REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université AMAR TELIDJI

Laghouat

Faculté des Sciences

Département de Mathématiques et Informatique

Filière : Informatique

Option : *Systèmes d'information et de décision*

Thème :

Elaboration d'un outil d'aide à la conception agile en utilisant la technologie Workflow

Présenté par:

- **Djeridane Fatima Zahra**
- **ABDI Fatima Zahra**

Soutenu devant le jury composé de :

Mme A.Chettih Université de Laghouat (Président)

Mr L. Kechna Université de Laghouat (Examinateur)

Mr L. Chellama Université de Laghouat (Examinateur)

Mme B. Kerrouche Université de Laghouat (Encadreur)

Année universitaire 2012/2013

Remerciements

En préambule à ce mémoire, nous souhaitons adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire de prés ou de loin.

Nous tenons à remercier sincèrement Madame ***KERROUCHE BADRA***, qui, en tant que Dirigeante de mémoire, s'est toujours montrée à l'écoute et très disponible tout au long de la réalisation de ce mémoire, ainsi pour l'inspiration, l'aide et le temps qu'elle a bien voulu nous consacrer et sans qui ce mémoire n'aurait jamais vu le jour.

Nos remerciements s’adressent également à tous nos enseignant(e)s pour leur disponibilité. Nous exprimons notre gratitude à tous les consultants et internautes rencontrés lors des recherches effectuées et qui ont accepté de répondre à nos questions avec gentillesse.
Nous n'oublions pas nos parents, pour leur contribution, leur soutien et leur patience. Nous tenons à exprimer notre profonde reconnaissance envers nos examinateurs qui ont eu la gentillesse de lire et corriger ce travail.
Grand merci à tout le personnel de la bibliothèque du département de mathématiques et informatique.

DEDICACES

Je dédie ce travail tout d'abord spécialement à mes chers parents qui ont tout fait pour ma réussite, a mon mari Othmane, mes frères Adel et Abdelghaffar à mes sœurs Radia et Anissa. Sans que j'oublie mon oncle Kamel qui m'a toujours encouragé, A toute la famille Ramdani, ainsi que mes chères amies qui m'ont accompagnées tout au long du parcours universitaire et tous ceux (celles) qui me sont cher(e)s.

DJERIDANE FATIMA ZAHRA.

Acronymes

1. UML : ***Unified Modeling Language***
2. UP : ***Unifield Process***
3. AM : ***Agile Modeling***
4. XP : ***eXtreme Programming***
5. RUP : ***Rational Unified Process***
6. IHM : ***Interface Home Machine***

Sommaire

Chapitre I : Généralités

Chapitre II : Etat de l'art

Chapitre III : Etude conceptuelle

Chapitre IV : Mise en œuvre

Liste des figures

Chapitre II : Etat de l'art

Chapitre III : Etude conceptuelle

Chapitre IV : Mise en œuvre

Résumé

Depuis les années 1960, le besoin des entreprises en termes de projets informatiques change, le taux de réussite des projets n'est pas très satisfaisant, les méthodes d'informatisation ne sont plus adaptées et doivent évoluer à leur tour. Ce changement a abouti, depuis les années 1990, à une philosophie appelée ***Modélisation Agile***, elle a pour but de mettre le client en avant et de lui livrer des parties opérationnelles du projet régulièrement pour qu'il puisse juger de la validité de celles-ci et donc supprimer les risques d'échec en fin de projet.

Le présent travail rentre dans le cadre de la préparation du mémoire de fin d'études en vue de l'obtention d'un diplôme de Master en informatique. Il consiste à ***concevoir***, ***développer*** et ***implémenter*** un outil d'aide à la conception agile qui permet à l'utilisateur et surtout à l'équipe de développement de suivre une vision parfaitement agile en concevant avec UML, au sein d'un cadre de travail parallèle collaboratif grâce à l'adoption de la technique Workflow. Pour mener à bien cette étude, on s'est essentiellement basé sur une démarche de modélisation rendant l'UML agile proposée par ***Pascal Roques***.

En effet, le travail porte, d'une part sur la modélisation de l'outil par le langage UML, et d'autre part sur son implémentation en utilisant l'outil ***BonitaOpenSolution*** dans le cadre du développement d'un logiciel de recherche.

Mot clés :

UML agile, conception agile, Modélisation Agile, Technique Workflow, BonitaOpenSolution.

Abstract

Since the 1960s, the need for companies in terms of IT project changes, the success rate of projects is not very satisfactory, computerized methods are no longer appropriate and should evolve in turn. This change resulted from the 1990s; a philosophy called Agile Modeling, to put the customer in front of him and deliver operational parts of the project on a regular basis so that it can assess the validity of these it is intended to and thus eliminate the risk of failure at the end of the project.

This work falls within the framework of the preparation of the dissertation studies to obtain a Master's degree in computer science. It is to ***design***, ***develop*** and ***implement*** a tool for agile design that allows the user and especially the development team to follow a perfectly agile vision in designing with UML, within a framework collaborative parallel with the adoption of workflow technology. To carry out this study, it is essentially based on a modeling approach making agile **UML** proposed by ***Pascal Roques***.

Indeed, the work focuses on the one hand on the modeling tool **UML**, and secondly on its implementation using ***BonitaOpenSolution*** tool in the software development research.

Key words:

Approach agile **UML,** tool for agile design, Agile Modeling, workflow technology, BonitaOpenSolution

Introduction générale

Le recours à la modélisation est une pratique indispensable au développement logiciel, les recherches dans ce domaine ont donné naissance à une multitude de méthodes. S'appuyer sur de telles méthodes résout le problème de la complexité des projets en terme de leur compréhension et leur maitrise globale, qui dépasse la capacité d'un seul individu ce qui met l'importance sur le travail d'équipe.

Les méthodes traditionnelles nécessitent une maitrise profonde pour donner de bons résultats ce qui implique un temps d'apprentissage important, de plus ces méthodes et selon plusieurs enquêtes ne sont jamais appliquées à la lettre.

Une méthode dite traditionnelle attend généralement du client une expression détaillée et validée du besoin en entrée de la réalisation, laissant peu de place au changement. La réalisation dure le temps qu'il faut et le rendez vous est repris avec le client qu'après réalisation. Cet effet tunnel peut être très néfaste et conflictuel, on constate souvent un déphasage entre le besoin initial et l'application réalisée.

De plus il n'est pas rare que certaines fonctionnalités demandées se révèlent finalement inutiles à l'usage alors que d'autres, découvertes en cours de route, auraient pu donner plus de valeur au produit.

Ce que recherche une équipe de développement en systèmes informatiques c'est une méthode efficace, flexible, permettant d'aboutir à de bons logiciels, ce qui revient à dire les produire en un temps et un coût moindre tout en répondant au mieux aux exigences de l'utilisateur. Ce besoin a donné naissance à la notion de développement agile et de méthodes agiles qui feront l'objet de ce mémoire.

En effet nous proposons dans ce mémoire un outil permettant de supporter le travail d'une équipe de développement en utilisant la technologie workflow et en s'inspirant de la démarche proposée par Pascal Roques qui rend le langage de modélisation UML agile. Il s'agit d'une limitation volontaire dans le processus de développement permettant une réduction significative du temps d'apprentissage en n'utilisant qu'un sous ensemble des diagrammes d'UML jugé nécessaire et suffisant; Cette démarche est basée sur les principes fondamentaux du processus unifié (UP) et du développement agile

Chapitre I

- **Introduction**
- **Problématique**
- **Objectifs**
- **Structure du mémoire**

Introduction

Quand le besoin et l'exigence informatique créent de nouveaux niveaux de complexité ainsi que de nouvelles demandes sur les projets informatiques d'une entreprise, a leur tour les recherches et les visions existantes doivent changer et évoluer pour garantir l'équilibre entre besoin et satisfaction.

Problématique

De nos jours l'informatique représente un pilier majeur dans les entreprises. Cependant les méthodes de développement des systèmes informatiques n'évoluent pas aussi vite que les technologies et les besoins. L'évolution des besoins rend les projets d'informatisation de plus en plus complexes, c'est pourquoi les concepteurs arrivent très souvent à des résultats insatisfaisants pour le client, ne correspondant pas à ses exigences et à ses attentes.

Le problème est donc de trouver une façon de penser et de travailler qui corresponde au besoin actuel qui est de fournir rapidement le produit répondant aux exigences métier.

De nouvelles méthodes de développement ont vu le jour pour faire face à cette forte demande et ce manque de réussite des projets. On parle de développement et de méthodes agiles.

Sachant que le langage de modélisation UML a fait ses preuves et est largement utilisé, l'auteur ***Pascal Rocque*** [PR1] a proposé de l'adapter au développement agile en s'appuyant sur les principes de ce dernier et en essayant de répondre à la question suivante : comment passer efficacement des besoins des utilisateurs au code ?

L'idée proposée dans ce mémoire est de développer un outil qui supporte le travail et la communication entre les membres d'une équipe de développement en s'inspirant de la démarche proposée par ***Pascal Rocque***.

