil génèrera (à la demande) un modèle COSMIC-FFP en utilisant les modèles UML élaborés des activités de la première phase de développement. Les phases de spécification des exigences et d'analyse produisent des artefacts considérés comme une base solide pour estimer la taille réelle du logiciel.

Bénéfices des utilisateurs	**Caractéristiques supportées**
La taille du logiciel peut être réajustée automatiquement au fur à mesure de l'avancement du processus de développement, ce qui permet de comparer la mesure par rapport à l'estimation de la taille du logiciel faite en début de projet et d'utiliser ce résultat pour aider les gestionnaires à ajuster la planification du projet.	Les processus contemporains de développement de logiciel sont généralement itératifs et incrémentaux. L'outil doit être appelé à la fin de chaque itération pour exploiter les modèles de cette activité, réajuster la taille du logiciel, et par conséquent utiliser le résultat pour planifier les itérations suivantes.
Les développeurs selon un processus auront un outil de plus qui permet de mesurer automatiquement la taille fonctionnelle et aider, par conséquent, les gestionnaires du projet à bien planifier les activités du processus.	L'outil doit s'intégrer avec les autres outils utilisés par le processus de développement pour l'élaboration des artefacts UML.

4. Activités de développement

Il est difficile de présenter dans ce document toutes les activités de développement de CFFP-GAUGE ainsi que les artefacts produits. On se contentera des plus importantes qui refléteront les besoins directs des utilisateurs, ainsi que l'application des règles de rapprochement identifiées dans le chapitre précédent.

4.1 Formalisation des besoins

Les besoins des utilisateurs formulés dans les sections précédentes peuvent être formalisés par le diagramme de cas d'utilisation de la *figure 6.1*.

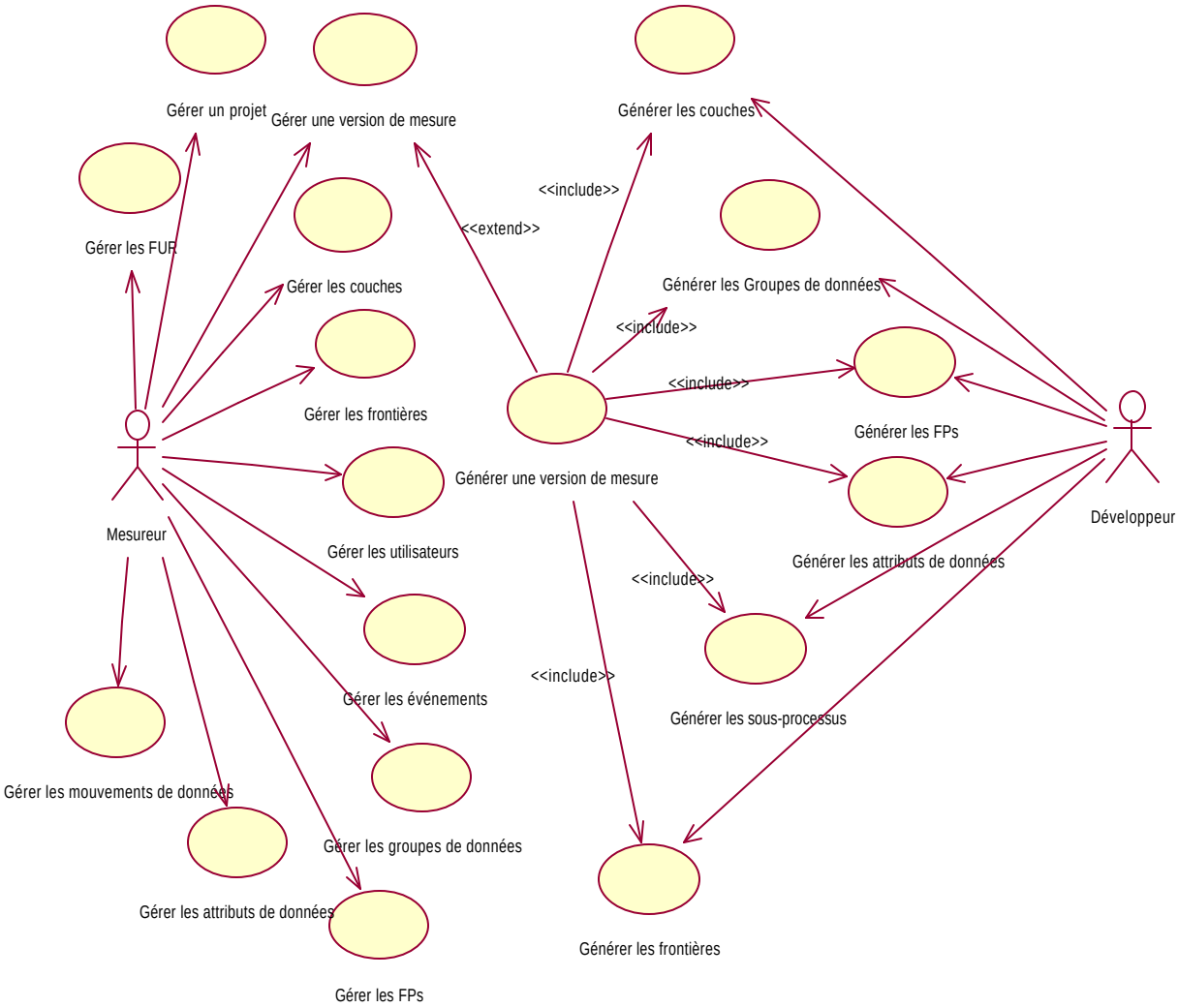

Figure 6.1 : diagramme de cas d'utilisation de CFFP-GAUGE

		Description
Acteurs	Mesureur	C'est un utilisateur qui gère un modèle COSMIC-FFP : crée, modifie, visualise, sauvegarde, … les éléments qui forment le modèle COSMIC-FFP. Il peut aussi demander la génération de ces éléments à partir des artefacts d'un processus de développement.
	Développeur	C'est un utilisateur qui, en présentant des artefacts produits dans des activités d'un processus de développement, demande au logiciel d'exécuter les phases de mesure pour générer un ou plusieurs éléments du modèle COSMIC-FFP.
Cas d'utilisation	Gérer un projet	Permet de créer, modifier, sauvegarder sous un format persistant et retrouver (à partir d'un format persistant) un modèle COSMIC-FFP spécifique à une application développée dans un projet logiciel.
	Gérer une version de mesure	Permet de créer, modifier, supprimer ou consulter une version de mesure. Une version de mesure correspond à un modèle COSMIC-FFP qui reflète la taille fonctionnelle d'un logiciel mesuré à un instant donné. Elle peut correspondre à un modèle généré à partir des artefacts résultat d'une ou plusieurs activités d'une itération dans un processus de développement.
	Gérer les FURs	Permet de créer, modifier, supprimer ou consulter les FURs.
	Gérer les couches	Permet de créer, modifier, supprimer ou consulter les couches.
	Gérer les frontières	Permet de créer, modifier, supprimer ou consulter les frontières
	Gérer les utilisateurs	Permet de créer, modifier, supprimer ou consulter les utilisateurs
	Gérer les processus fonctionnels	Permet de créer, modifier, supprimer ou consulter les processus fonctionnels
	Gérer les événements	Permet de créer, modifier, supprimer ou consulter les événements.
	Gérer les groupes de données	Permet de créer, modifier, supprimer ou consulter les groupes de données
	Gérer les attributs de données	Permet de créer, modifier, supprimer ou consulter les attributs de données
	Gérer les mouvements de données	Permet de créer, modifier, supprimer ou consulter les attributs de données et mettre à jour, par conséquent, la taille fonctionnelle des couches.
	Générer une version de mesure	Permet de générer automatiquement un modèle COSMIC-FFP à partir des artefacts UML produit par une ou plusieurs activités d'un processus de développement.
	Générer les couches	Permet de générer les couches COSMIC-FFP à partir d'un ou plusieurs artefacts UML du processus de développement. Il correspond à l'activité Identification des couches de la phase de mappage du processus de mesure COSMIC-FFP.

		Description
	Générer les frontières	Permet de générer les frontières et les utilisateurs COSMIC-FFP à partir d'un ou plusieurs artefacts UML du processus de développement. Il correspond à l'activité Identification des frontières de la phase de mappage du processus de mesure COSMIC-FFP.
	Générer les processus fonctionnels	Permet de générer les processus fonctionnels et leurs événements déclencheurs COSMIC-FFP à partir d'un ou plusieurs artefacts UML du processus de développement. Il correspond à l'activité Identification des processus fonctionnels de la phase de mappage du processus de mesure COSMIC-FFP.
	Générer les groupes de données	Permet de générer les groupes de données COSMIC-FFP à partir d'un ou plusieurs artefacts UML du processus de développement. Il correspond à l'activité Identification des groupes de données de la phase de mappage du processus de mesure COSMIC-FFP.
	Générer les attributs de données	Permet de générer les attributs de données COSMIC-FFP à partir d'un ou plusieurs artefacts UML du processus de développement. Il correspond à l'activité Identification des attributs de données de la phase de mappage du processus de mesure COSMIC-FFP.
	Générer les sous-processus	Permet de générer les mouvements de données COSMIC-FFP à partir d'un ou plusieurs artefacts UML du processus de développement. Il correspond à toutes les activités de la phase de mesure du processus de mesure COSMIC-FFP (identification de sous-processus, application de la fonction mesure et agrégation des résultats de mesure).

On distingue, selon le modèle de cas d'utilisation et la description des cas d'utilisation, deux grandes parties :

- Gestion du modèle COSMIC-FFP qui consiste à créer, modifier, consulter et supprimer les éléments d'un modèle correspondant à une version de mesure pour un projet logiciel à mesurer.
- Génération des éléments d'un modèle COSMIC-FFP à partir des artefacts UML produits dans un processus de développement.

4.2 Architecture de l'application

Une première version de CFFP-GAUGE sera développée en tenant compte des conditions suivantes :

- Considérer que le processus de développement qui génère les artefacts UML sera RUP. Ceci nous permet d'appliquer les règles de rapprochement entre les artefacts RUP et le modèle COSMIC-FFP identifiées dans le chapitre 5.
- Considérer que les artefacts RUP seront élaborés par Rational Rose et que les modèles produits seront exploités par son API Java2REI.

- CFFP-GAUGE sera développé avec la technologie Java et que la gestion de persistance de données utilisera XML.
- Qu'à la demande, tout le modèle COSMIC-FFP se génère d'un seul coup, et qu'il n y aura pas de génération individuelle des éléments du modèle.