Objectifs

Afin de bien mener ce travail les objectifs suivants ont bien été fixés :

- L'objectif principal que nous voulons atteindre est la réalisation d'un outil supportant le travail et les échanges d'une équipe de développement qui adopte la démarche de l'UML Agile.
- Augmenter le niveau de satisfaction du client en adoptant les approches agiles.

- Comprendre le système à faire et communiquer en interne et externe.
- Trouver le juste milieu entre: Pas de modélisation du tout et trop de modélisation.

Structure du mémoire

Le présent document est organisé comme suit :

I. Après une introduction générale, dans le premier chapitre nous présentons le recours à la modélisation agile ainsi que l'apparition des méthodes agiles. Nous présentons la problématique posée dans ce travail et les objectifs de ce dernier.

II. Dans le second chapitre nous présentons un état de l'art sur le développement des SI, ainsi nous mettons l'accent sur les différences majeures entre les méthodes traditionnelles et agiles en citant différents concepts en rapport direct avec le sujet.

III. Nous verrons dans le troisième chapitre l'étude conceptuelle de l'outil envisagé.

IV. Le quatrième chapitre sera dédié à la mise en œuvre de notre outil. Nous présenterons la notion de travail collaboratif entre les acteurs du système tout en mettant l'accent sur la circulation automatique de flux d'informations.

Pour terminer nous concluons et indiquons quelques perspectives.

Chapitre II

II.1. Introduction

L'élaboration de l'outil d'aide à la conception agile dans le cadre d'un travail parallèle et collaboratif qui fait l'objet de ce mémoire repose essentiellement sur le suivi d'une démarche rendant l'UML agile en implémentant une application workflow.

Dans cette partie de l'étude, on s'intéresse à la présentation des notions et des concepts sur lesquels se base l'élaboration de notre outil.

On mettra l'accent sur l'apparition du manifeste agile et ce que c'est, ainsi que sur le processus de développement UP (Unifield Process) dont les méthodes agiles se sont inspirées. Puis une présentation de la démarche proposée par ***Pascal Roques*** qui explique comment rendre l'UML agile. Et enfin on présente ce que c'est la technologie workflow qui supporte le travail collaboratif.

II.2. Unifield Process (UP)

Contrairement au processus en cascade (souvent appelé cycle en V, en France), le Processus Unifié ne considère pas que les disciplines sont purement séquentielles. En fait, une itération comporte une certaine quantité de travail dans la plupart des disciplines. Cependant, la répartition de l'effort relatif entre celles-ci change avec le temps. Les premières itérations ont tendance à mettre plus l'accent sur les exigences et la conception, les autres moins, à mesure que les besoins et l'architecture se stabilisent grâce au processus de feedback et d'adaptation. [PR1]

II.2.1. L Les principes fondamentaux de l'UP (Unifield Process) :

Le Processus Unifié est un processus de développement logiciel « itératif et incrémental, centré sur l'architecture, conduit par les cas d'utilisation et piloté par les risques ». [PR1]

- ***Itératif et incrémental* :** le projet est découpé en itérations de courte durée (environ 1 mois) qui aident à mieux suivre l'avancement global. À la fin de chaque itération, une partie exécutable du système final est produite, de façon incrémentale.
- ***Centré sur l'architecture* :** tout système complexe doit être décomposé en parties modulaires afin de garantir une maintenance et une évolution facilitées. Cette architecture (fonctionnelle, logique, matérielle, etc.) doit être modélisée en UML et pas seulement documentée en texte.
- ***Piloté par les risques* :** les risques majeurs du projet doivent être identifiés au plus tôt, mais surtout levés le plus rapidement possible. Les mesures à prendre dans ce cadre déterminent l'ordre des itérations.
- ***Conduit par les cas d'utilisation* :** le projet est mené en tenant compte des besoins et des exigences des utilisateurs. Les cas d'utilisation du futur système sont identifiés, décrits avec précision et priorisés. [UP2]

II.2.2. Les phases et les disciplines de l'UP :

La gestion d'un tel processus est organisée suivant les quatre phases suivantes : ***initialisation***, ***élaboration***, ***construction*** et ***transition***.

Chaque phase est elle-même décomposée séquentiellement en itérations limitées dans le temps (entre 2 et 4 semaines). Le résultat de chacune d'elles est un système testé, intégré et exécutable.

L'approche itérative est fondée sur la croissance et l'affinement successifs d'un système par le biais d'itérations multiples, feedback et adaptation cycliques étant les moteurs principaux permettant de converger vers un système satisfaisant. Le système croît avec le temps de façon incrémentale, itération par itération, et c'est pourquoi cette méthode porte également le nom de développement itératif et incrémental. Il s'agit là du principe le plus important du Processus Unifié. [VM4]

- ***La phase d'initialisation :*** conduit à définir la « vision » du projet, sa portée, sa faisabilité, son business case, afin de pouvoir décider au mieux de sa poursuite ou de son arrêt.

Nb : Le Business Case met en avant l'objectif du projet. Pour ce faire, il énumère les raisons pour lesquelles le projet a été initié, les bénéfices attendus, les options à considérer (avec les facteurs de rejet ou d'approbation expliquant la prise en compte ou pas de chaque option), les coûts prévisibles, l'analyse des carences et les risques.

- ***La phase d'élaboration :*** poursuit trois objectifs principaux en parallèle :
    - identifier et décrire la majeure partie des besoins des utilisateurs.
    - construire (et pas seulement décrire dans un document !) l'architecture de base du système.
    - lever les risques majeurs du projet.
- ***La phase de construction :*** consiste surtout à concevoir et implémenter l'ensemble des éléments opérationnels (autres que ceux de l'architecture de base). C'est la phase la plus consommatrice en ressources et en effort.
- ***La phase de transition :*** permet de faire passer le système informatique des mains des développeurs à celles des utilisateurs finaux. Les mots-clés sont : conversion des données, formation des utilisateurs, déploiement, béta-tests.

NB : Le bêta-test est la deuxième période d'essai d'un produit informatique avant sa publication. Un produit en période de bêta test est généralement soumis à un nombre important ou représentatif de personnes : les bêta-testeurs. Ils peuvent être soit des employés de la société qui développe le logiciel, soit des bénévoles notamment dans le cas des logiciels libres. Ces personnes ont pour but d'utiliser le logiciel et de rapporter les problèmes rencontrés ainsi que leurs suggestions.

Le bêta test sert essentiellement à trouver des bugs résiduels, ou bien à modifier l'interface utilisateur.

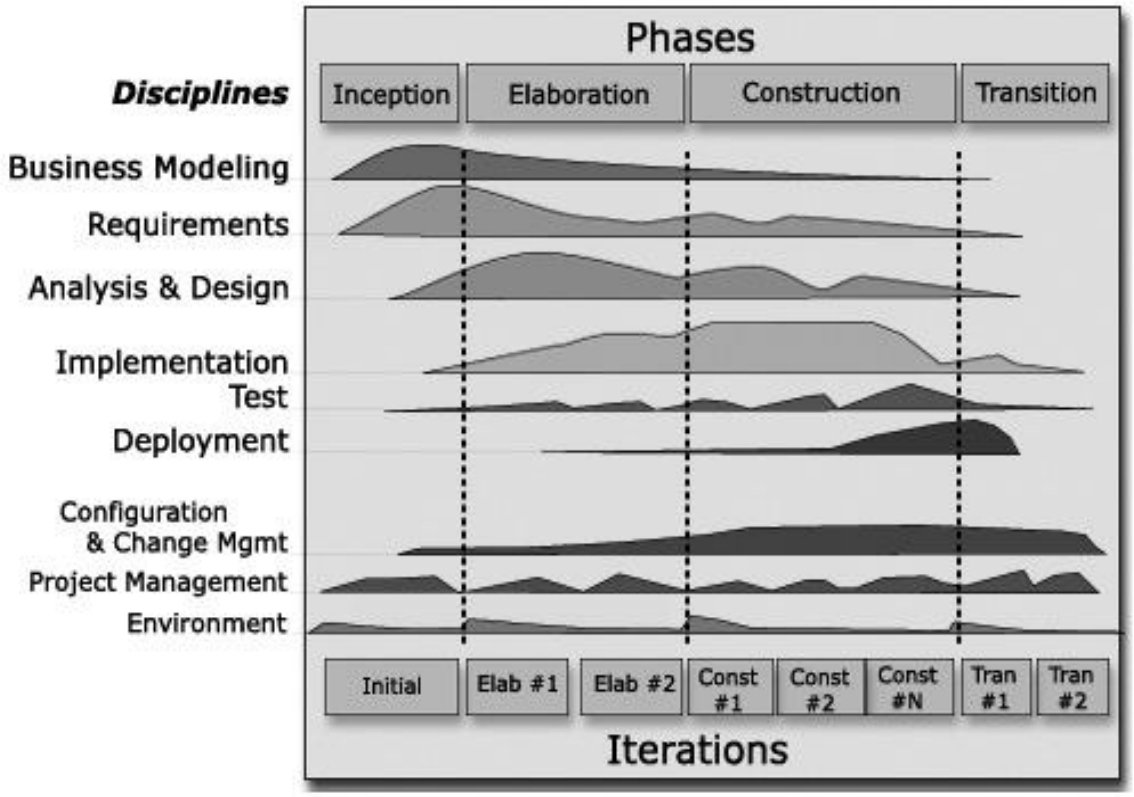


Figure01 : les phases de l'UP

[***Source*** : Les cahiers du programmeur UML2, 4ème édition, pascal Roques, page : 11]

II.3. Le manifeste agile

II.3.1. Apparition

La notion de méthode agile est née à travers un manifeste signé en 2001 par 17 personnalités du développement logiciel (parmi lesquelles Ward Cunningham, Alistair Cockburn, Kent Beck, Martin Fowler, Ron Jeffries,Steve Mellor, Robert C. Martin, Ken Schwaber, Jeff Sutherland, etc.).

II.3.2. Les bonnes pratiques de la modélisation agile

La « modélisation agile » prônée par Scott Ambler s'appuie sur des bonnes pratiques simples et de bon sens, parmi lesquels :

•Avoir une grande palette de techniques à disposition et connaître les forces et les faiblesses de chacune de manière à pouvoir appliquer la meilleure au problème courant.

• Ne pas hésiter de changer de diagramme quand on sent qu'on n'avance plus avec le modèle en cours. Le changement de perspective va permettre de voir le problème sous un autre angle et de mieux comprendre ce qui bloquait précédemment.

• On trouve souvent qu'on est plus productif si on crée plusieurs modèles simultanément plutôt qu'en on focalisant sur un seul type de diagramme. [PR1]

II.3.3. Les bases du manifeste agile

Ce manifeste prône quatre valeurs fondamentales :

« Personnes et interactions plutôt que processus et outils » **:** dans l'optique agile, l'équipe est bien plus importante que les moyens matériels ou les procédures. Il est préférable d'avoir une équipe soudée et qui communique, composée de développeurs moyens, plutôt qu'une équipe composée d'individualistes, même brillants. La communication est une notion fondamentale.

« Logiciel fonctionnel plutôt que documentation complète » **:** il est vital que l'application fonctionne. Le reste, et notamment la documentation technique, est secondaire, même si une documentation succincte et précise est utile comme moyen de communication. La documentation représente une charge de travail importante et peut être néfaste si elle n'est pas à jour. Il est préférable de commenter abondamment le code lui-même, et surtout de transférer les compétences au sein de l'équipe (on en revient à l'importance de la communication).

« Collaboration avec le client plutôt que négociation de contrat » : le client doit être impliqué dans le développement. On ne peut se contenter de négocier un contrat au début du projet, puis de négliger les demandes du client. Le client doit collaborer avec l'équipe et fournir un feedback continu sur l'adaptation du logiciel à ses attentes.

« Réagir au changement plutôt que suivre un plan » **:** la planification initiale et la structure du logiciel doivent être flexibles afin de permettre l'évolution de la demande du client tout au long du projet. Les premières releases du logiciel vont souvent provoquer des demandes d'évolution. [PR1]

II.3.4. Valeur de la modélisation agile

Les clés d'une modélisation réussie à promouvoir une communication efficace entre tous les participants du projet, de se forcer à développer la solution la plus simple possible qui satisfasse tous les besoins, à obtenir des retours rapides et fréquents, à avoir le courage de prendre des décisions et de à s'y tenir et à avoir l'humilité de reconnaitre qu'on ne sait pas tout et que les autres ont une valeur à apporter à ses propres efforts.

La figure suivante résume les valeurs agiles :

Figure02: Les valeurs agiles

[***Source :*** Modelisation agile & UML, Agnès CREPET ,3 décembre 2011, Page :07]

II.3.5. Etude comparative entre méthodes traditionnelles et méthodes agiles

Une enquête de 1994 du « Standish Group » [EQ2] (certes controversée, comme toutes les enquêtes qui traitent d'un sujet sensible) fait le constat suivant sur les méthodes traditionnelles: « 31 % des projets informatiques sont arrêtés en cours de route, 52 % n'aboutissent qu'au prix d'un important dépassement des délais et du budget et en offrent

moins de fonctionnalités qu'il n'en était demandé ; seuls 16 % des projets peuvent être considérés comme des succès. ». [EQ3]

Cette même enquête renouvelée en 2008 indique un taux de réussite de 35%, ce qui est plutôt positif mais demeure assez faible surtout que c'est méthodes ne sont jamais appliquée à la lettre. Le problème reste donc entier. Parmi les motifs d'échecs, arrivent en tête :

- Manque d'implication des utilisateurs finaux : 12,8 %.
- Changement de spécifications en cours de projet : 11,8 %.

L'approche agile propose au contraire de réduire considérablement voir complètement l'effet tunnel en donnant davantage de visibilité, en impliquant le client du début à la fin du projet et en adoptant un processus de développement itératif et incrémental. Elle considère que le besoin ne peut être figé et propose au contraire de s'adapter aux changements de ce dernier. Mais pas sans un minimum de règles. [EQ3]

Le tableau suivant résume les principales différences entre une approche agile et une approche traditionnelle :

Méthodes traditionnelles	Méthodes agiles
Cycle long, livraison finale	**Cycle court, livraison fréquente**
Processus	**Productivité et rentabilité**
Documentation détaillée	**Logiciel fonctionnel avec un minimum de documentation**
Contrat	**Collaboration**
Minimiser les changements	**Accueillir les changements**

Tableau1 **:** Comparaison entre les méthodes traditionnelles et les méthodes agiles

II.3.6. Quelques méthodes agiles et leurs pratiques

Les méthodes agiles sont nombreuses citons quelques unes :

II.3.5.1. L'eXtremeProgramming (XP)

Est un ensemble de pratiques qui couvre une grande partie des activités de la réalisation d'un logiciel, de la programmation proprement dite à la planification du projet, en passant par l'organisation de l'équipe de développement et les échanges avec le client. Ces pratiques ne sont pas révolutionnaires : il s'agit simplement de pratiques de bon sens mises en œuvre par des développeurs ou des chefs de projet expérimentés, telles que:

- ***Un utilisateur à plein-temps dans la salle projet :*** Ceci permet une communication intensive et permanente entre les clients et les développeurs, aussi bien pour l'expression des besoins que pour la validation des livraisons.
- ***Écrire le test unitaire avant le code qu'il doit tester*** : Afin d'être certain que le test sera systématiquement écrit et non pas négligé.
- ***Programmer en binôme*** : Afin d'homogénéiser la connaissance du système au sein des développeurs, et de permettre aux débutants d'apprendre auprès des experts. Le code devient ainsi une propriété collective et non individuelle, que tous les développeurs ont le droit de modifier.
- ***Intégrer de façon continue :*** Pour ne pas repousser à la fin du projet le risque majeur de l'intégration des modules logiciels écrits par des équipes ou des personnes différentes.

En résumé : XP est une méthodologie légère qui met l'accent sur l'activité de programmation et qui s'appuie sur les valeurs suivantes : communication, simplicité et feedback. Elle est bien adaptée pour des projets de taille moyenne où le contexte (besoins des utilisateurs, technologies informatiques) évolue en permanence.

II.3.5.1. La méthode Scrum

Le principe de base de Scrum est de focaliser l'équipe de façon itérative sur un ensemble de fonctionnalités à réaliser, dans des itérations de 30 jours, appelées Sprints.

Chaque Sprint possède un but à atteindre, défini par le directeur de produit (Product owner), à partir duquel sont choisies les fonctionnalités à implémenter dans ce Sprint. Un Sprint aboutit toujours sur la livraison d'un produit partiel fonctionnel.

Pendant ce temps, le scrum master a la charge de réduire au maximum les perturbations extérieures et de résoudre les problèmes non techniques de l'équipe.

Un principe fort en Scrum est la participation active du client pour définir les priorités dans les fonctionnalités du logiciel, et choisir lesquelles seront réalisées dans chaque Sprint. Il peut à tout moment ajouter ou modifier la liste des fonctionnalités à réaliser, mais jamais ce qui est encours de réalisation pendant un Sprint.

II.3.5.1. RUP (Rational UnifiedProcess)

RUP (Rational UnifiedProcess) proposée en 1998 par Rational Software (IBM) est l'une des plus célèbres implémentations de l'UP (Unified Process).

RUP permet plutôt de donner un cadre au développement logiciel afin d'allier la philosophie **agile** et modélisation répondant aux exigences fondamentales préconisées par les créateurs d'UML (Unified Modeling Language).

- Une méthode de développement doit être guidée par les besoins des utilisateurs ;
- Elle doit être centrée sur l'architecture logicielle ;
- Elle doit être itérative et incrémentale ;

Le Rational Unified Process est un logiciel d'aide au développement qui met en valeur la productivité de l'équipe et fournit les meilleures lignes directrices en matière de conception logiciel à tous les membres de celle-ci.

- Il fournit les directives et les exemples appropriés pour toutes les activités critiques du développement, en même temps que les outils permettant de modéliser et de suivre l'exigence de ses directives.
- Il est non-intrusif et étroitement intégré avec les outils logiciels Rational, permettant à des équipes de développement de profiter des pleins avantages de l'UML, et de l'automatisation du logiciel.

- ***Développement du travail de l'équipe :***

Le Rational UnifiedProcess unifie l'équipe entière de développement et met en valeur la communication à travers l'équipe en fournissant à chaque membre une connaissance, un langage et une vue commune dans la façon développer le logiciel.