4.2.1 Schéma général de CFFP-GAUGE

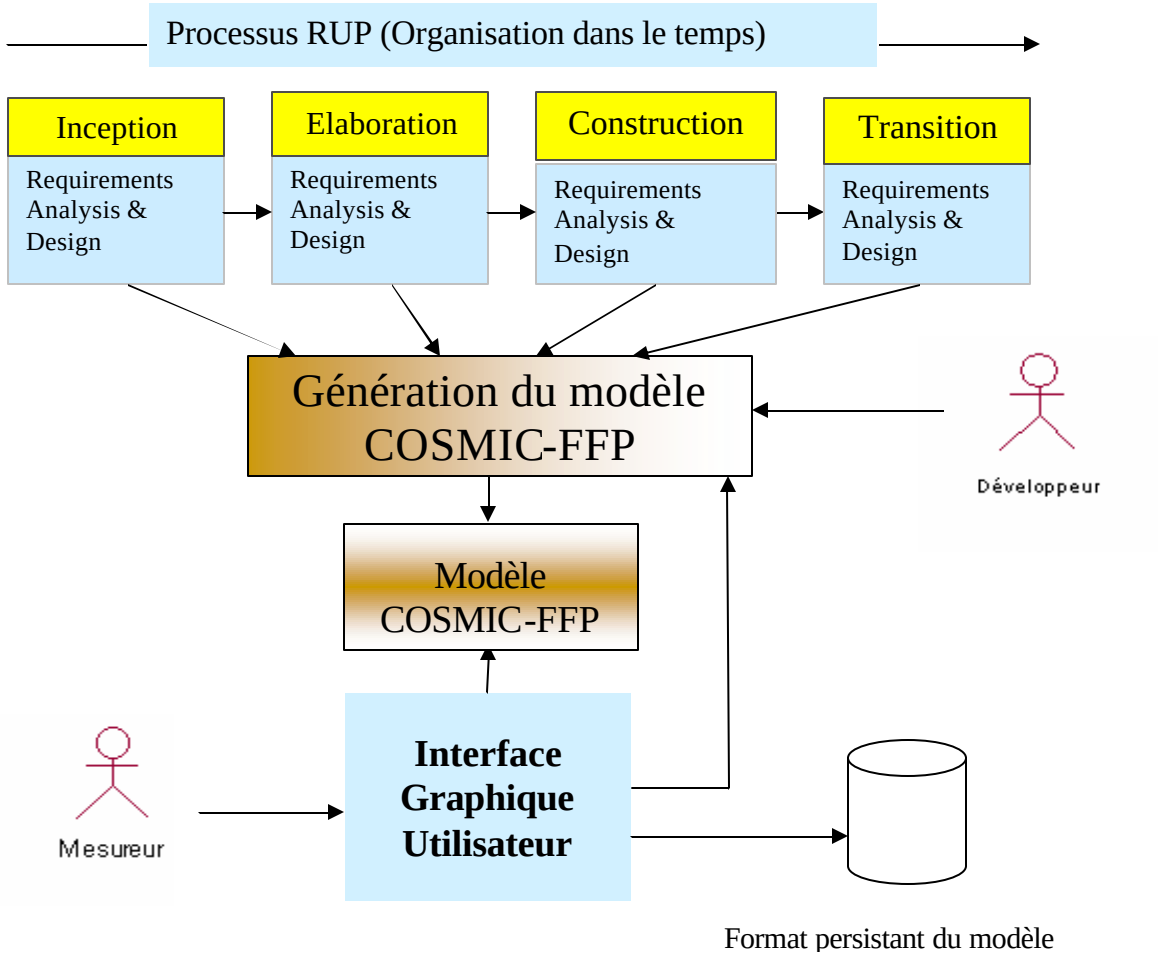

Figure 6.2 : Schéma général de l'application

4.2.2 Architecture en couche de CFFP-GAUGE

On distingue principalement 3 couches (*Figure 6.3*)

Couche de présentation : elle contient un seul paquetage contenant toutes les classes frontières réalisant l'interface utilisateur GUI (com.cosmicffp.gui).

Couche du domaine d'affaire : elle contient trois paquetages :

- Paquetage du modèle d'affaire (com.cosmicffp.bmdl) : il contient le modèle d'affaire décrit dans la *figure 6.4* et qui interagit avec les paquetages de génération et de la gestion de persistance.
- Paquetage de génération (com.cosmicffp.generation) : il interagit avec l'API Java2REI pour générer le modèle COSMIC-FFP.
- Paquetage de la gestion de persistance (com.cosmicffp.xml): il permet de sauvegarder en format XML le modèle COSMIC-FFP et de le retrouver. Il interagit avec le paquetage org.exolab.castor d'apache.

Couche de middleware : elle contient les paquetages fournis que ce soit par Java pour le développement, Rational Rose avec Java2REI qui permet d'extraire les artefacts RUP ou d'Apache tel que Castor pour transformer un modèle objet sous format XML ou parser un fichier XML.