- Imprime une direction à l'activité de l'équipe

- Décide quoi développer et quand le développer
- Oriente l'activité de l'individu

- ***Des pratiques robustes et performantes:***

Le Rational UnifiedProcess optimalise la productivité de chaque membre de l'équipe en lui faisant profiter de l'expérience dérivée de milliers de projets et de beaucoup de leaders à travers le monde. Il fournit notamment des outils permettant de mesurer productivité, performance et fiabilité.

- **Méthodologie de Rational UnifiedProcess**

  - Itérative et incrémentale
  - Orientée objet
  - Fortement automatisée
  - Elle est assez générique pour être adapté à une très large variété de produits logiciels, que se soit au niveau de la taille ou du domaine d'application.

Le procédé de Rational pour être vue sous deux aspects :

- Une approche de management, incluant l'aspect financier, stratégique, commercial et humain ;
- Une approche technique, incluant la qualité, le développement et le design ;

Ces approches sont déterminées par trois grands centres d'intérêts:

- Les personnes ;
- Le développement ;
- Les outils et les méthodes ;

Afin de développer le travail d'équipe, Rational pose les questions suivantes :

- Qui ?
- Comment ?
- Quoi ?
- Quand ?

II.4. Démarche rendant l'UML agile

Une démarche proposée inspirée de l'UP (UnifiedProcess) et les méthodes agiles ainsi que les bonnes pratiques de la modélisation agile sera abordée en ce qui suit :

II.4.1.UML (Raisons du choix de l'UML)

De la même façon qu'il vaut mieux dessiner une maison avant de la construire, il vaut mieux modéliser un système avant de le réaliser.

UML étant le langage de modélisation le plus courant avec lequel les concepteurs ont acquis un certain niveau de maitrise et d'expérience.

UML pour :

- Obtenir une modélisation de très haut niveau indépendante des langages et des environnements ;
- Faire collaborer des participants de tous horizons autour d'un même document de synthèse ;
- Faire des simulations avant de construire un système ;
- Exprimer dans un seul modèle tous les aspects statiques, dynamiques, juridiques, spécification, etc. ;
- Documenter un projet ;
- Gérer automatiquement la partie logicielle d'un système.

II.4.2. Présentation de la démarche

Une démarche proposée par Pascal Roques dans son livre le Cahier des programmeurs dans sa 4ème édition [PR1] qui consiste en un processus simplifié pour les applications informatiques. C'est un processus se situant a mi-chemin entre UP (Unifieldprocess) qui est un cadre général très complet de processus de développement et les méthodes agiles en vogues actuellement, telles que XP (eXtremeProgramming) et Scrum. Il s'inspire également des bonnes pratiques prônées par les tenants de la modélisation agile (Agile Modeling). C'est un processus définissant un passage des besoins au code du système.

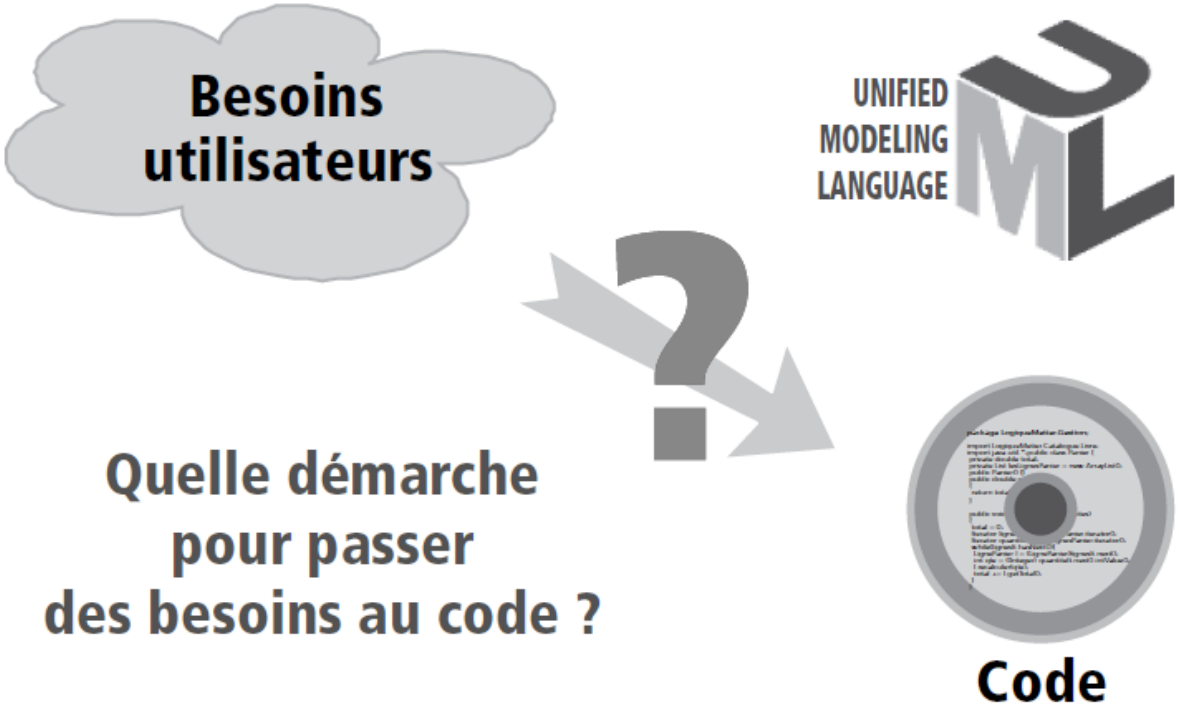


***Figure03*: Démarche pour passer des besoins au code**

[***Source*** **:** Les cahiers du programmeur UML2, 4ème édition, pascal Roques, page : 01]

II.4.3. Principe de la démarche

C'est un processus rendant l'UML agile dont le principe est:

• Conduit par les cas d'utilisation, comme UP, mais beaucoup plus Simple.

• Relativement léger et restreint, comme les méthodes agiles, mais sans négliger les activités de modélisation en analyse et conception.

• Fondé sur l'utilisation d'un sous-ensemble nécessaire et suffisant du langage UML, conformément à AM (Agile Modeling).

Dans cette démarche les parties critiques de systèmes sont modélisées avec précision tout en négligeant les activités de modélisation qui ne sont pas indispensables pour éviter la surcharge des concepts.

C'est donc de chercher comment passer des besoins des utilisateurs au code de l'application ? Autrement dit : « J'ai une bonne idée de ce que mon application doit faire, des

fonctionnalités attendues par les futurs utilisateurs. Comment obtenir le plus efficacement possible un code informatique opérationnel, complet, testé, et qui réponde parfaitement au besoin ? ».

Figure04: Comment passer des besoins au code ?

[***Source*** **:** Les cahiers du programmeur UML2, 4ème édition, pascal Roques, page : 14]

Il ne s'agit pas de se jeter sur l'écriture de code en omettant de formaliser les besoins des utilisateurs et d'élaborer une architecture robuste et évolutive. D'un autre côté, le but n'est pas de faire de la modélisation pour le plaisir, mais bien de produire le plus rapidement possible une application qui satisfasse au mieux ses utilisateurs.

Cette démarche de modélisation nécessaire et suffisante afin de construire efficacement des applications web. Pour cela, un sous-ensemble du langage de modélisation UML qui sera également nécessaire et suffisant pour la plupart des projets de même nature. Cette approche est le résultat de plusieurs années d'expérience dans des domaines variés. Elle a donc montré son efficacité dans la pratique.

Nous exposons cette démarche en plusieurs parties :

Partie 01

Dans un premier temps les besoins vont être modélisés au moyen des cas d'utilisation UML. Ils seront représentés de façon plus concrète par une maquette d'IHM (Interface Homme-Machine) destinée à faire réagir les futurs utilisateurs.

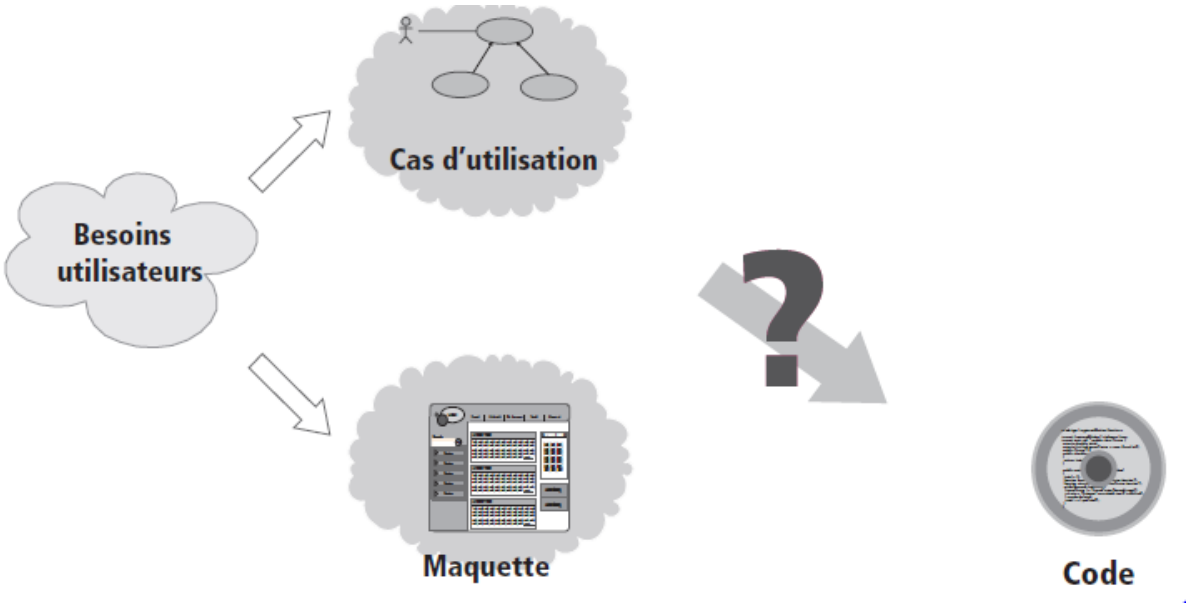


Figure05: Les besoins donnent lieu à des cas d'utilisation et des maquettes

[***Source*** **:** Les cahiers du programmeur UML2, 4ème édition, pascal Roques, page : 15]

NB : Une maquette est un produit jetable donnant aux utilisateurs une vue concrètes mais non définitive de la future interface de l'application. Cela peut consister en un ensemble de dessins réalisés avec des outils spécialisés tels que Dreamweaver ou plus simplement avec Powerpoint ou même Word. Par la suite, la maquette intégrera des fonctionnalités de navigation pour que l'utilisateur puisse tester l'enchainement des écrans, même si les fonctionnalités restent fictives.

La maquette est développée rapidement afin de provoquer des retours de la part des utilisateurs. Elle permet ainsi d'améliorer la relation développeur-client. La plupart du temps, la maquette est considérée comme jetable, c'est-à-dire que la technologie informatique employée pour la réaliser n'est pas forcément assez robuste et évolutive pour être intégrée telle quelle.

Partie 02

Repartons maintenant du but, c'est-à-dire du code que nous voulons obtenir et remontons en arrière, pour mieux expliquer le chemin minimal qui va nous permettre de joindre les deux « bouts ». Dans le cadre de systèmes orientés objet, la structure du code est définie par les classes logicielles et leurs regroupements en ensembles appelés packages. Nous avons donc besoin de diagrammes représentant les classes logicielles et montrant les données qu'elles contiennent (appelées attributs), les services qu'elles rendent (appelés opérations) ainsi que leurs relations. UML propose les diagrammes de classes pour véhiculer toutes ces informations.

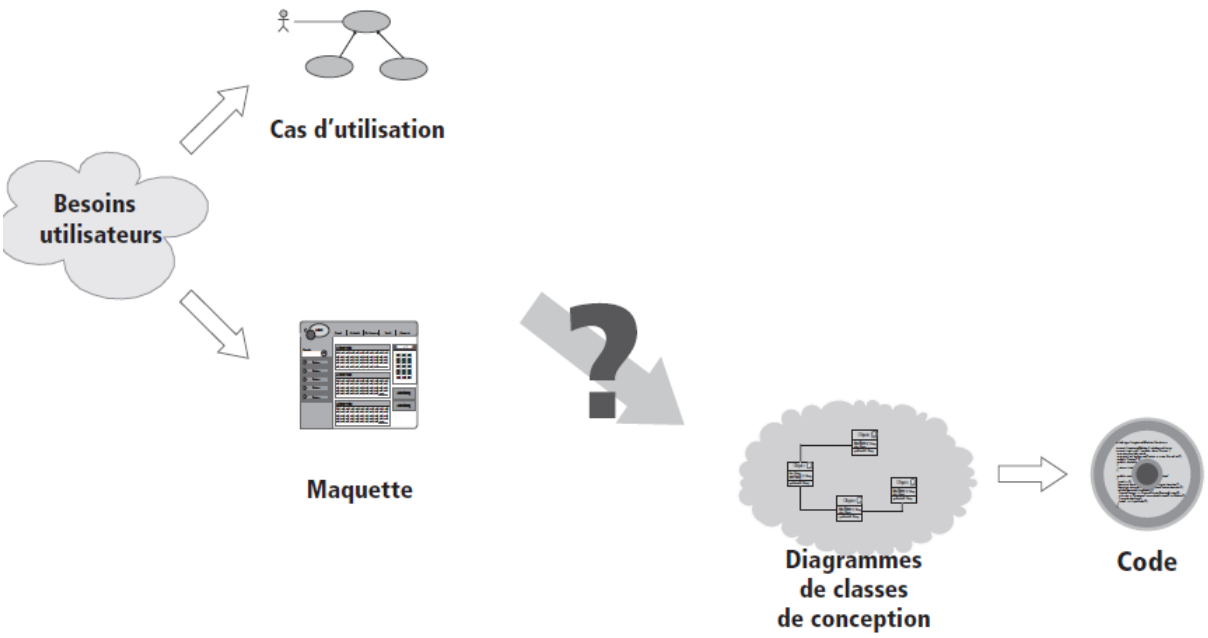


Figure06: Les diagrammes de classes de conception donnent la structure du code

[***Source*** : Les cahiers du programmeur UML2, 4ème édition, pascal Roques, page : 16]

Les diagrammes de classes de conception représentent bien la structure statique du code, par le biais des attributs et des relations entre classes, mais ils contiennent également les opérations (aussi appelées méthodes) qui décrivent les responsabilités dynamiques des classes logicielles.

L'attribution des bonnes responsabilités aux bonnes classes est l'un des problèmes les plus délicats de la conception orientée objet. Pour chaque service ou fonction, il faut décider quelle est la classe qui va le/la contenir.

Tout le comportement du système entre les classes de conception doit être réparti, et décrire les interactions induites.

Les diagrammes d'interactions UML (séquence ou communication) sont particulièrement utiles au concepteur pour représenter graphiquement ses décisions d'allocation de responsabilités. Chaque diagramme d'interaction va ainsi représenter un ensemble d'objets de classes différentes collaborant dans le cadre d'un scénario d'exécution du système.

Dans ce genre de diagramme, les objets communiquent en s'envoyant des messages qui invoquent des opérations sur les objets récepteurs.

On peut donc suivre visuellement les interactions dynamiques entre objets, et les traitements réalisés par chacun. Les diagrammes d'interaction aident également à écrire le code à l'intérieur des opérations, en particulier les appels d'opérations imbriqués.

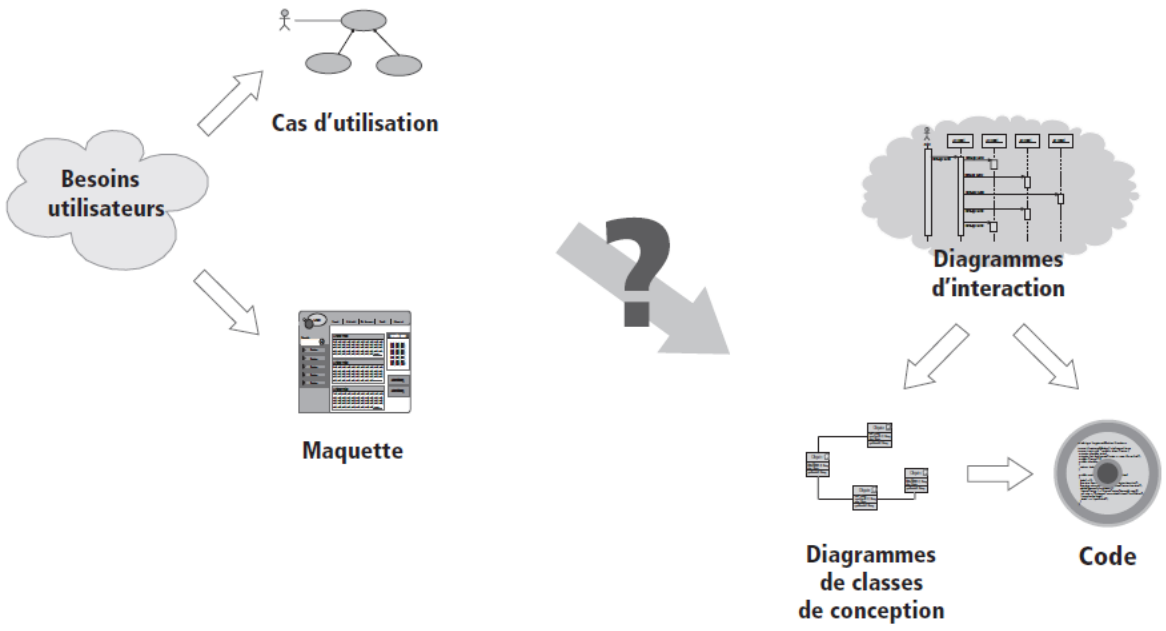


Figure07: Les diagrammes d'interaction aident à attribuer les responsabilités aux classes

[***Source*** : Les cahiers du programmeur UML2, 4ème édition, pascal Roques, page : 17]

Partie 03

Comment passer des cas d'utilisation aux diagrammes d'interaction ? Ce n'est ni simple ni direct, car les cas d'utilisation sont au niveau d'abstraction des besoins des utilisateurs alors que les diagrammes d'interaction se placent au niveau de la conception objet. Il faut donc au moins une étape intermédiaire.