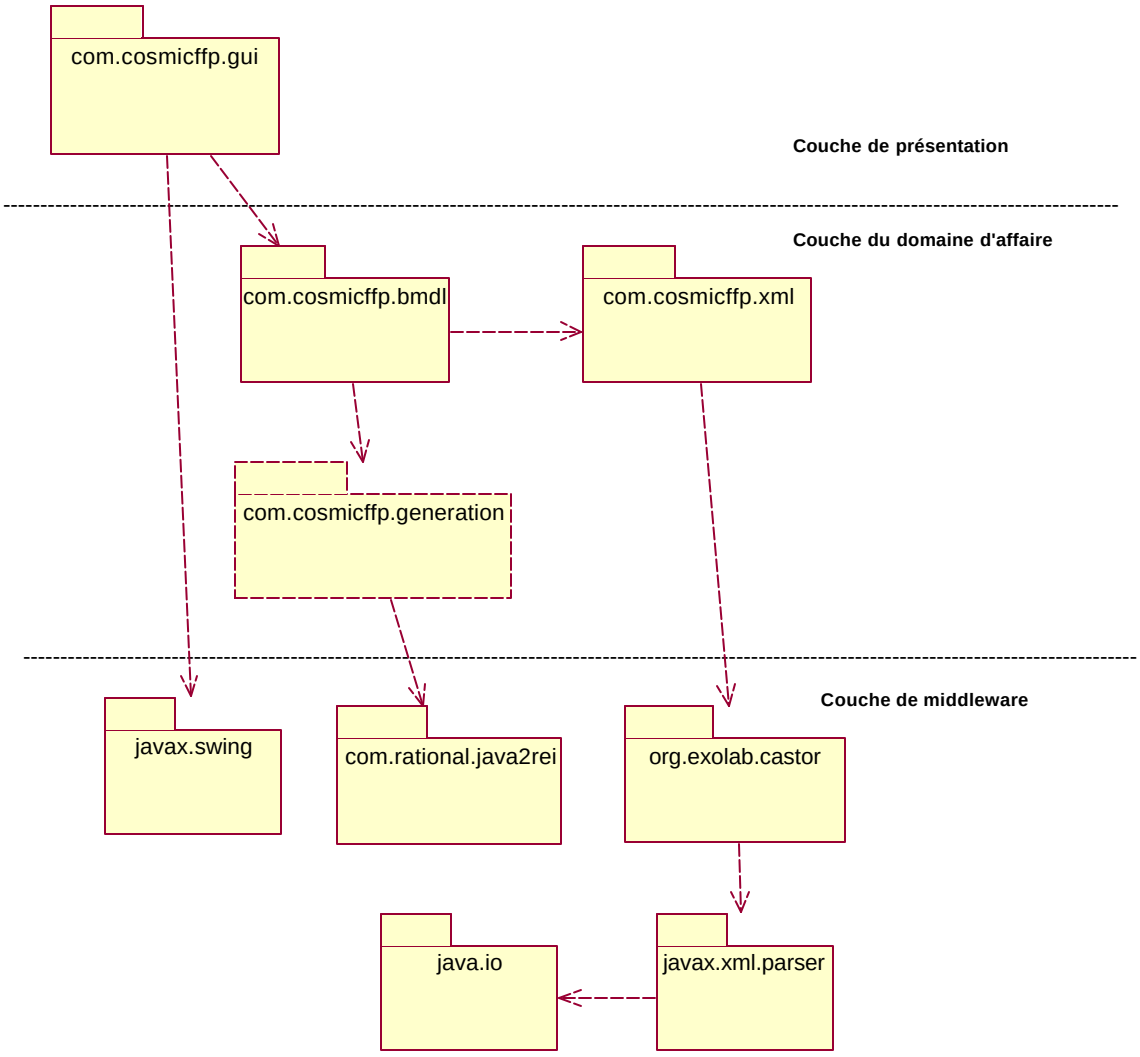

Figure 6.3 : Architecture en couche de CFFP-GAUGE

4.2.3 Le modèle d'affaire CFFP-GAUGE

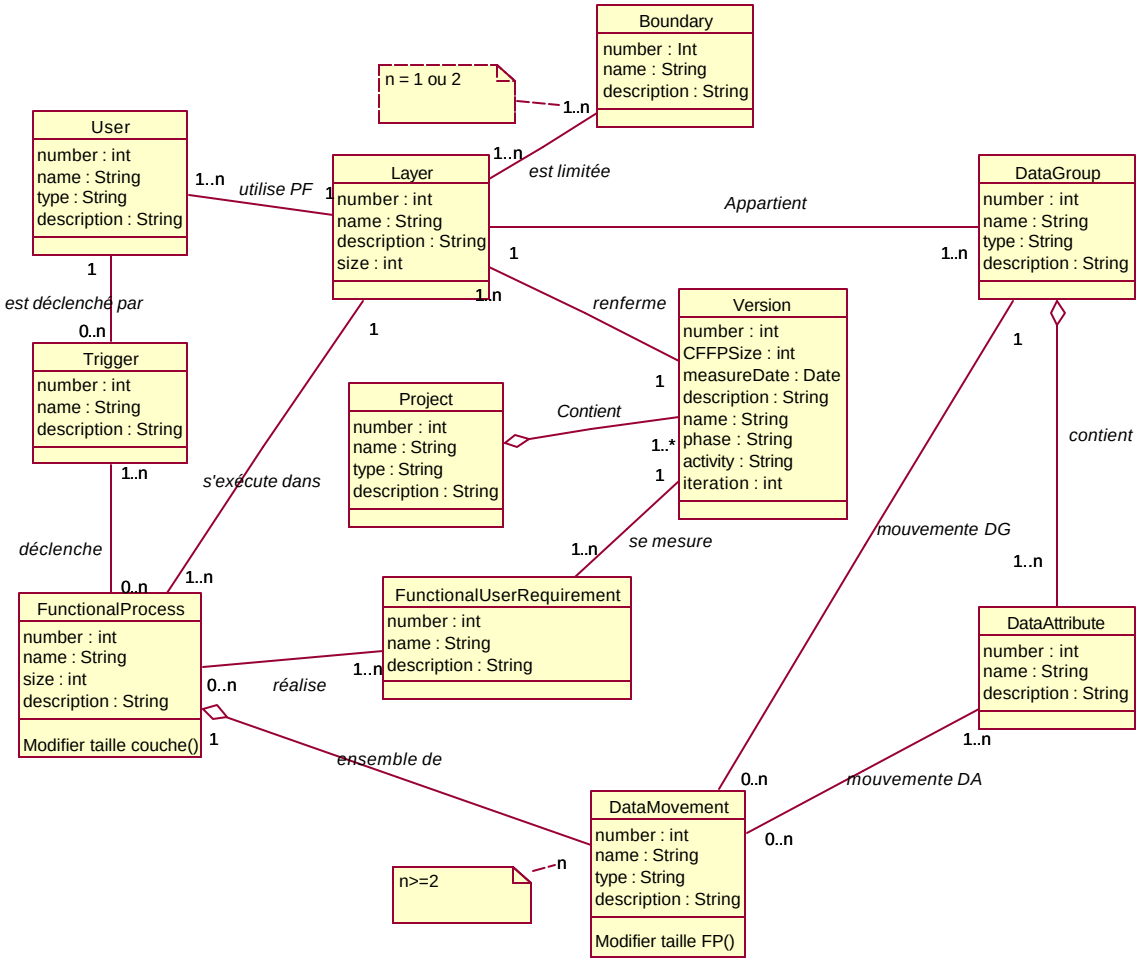

Figure 6.4 : Modèle d'affaire CFFP-GAUGE

Un projet est un logiciel à mesurer qui contient plusieurs versions de mesure. Chaque version de mesure correspond à un modèle COSMIC-FFP mappé à partir des exigences fonctionnelles utilisateurs (FURs). Un modèle COSMIC-FFP contient une ou plusieurs couches logicielles. Chaque couche logicielle est limitée par une ou deux frontières au delà de laquelle un ou plusieurs utilisateurs interagissent avec la pièce de logiciel y contenue, des groupes de données et des processus fonctionnels déclenchés par des triggers à travers les utilisateurs. Un processus

fonctionnel contient deux ou plusieurs mouvements de données et les groupes de données contiennent des attributs de données.

4.2.3 Gestion de la persistance

Pour un besoin de portabilité, on a choisi XML pour la gestion de la persistance de données en utilisant Castor d'Apache. Castor est un framework de gestion de données XML qui permet de mapper une instance d'un schéma XML à un modèle objet représentant ces données. Le modèle objet inclut un ensemble des classes et des types ainsi que des descripteurs permettant d'obtenir des informations au sujet de classes et ses champs. Ces descripteurs permettent de sauvegarder un modèle objet sous un format XML (Marshalling) et de créer un modèle objet Java à partir d'un modèle XML (Unmarshalling). Castor utilise un schéma XML pour créer les classes Java au lieu de l'utilisation de DTD, car un schéma XSD est plus robuste et plus flexible et possède plusieurs avantages car lui-même est un document XML. Le schéma XSD du modèle COSMIC-FFP est décrit dans *l'annexe A*.

5. Fonctionnalités de CFFP-GAUGE

CFPP-GAUGE contient deux fonctions principales : présentation du modèle COSMIC-FFP sous un format graphique conviviale et génération automatique du modèle COSMIC-FFP à partir des artefacts RUP élaborés avec Rational Rose.

5.1 L'interface GUI de CFFP-GAUGE

Le modèle COSMIC-FFP est représenté sous forme d'une arborescence qui met en relief tous les éléments COSMIC-FFP ainsi que leurs relations. Dès le lancement, CFFP-GAUGE crée un projet « Untitled » ayant une version de mesure « Version1 » qui contient tous les éléments de COSMIC-FFP (*Figure 6.5*). Ceci permet à l'utilisateur d'avoir une idée globale sur le modèle. L'utilisateur peut charger un modèle déjà sauvegardé sous un format XML, ou bien modifier le projet existant et le sauvegarder. Tous les éléments de COSMIC-FFP (FUR, Couche, FP, …) sont modifiables. On peut ajouter, supprimer ou modifier un élément dans des fenêtres spécifiques, tout en respectant les règles et les principes de COSMIC-FFP. Cette interface peut être enrichie (dans une version ultérieure) par un outil qui permet de schématiser graphiquement le modèle.

5.2 Génération automatique du modèle COSMIC-FFP

Un projet de mesure contient plusieurs versions de mesure. Chaque version correspond à un modèle COSMIC-FFP. Une version peut être créée manuellement ou générée automatiquement en spécifiant un fichier « .mdl » de Rational Rose (*Figure 6.6*). La version de mesure à générer est spécifique à une phase, une discipline et une itération RUP. Aucune contrainte n'est imposée

sur le modèle Rational Rose, on exploite le modèle comme il est fait pour le développement de l'application. On demande seulement le nom du paquetage à partir duquel on fait notre génération en appliquant les règles de rapprochement identifiées dans le chapitre précédent. On génère le modèle COSMIC-FFP à partir des artefacts des activités des disciplines d'expression des besoins (Requirements) ou d'analyse (Analysis). Si un élément UML dans Rose contient une documentation, celle-ci sera copiée à l'élément correspondant dans le modèle généré de COSMIC-FFP. La génération se déroule tout en affichant un fichier «Log» contenant des messages, spécifiant le succès ou l'échec et sa cause, de chaque étape franchie de la procédure d'identification des éléments du modèle COSMIC-FFP. En cas de succès, une nouvelle version de mesure sera affichée à l'écran, grâce à l'interface GUI, et l'utilisateur peut apporter des modifications sur le modèle, comme il peut le supprimer ou le sauvegarder. Plusieurs versions de mesure peuvent être générées selon le besoin pour réajuster la taille fonctionnelle à cause d'une modification des artefacts de développement du logiciel à mesurer.

Le résultat obtenu est très satisfaisant, la figure 6.7 montre le résultat de la version de mesure suite aux activités de la discipline d'expression des besoins (Requirements) de cas Gestion Glossaire du chapitre 4; et la figure 6.8 montre la version de mesure à partir des artefacts des activités de l'analyse.