Chaque cas d'utilisation est décrit textuellement de façon détaillée, mais donne également lieu à un diagramme de séquence simple représentant graphiquement la chronologie des interactions entre les acteurs et le système vu comme une boîte noire, dans le cadre du scénario nominal.

Ce diagramme est appelé : « diagramme de séquence système ».

Par la suite, en remplaçant le système vu comme une boîte noire par un ensemble choisi d'objets de conception, on décrit l'attribution des responsabilités dynamiques, tout en conservant une traçabilité forte avec les cas d'utilisation.

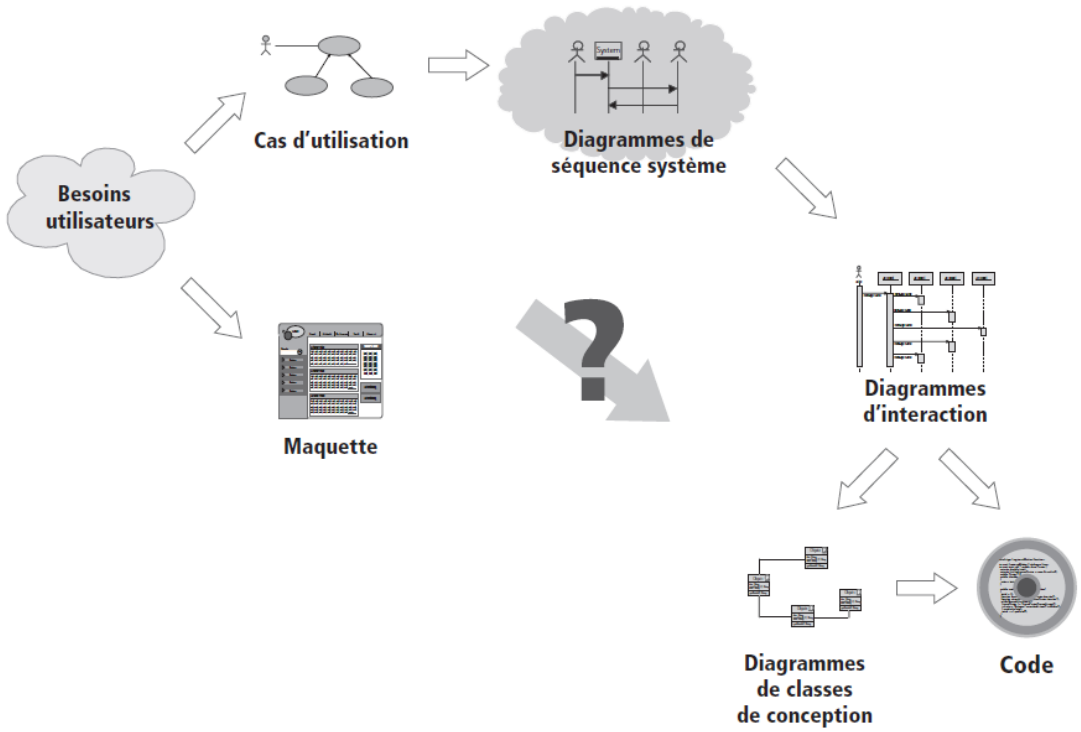


***Figure08*: Les diagrammes de séquence système fournissent le squelette des diagrammes d'interaction**

[***Source*** : Les cahiers du programmeur UML2, 4ème édition, pascal Roques, page : 18]

Partie 04

Maintenant, comment trouver ces fameuses classes de conception qui interviennent dans les diagrammes d'interaction ?

Le chaînon manquant de notre démarche s'appelle les diagrammes de classes participantes. Il s'agit là de diagrammes de classes UML qui décrivent, cas d'utilisation par cas d'utilisation, les trois principales classes d'analyse et leurs relations : les dialogues, les contrôles et les entités. Ces classes d'analyse, leurs attributs et leurs relations vont être décrits en UML par un diagramme de classes simplifié utilisant des conventions particulières.

Un avantage important de cette technique pour le chef de projet consiste en la possibilité de découper le travail de son équipe d'analystes suivant les différents cas d'utilisation, plutôt que de vouloir tout traiter d'un bloc. Comme l'illustre la figure suivante, les diagrammes de classes participantes sont particulièrement importants car ils font la jonction entre les cas d'utilisation, la maquette et les diagrammes de conception logicielle (diagrammes d'interaction et diagrammes de classes).

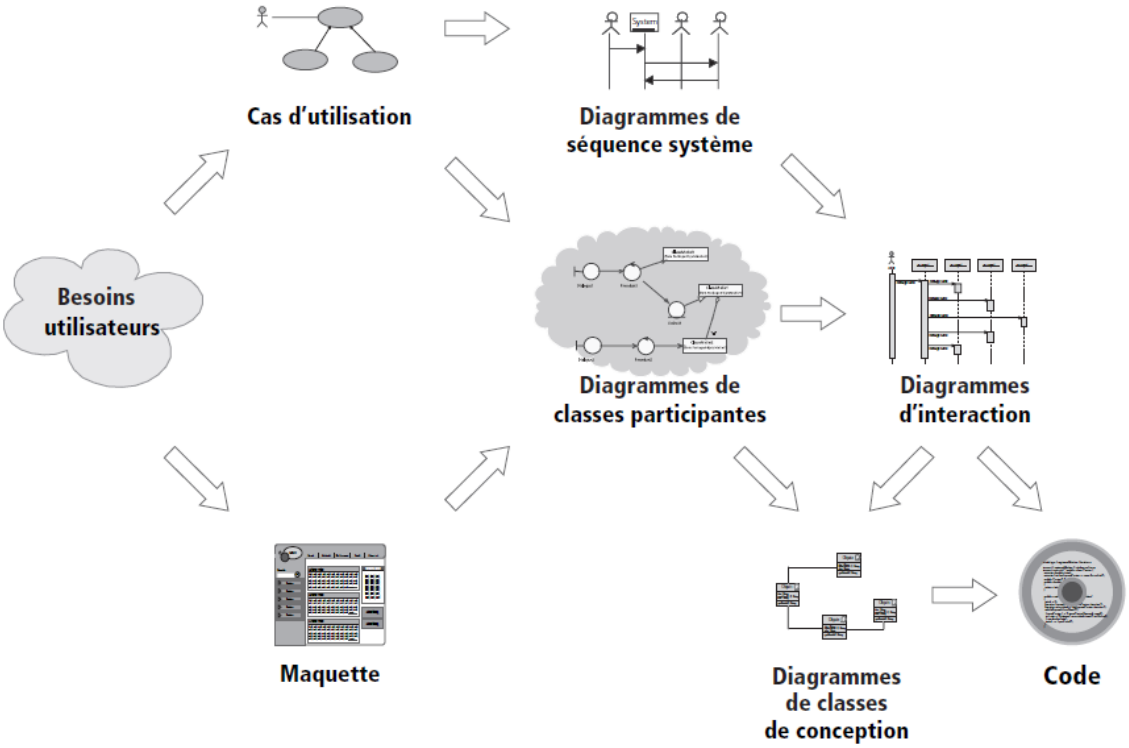


Figure08: Les diagrammes de classes participantes font la jonction entre les cas d'utilisation, la maquette et les diagrammes de conception logicielle

[***Source*** **:** Les cahiers du programmeur UML2, 4ème édition, pascal Roques, page : 19]

Partie 05

Pour que le tableau soit complet, il reste à détailler une exploitation supplémentaire de la maquette. Elle va nous permettre de réaliser des diagrammes dynamiques représentant de manière formelle l'ensemble des chemins possibles entre les principaux écrans proposés à l'utilisateur.

Ces diagrammes, qui mettent en jeu les classes participantes de type« dialogues » et « contrôles », s'appellent des diagrammes de navigation.