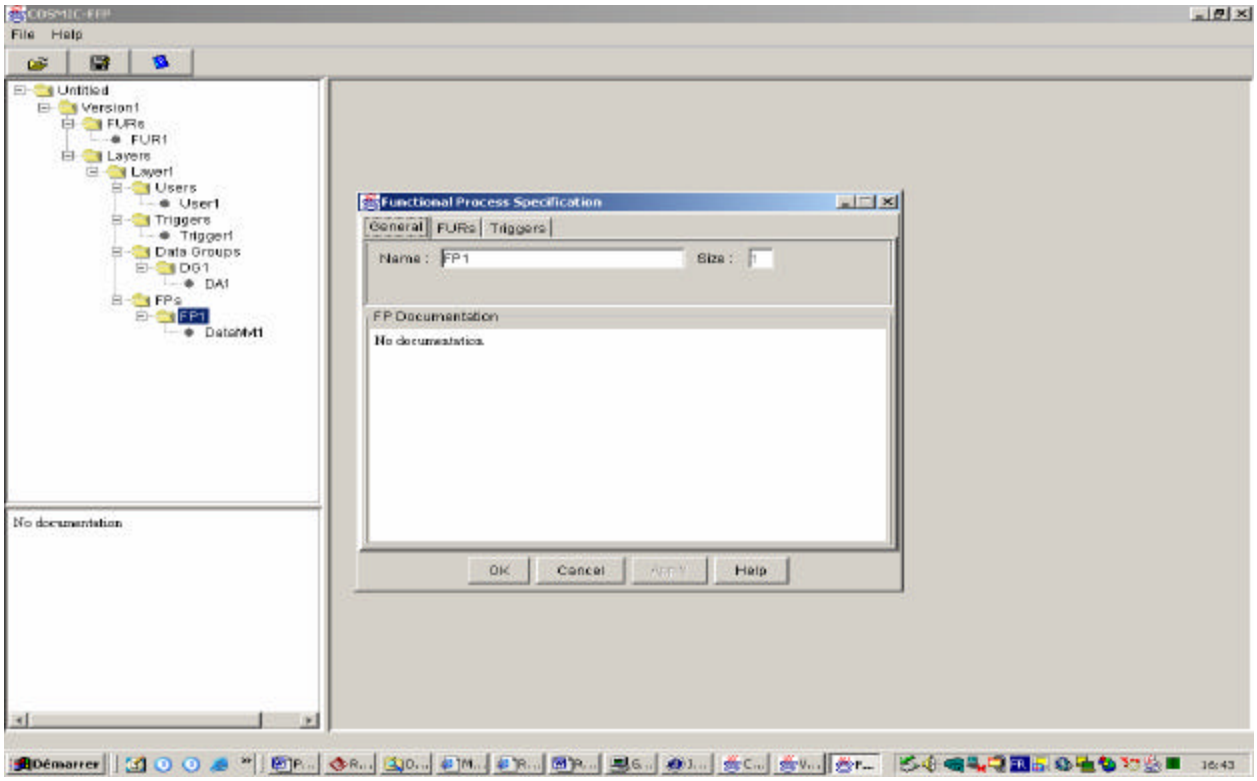

Figure 6.5 l'interface GUI de COSMIC-FFP

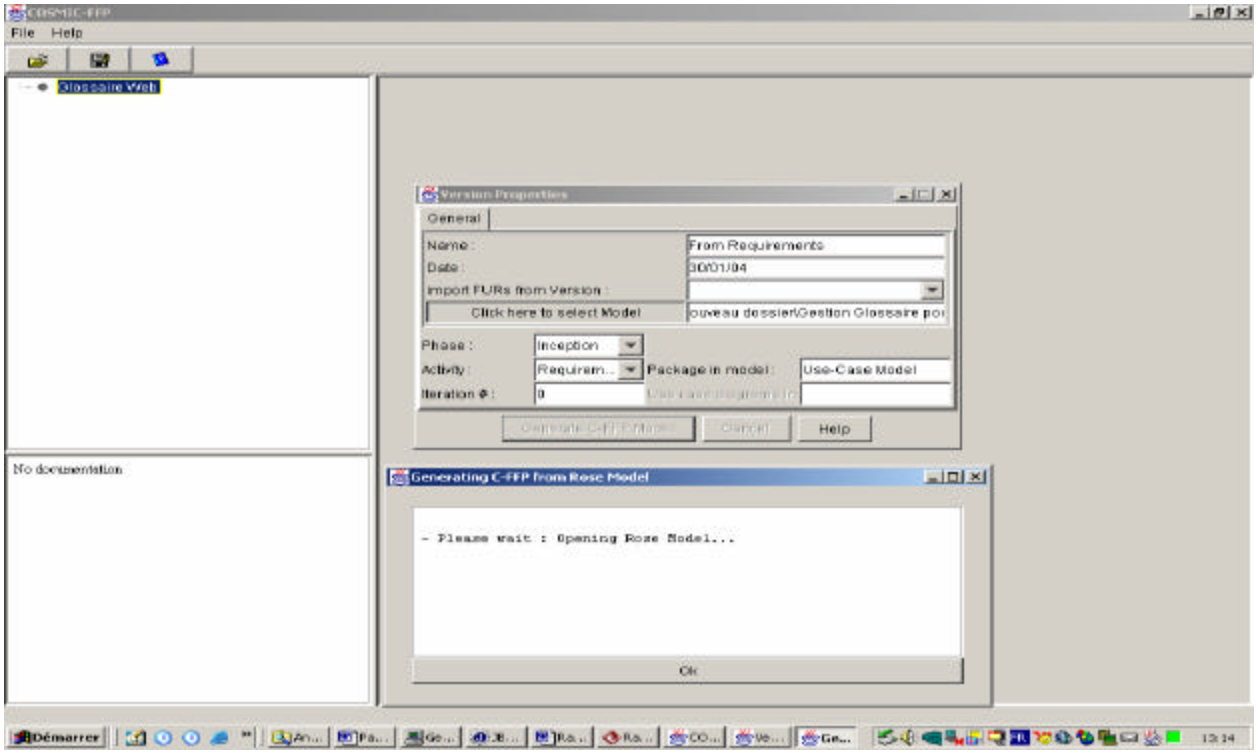

Figure 6.6 Génération automatique du modèle COSMIC-FFP

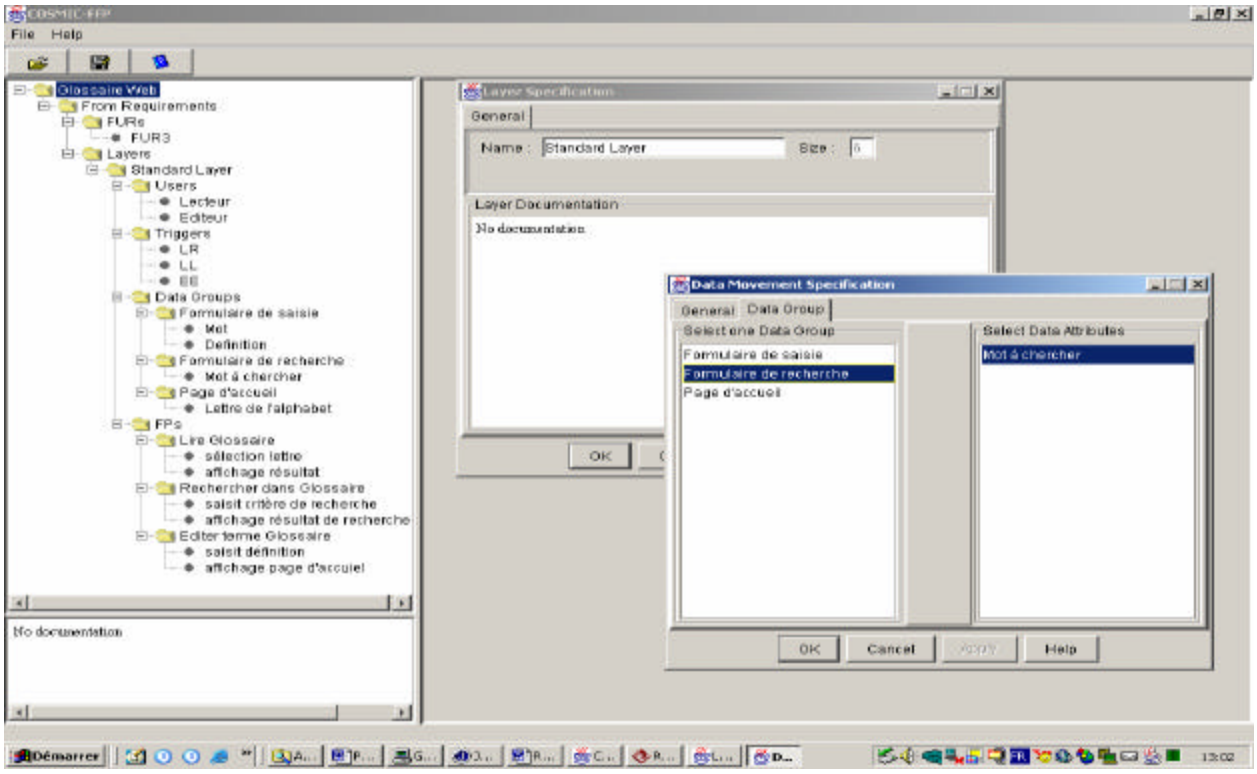

Figure 6.7 Version de mesure à partir des artefacts d'expression des besoins (Requirements)