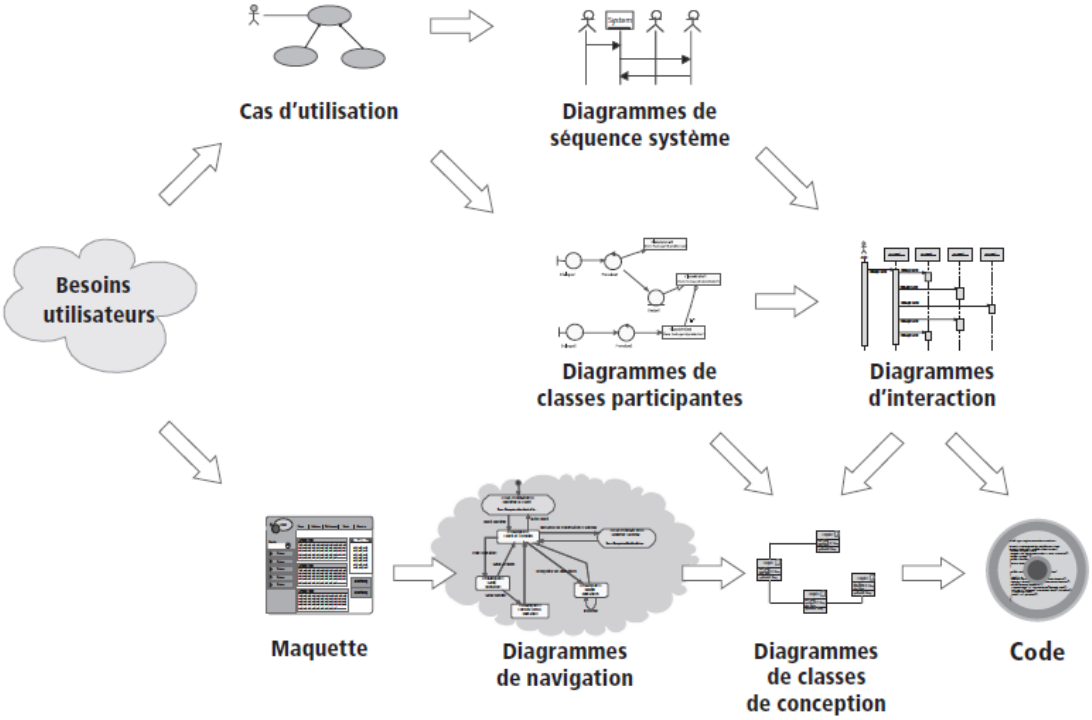


***Figure09*: Shéma Complet du processus de modelisation d'une application web**

[***Source*** **:** Les cahiers du programmeur UML2, 4ème édition, pascal Roques, page : 20]

Un schéma complet montrant comment, en partant des besoins des utilisateurs formalisés par des cas d'utilisation et une maquette, et avec l'apport des diagrammes de classes participantes, on peut aboutir à des diagrammes de conception à partir desquels on dérivera du code assez directement.[PR1]

II.5. La technologie Workflow

Le développement des démarches de réorganisation de processus est une réponse adaptée à la recherche d'optimisation des processus. Elle peut se traduire par l'utilisation d'une solution logicielle appropriée, comme un Workflow. Cette dernière permet d'être un support à une nouvelle organisation.

II.5.1. Définition

On appelle « Workflow » traduisez littéralement « flux de travail » et en français par « gestion électronique des processus métier ». La modélisation et la gestion informatique de l'ensemble des tâches à accomplir et des différents acteurs impliqués dans la réalisation d'un processus métier (aussi appelé processus opérationnel ou bien procédure d'entreprise).

Un processus métier représente les interactions sous forme d'échange d'informations entre divers acteurs tels que :

- Des humains,
- Des applications ou services,
- Des processus tiers.

De façon pratique, un Workflow peut décrire :

- Le circuit de validation,
- Les tâches à accomplir entre les différents acteurs d'un processus,
- Les délais à respecter,
- Les modes de validation…

Le Workflow sait ainsi décomposer les cas en tâches et affecter chaque tâche à un acteur selon les règles de gestion en vigueur, et en suite fournir à chacun des acteurs les informations nécessaires pour la réalisation de sa tâche.

Le Workflow permet de savoir aussi à tout moment où en est un document, donc Il permet une surveillance centralisée, parfois mal ressentie par les personnels concernés. [SM6]

II.5.2. Historique

L'industrie de l'imagerie électronique et de la gestion de la production assistée par ordinateur a été la première à réclamer une technologie qui permette l'automatisation des procédures de travail, jusqu'alors réalisées à la main.

À partir de l'année 1975 et jusqu'à 1985, la nouvelle technologie dite de WorkFlow a connu un essor important en mettant en place un système capable d'automatiser au mieux les flux de travail.

Or, le regain d'intérêt pour le génie logiciel au début des années 1990 a permis de relancer les recherches concernant les systèmes WorkFlow afin de mettre en place des systèmes plus simples à utiliser.

La figure suivante résume les différentes étapes de transfert d'informations avec workflow au sein d'une organisation.

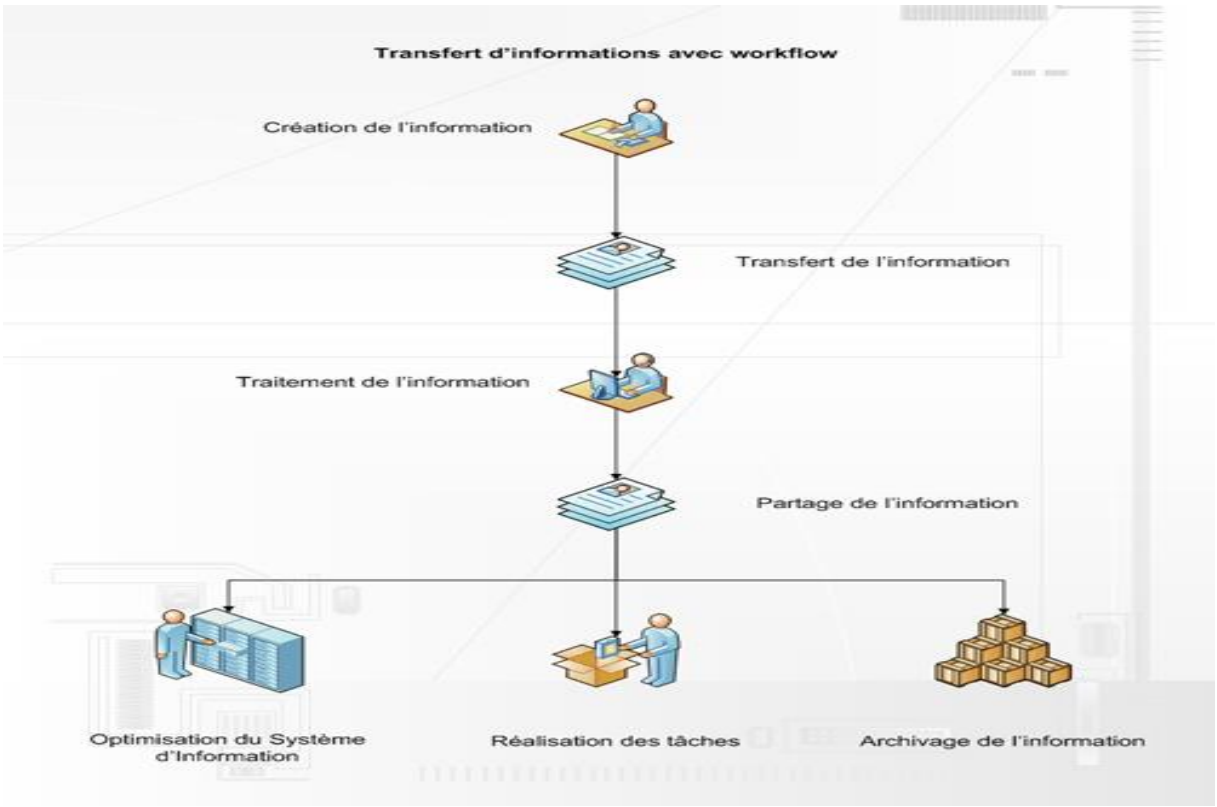


Figure10: Transfert d'informations avec workflow

[Source : Mémoire de Licence professionnelle, SEMKAOUI Amal ; AKAOUCH Mohamed 2007/2008, [SM6]]

II.5.3. Types du Workflow

Dans les applications de workflow, on distingue classiquement quatre catégories:

- ***Le workflow de production :*** (aussi appelé workflow procédural ou workflow directif), Il correspond à la gestion des processus de base de l'entreprise.
- ***Le workflow administratif:*** Il correspond à tout ce qui est routage de formulaires, basé en général sur une infrastructure de messagerie.
- ***Le workflow ad-hoc :*** Ce type de workflow est utilisé pour la gestion des procédures non déterminées ou mouvantes.
- ***Le workflow coopératif:*** Il gère des procédures évoluant assez fréquemment et liées à un groupe de travail restreint dans l'entreprise.

II.5.4. Avantages et inconvénients d'un Workflow

A. Avantages

- ***Amélioration de la productivité :*** gains de productivité réalisés par une application de workflow sont de 20 à 50% sur la part des tâches qu'elle automatise.
- ***Temps de réponse réduit :*** la réduction du temps de réponse entre la prise en compte de l'événement déclencheur d'un cas et son traitement complet est une des caractéristiques essentielles des applications de workflow.
- ***Information claire sur l'état d'avancement :*** l'application de workflow permet de renseigner avec précision sur l'état d'un cas.
- ***Sécurité accrue :*** avec une application de workflow, c'est le système qui affecte les tâches aux participants.
- ***Maîtrise de la qualité et des coûts :*** une application de workflow enregistre systématiquement le journal de tous les événements qu'elle contrôle, avec la date et l'heure, la procédure et la tâche concernée et le participant actif. Des outils de traitement de ce journal fournissent des rapports de synthèse sur les coûts et délais de traitement des tâches et des procédures.

B. Inconvénients

- Inadapté si l'on n'est pas dans un contexte « procédural ».
- ***Analyse longue et difficile :*** car la création d'un Workflow nécessite une analyse du projet. Cette analyse est souvent longue et difficile.