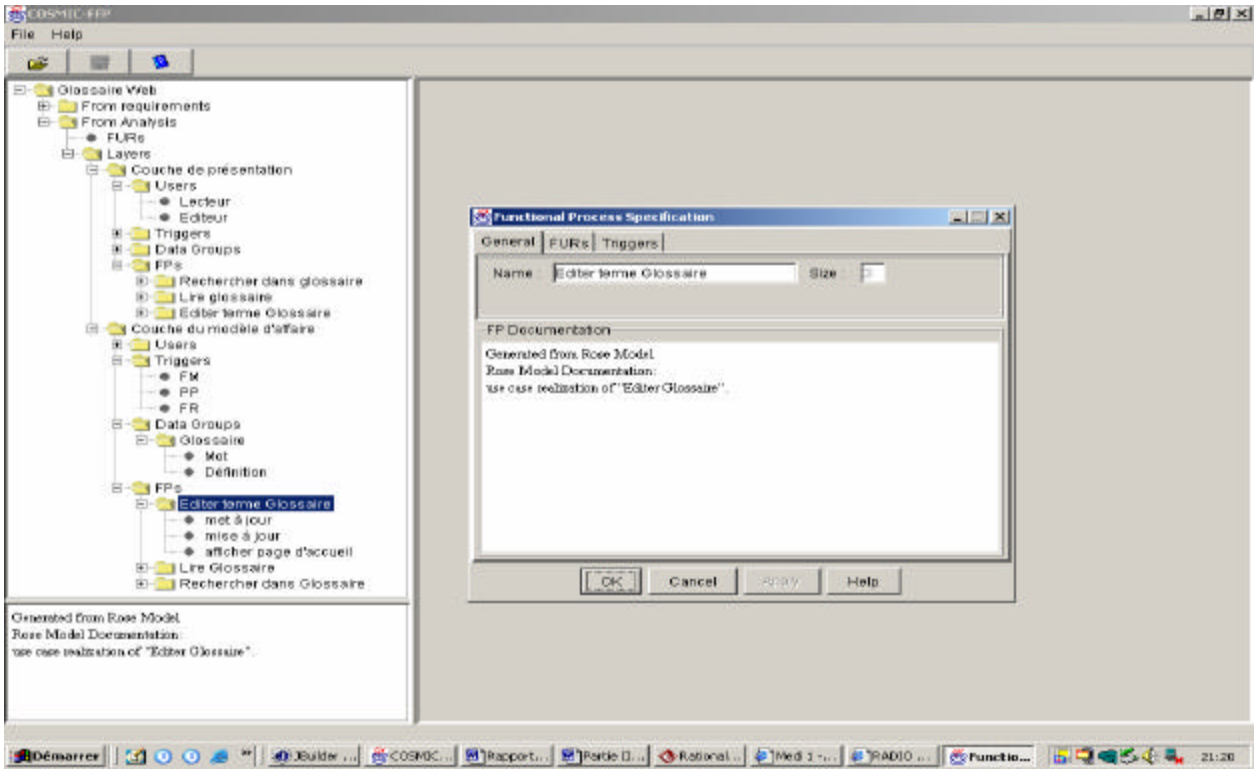

Figure 6.8 Version de mesure à partir des artefacts d'analyse (Analysis)

6. Conclusion

CFFP-GAUGE est une première expérience pour l'automatisation du processus de mesure COSMIC-FFP, il exploite l'étude théorique développé dans les chapitres précédents pour mettre en application les règles de rapprochement entre les éléments COSMIC-FFP et celui des modèles UML de RUP.

Cette première version de CFFP-GAUGE est considérée comme une réussite vue qu'elle exploite la totalité des règles de rapprochement tout en représentant graphiquement le modèle COSMIC-FFP de façon ergonomique à l'utilisateur. Plusieurs améliorations doivent être entreprises pour créer un outil complet équivalent à la valeur que représente actuellement COSMIC-FFP. On note essentiellement :

- Ajouter un nouveau package qui permet de construire graphiquement le modèle COSMIC-FFP semblable a ce qui existe dans Rational Rose, en adoptant ce qui est décrit dans la section 2 du chapitre précédent, à savoir un processus de mesure COSMIC-FFP basé sur la notation UML. En fait, la partie droite de l'interface GUI de la version courante n'est pas exploitée, car elle est prévue pour ce besoin.
- Créer un site Web semblable à celui de RUP pour décrire le processus de mesure COSMIC-FFP en adoptant CFFP-GAUGE comme un « tool mentor » pour l'activité de mesure (semblable à Rational Rose pour RUP).
- Adapter le paquetage de génération actuelle à d'autres outils de modélisation semblables à Rose tel que Together par exemple.
- Créer une version Web de COSMIC-FFP lui permettant d'être exploité à travers le réseau Web pour qu'il soit bien testé, et favorisant ainsi l'expansion de COSMIC-FFP.

Annexe A:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:annotation>
    <xsd:documentation>
      This is a XML Schema for Castor XML.
    </xsd:documentation>
  </xsd:annotation>
<!--
Project Class
-->
  <xsd:element name="projectXML">
    <xsd:annotation>
      <xsd:documentation>
        ProjectXML let to get and set all Project attributes to and from
        XML file (persistent Data).
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="count" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="number" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="type" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="description" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="versionXML" maxOccurs="unbounded" minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
<!--
Version Class
-->
  <xsd:element name="versionXML">
```

```
    <xsd:annotation>
     <xsd:documentation>
       versionXML let to get and set all Version attributes to and from
       XML file (persistent Data).
     </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="count" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
       <xsd:element name="number" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
       <xsd:element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
       <xsd:element name="phase" type="xsd:string" minOccurs="0" maxOccurs="1"/>
       <xsd:element name="activity" type="xsd:string" minOccurs="0" maxOccurs="1"/>
       <xsd:element name="iteration" type="xsd:integer" minOccurs="0" maxOccurs="1"/>
       <xsd:element name="size" type="xsd:int" minOccurs="1" maxOccurs="1"/>
       <xsd:element name="date" type="xsd:date" minOccurs="0" maxOccurs="1"/>
       <xsd:element name="description" type="xsd:string" minOccurs="1" maxOccurs="1"/>
       <xsd:element ref="layerXML" maxOccurs="unbounded" minOccurs="0"/>
       <xsd:element ref="furXML" maxOccurs="unbounded" minOccurs="0"/>
     </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
<!--
FunctionalUserRequirement Class
-->
  <xsd:element name="furXML">
    <xsd:annotation>
     <xsd:documentation>
       furXML let to get and set all FunctionalUserRequirement attributes to and from
       XML file (persistent Data).
     </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
     <xsd:sequence>
```

```
      <xsd:element name="count" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="number" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="description" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
<!--
Layer Class
-->
 <xsd:element name="layerXML">
  <xsd:annotation>
   <xsd:documentation>
    layerXML let to get and set all Layer attributes to and from
    XML file (persistent Data).
   </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="count" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="number" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="size" type="xsd:int" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="description" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element ref="userXML" maxOccurs="unbounded" minOccurs="0"/>
    <xsd:element ref="dataGroupXML" maxOccurs="unbounded" minOccurs="0"/>
    <xsd:element ref="triggerXML" maxOccurs="unbounded" minOccurs="0"/>
    <xsd:element ref="fpXML" maxOccurs="unbounded" minOccurs="0"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
<!--
User Class
-->
```

```
  <xsd:element name="userXML">
    <xsd:annotation>
      <xsd:documentation>
        UserXML let to get and set all User attributes to and from
        XML file (persistent Data).
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="count" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="number" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="type" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="description" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
<!--
DataGroup Class
-->
  <xsd:element name="dataGroupXML">
    <xsd:annotation>
      <xsd:documentation>
        DataGroupXML let to get and set all DataGroup attributes to and from
        XML file (persistent Data).
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="count" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="number" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="type" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="description" type="xsd:string" minOccurs="1" maxOccurs="1"/>
```

```
      <xsd:element ref="dataAttributeXML" maxOccurs="unbounded" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
<!--
Trigger Class
-->
 <xsd:element name="triggerXML">
  <xsd:annotation>
   <xsd:documentation>
    TriggerXML let to get and set all Trigger attributes to and from
    XML file (persistent Data).
   </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="count" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="number" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="description" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element ref="userXML" minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
<!--
Functional Process Class
-->
 <xsd:element name="fpXML">
  <xsd:annotation>
   <xsd:documentation>
    FpXML let to get and set all Functional Process attributes to and from
    XML file (persistent Data).
   </xsd:documentation>
  </xsd:annotation>
```

```
    <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="count" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
       <xsd:element name="number" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
       <xsd:element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
       <xsd:element name="size" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
       <xsd:element name="description" type="xsd:string" minOccurs="1" maxOccurs="1"/>
       <xsd:element ref="dataMvtXML" maxOccurs="unbounded" minOccurs="0"/>
       <xsd:element ref="triggerXML" maxOccurs="unbounded" minOccurs="0"/>
       <xsd:element ref="furXML" maxOccurs="unbounded" minOccurs="0"/>
     </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
<!--
DataAttribute Class
-->
  <xsd:element name="dataAttributeXML">
    <xsd:annotation>
      <xsd:documentation>
        DataAttributeXML let to get and set all DataAttribute attributes to and from
        XML file (persistent Data).
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="count" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="number" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="description" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
<!--
DataMovement Class
```

```
-->
  <xsd:element name="dataMvtXML">
    <xsd:annotation>
      <xsd:documentation>
        DataMvtXML let to get and set all DataMovement attributes to and from
        XML file (persistent Data).
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="count" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="number" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="type" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="description" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="dataGroupXML" minOccurs="0" maxOccurs="1"/>
        <xsd:element ref="dataAttributeXML" maxOccurs="unbounded" minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Références :

[1] : Application de la méthode FFP à partir d'une spécification selon la notation UML : compte-rendu des premiers essais d'application et questions ; Valéry Bévo, Ghislain Lévesque et Alain Abran, Département d'informatique UQAM, 1999.

[2] : A formal definition of COSMIC-FFP for automated measurement of ROOM specifications; H. Diab, M. Frappier, and R. St-Denis, Département de mathématique et d'informatique, Université de Sherbrooke, 2001

[3] : Approche multi-agents pour la mesure de la taille fonctionnelle des logiciels ; Valéry Bévo, Ghislain Lévesque et Alain Abran et Jean-Guy Meunier LRGL – LANCI, UQAM, 2001

[4] : Automation of counting of functional size using COSMIC-FFP in UML; Malcolm S Jenner School of Computing and Information Technology, University of Wolverhampton, UK; 2001.

[5] : A Proposed Measurement Role in the Rational Unified Process an dits Implementation with ISO 19761: COSMIC-FFP; Saadi Azzouz et Alain Abran; ETS; 2003.

[6] : COSMIC-FFP Measurement Manuel, Version 2.1, Mai 2001.

[7] : UML principes de Modélisation, Rémy Fannader et Hervé Leroux ; Édition Dunod, Paris, 1999.

[8] : *UML User Guide*; G. Booch, J. Rumbaugh, and I. Jacobson, 1998. Addison-Wesley Longman

[9] : Site RUP à l'ETS: https://intra.ele.etsmtl.ca/academique/documents/RationalUnifiedProcess/

[10] : The Unified Software Development Process; Ivar Jacobson, Grady Booch et James Rumbaugh; Addison-Wesley Pub Co; 04 February, 1999.

[11] : Software Project Management : A Unified Framework; Walker Royce, Addison-Wesley, 1998.

[12] : Building Web Applications with UML, Jim Conallen, Pearson Education; March, 2000.