- ***Contraintes imposées par le logiciel :*** L'utilisation d'un Workflow nécessite que tous les participants à un projet saisissent leur état d'avancement dans chacune des tâches qu'ils ont à effectuer. Ceci est très lourd et souvent les utilisateurs n'en voient pas la nécessité. Ils ne le font donc pas régulièrement et parfois ne le font pas du tout. [SM]
- Lors de l'utilisation de l'application, les utilisateurs peuvent rencontrer différents problèmes liés à leur matériel ou à un dysfonctionnement du workflow. Il est donc nécessaire de mettre en place un système d'assistance et de résolution de problèmes. (Par exemple, des problèmes d'impression des documents du workflow à partir du navigateur web, ou bien des indisponibilités réseau. [SM6]

II.5.5. La conduite d'un projet de workflow

Un projet de workflow ressemble pour une part à un projet d'informatique. Il suppose une analyse détaillée des procédures à prendre en compte et implique parfois des développements qui font appel aux techniques de programmation les plus traditionnelles.

Il ressemble aussi à un projet bureautique avec une importance toute particulière à donner aux aspects installation du réseau et des stations, formation des utilisateurs et organisation du travail en fonction du nouvel outil.

II.5.6. Le moteur workflow

Le moteur de workflow est le dispositif logiciel permettant d'exécuter une ou plusieurs définitions de workflow. Par abus de langage, on peut appeler ce dispositif logiciel tout simplement "workflow".

Le moteur de workflow est l'outil permettant de modéliser et d'automatiser les processus métiers de l'entreprise. Là on peut parler plus spécifiquement du Workflow en tant qu'objet pouvant être décrit par un langage descriptif dans un fichier informatique, qu'une application adaptée peut alors interpréter et exécuter. Ainsi on peut automatiser un processus de travail.

Les entreprises doivent donc faire leur choix parmi un grand nombre de logiciels de workflow. Aucun produit ne remplit toutes les conditions. Il est donc important que les utilisateurs définissent leurs besoins spécifiques et leurs objectifs principaux.

Les moteurs de workflow utilisent les techniques les plus modernes (réseaux, base de donnée, bureautique, interfaces homme machine, technologies objet) tout en exploitant au mieux les applications existantes pour faire à l'entreprise un bon en avant dans l'exploitation des outils modernes de traitement de l'information au profit d'une meilleure productivité et d'une qualité de service accrue. Ces retours ne se matérialiseront pourtant que si tout l'effort de préparation est correctement orienté pour tenir compte des objectifs de l'entreprise et si la conduite du projet est centrée sur le suivi des progrès et leur recentrage sur les cibles critiques.

II.6. Conclusion

Passer des besoins des utilisateurs à un code source n'est généralement pas aisée. Vu la nécessité de se servir de méthodes pour la modélisation, nous avons adopté une démarche centrée autour de l'utilisateur et qui est inspirée de la philosophie proposée par ***Pascal Roques*** dans l'article « une démarche de modélisation agile pour passer des besoins des utilisateurs au code ». Une méthode agile est une approche itérative et incrémentale qui est menée dans un esprit collaboratif avec juste ce qu'il faut de formalismes.

Le travail proposé dans ce mémoire va consister à développer un outil permettant de supporter le travail de l'équipe de conception de n'importe quel projet d'informatisation en s'inspirant de la démarche basée sur UML et qui permet d'obtenir le plus efficacement possible un code informatique opérationnelle, complet, testé et qui répond parfaitement aux besoins dans un cadre de travail parallèle collaboratif grâce a la technologie Workflow.

Ce chapitre a présenté un état de l'art sur les différents concepts et sur lesquels va reposer l'étude conceptuelle de notre outil dans le chapitre suivant.

Chapitre III

III.1. Introduction

La phase la plus importante dans la réalisation de notre travail est incontestablement le choix d'une méthodologie d'un raisonnement pour atteindre nos objectifs.

Rappelons que notre travail consiste en la réalisation d'un « outil d'aide a la conception agile en utilisant la technologie Workflow». Ce dernier doit supporter la démarche de ***Pascal Roques*** et le travail d'une équipe de développement de projets informatiques, et permettre la communication entre les membres de l'équipe où le client est impliqué tout le long du développement.

La présentation de notre étude conceptuelle sera faite en deux points pour chaque fonctionnalité du système:

- ✓ Un scénario décrivant le raisonnement et la manière de procéder.
- ✓ Un diagramme d'activité illustrant ce raisonnement.

III.2. Les acteurs

Notre système est composé de six (06) acteurs principaux :

III.2.1. Le chef de projet

C'est l'acteur de base de notre système, il doit être au courant de tout, et tout ne se fait qu'après sa validation.

III.2.2. Le client :

Déclencheur du système, peut être interne (service, département) ou externe, cet acteur est présent tout le long du développement, il collabore et fourni un feed-back continu sur l'adaptation du projet a ces attentes jusqu'à livraison finale.

III.2.3 l'équipe de prise en charge du projet d'informatisation :

A ce niveau quatre équipes sont définies, elles sont formées de binômes ou plus en se basant sur le principe d'agilité.

Pour chaque équipe, chaque module terminé, testé et validé déclenche le travail de l'autre équipe. C'est du travail parallèle et Les tests se font au fur et à mesure, le passage d'une

étape à une autre ne se fait qu'après validation du client ainsi que le chef de projet. Le travail se fait d'une manière modulaire.

III.2.3.1 L'équipe de pilotage

C'est une équipe qui réalise l'intermédiaire entre le client et le chef du projet. Elle est formée de deux ou trois membres ou le client est impliqué. Cette équipe contient éventuellement un commercial et une personne qualifiée en la communication.

III.2.3.2 L'équipe de collecte des besoins

Identifiée par le chef du projet, c'est l'équipe chargée simultanément d'établir les maquettes et les cas d'utilisation montrant les fonctionnalités système à développer plus des rapports.

III.2.3.3 L'équipe de conception

En se basant sur la démarche proposée dans notre mémoire et la modélisation basée sur UML agile, cette équipe est chargée d'élaborer les diagrammes (avec rapport) dans l'ordre préconisé par la démarche:

A partir des cas d'utilisation établir :

- Le diagramme de séquence système.
- Le diagramme des classes participantes.

A partir des maquettes établir :

- Le diagramme de navigation.

A partir des diagrammes « séquence et de classes participantes » établir :

- Le diagramme d'interaction.

A partir des diagrammes « Interaction, classes participantes et navigation » établir :

- Le diagramme de classe de conception.

III.2.3.4 L'équipe d'implémentation

C'est l'équipe chargée du passage au code à partir des diagrammes élaborés par l'équipe de conception et l'intégration des modules développés séparément avec rédaction de rapport d'implémentation.

III.3. L'espace virtuel commun

Afin qu'il ait un travail parallèle collaboratif et une circulation automatique de l'information entre les acteurs, notre outil dispose d'un espace de travail commun virtuel permettant aux équipes d'un projet P d'afficher leurs travaux en les diffusant vers tous les autres acteurs.

En se basant sur le principe de l'agilité, cet espace permet la communication et la collaboration entre les différents acteurs de développement d'un projet d'informatisation donné, ou toujours le client est membre principal dès le lancement du projet jusqu'à livraison finale et est une partie prenante du développement par sa présence dans l'espace.

Chaque acteur du système reçoit les publications des autres acteurs et peut depuis sa session critiquer ou poster des observations ou des appréciations pouvant améliorer la qualité du système et contrôler l'avancement du projet.

III.4. Conceptualisation de l'outil

Le but essentiel de la conceptualisation est de comprendre et structurer les objectifs que nous voulions atteindre, non pas en cherchant l'exhaustivité, mais en clarifiant, filtrant et organisant les besoins.

Une fois identifiés et structurés, ces besoins :

- Définissent le contour de l'outil à modéliser.
- Permettent d'identifier les fonctionnalités principales.

III.4.1. Pourquoi les diagrammes d'activités UML?

La description de flux de travail (Workflow) est souvent représentée par le diagramme d'activités UML vu que ce dernier est un diagramme comportemental permettant de représenter le déclenchement d'événements en fonction des états du système.

Le diagramme d'activité est une représentation proche de l'organigramme, la description d'une fonctionnalité par un diagramme d'activité correspond à sa traduction algorithmique.

Le diagramme d'activités présente une vision macroscopique et temporelle du système modélisé.

III.4.2. Scénarios et diagrammes d'activités

III.4.2.1.a. Scénario général

1. Le client se présente a l'entreprise avec une demande d'un projet d'informatisation.

NB:

- La demande se fait verbalement et sera détaillée avec des descriptions en pièces jointes.
- Le client est en relation directe avec l'équipe de pilotage pour le traitement de sa demande.
- L'équipe de pilotage prend le soin de rédiger un document organisé qui sera attaché en pièces jointes pour l'envoyer au chef de projet.
- A la réception du document envoyé par l'équipe de pilotage, le chef de projet, après une étude de faisabilité du sujet d'informatisation est en face a trois réactions :
    1. Acceptation.
    2. Réorientation (avec causes).
    3. Demande de plus d'information.
- En cas de réorientation le chef de projet donne les causes, la réorientation se fait en cas ou le sujet est hors de la maitrise de l'équipe d'informatisation.
- Le chef de projet est en état d'indécision s'il ya manque d'informations ou de détails, dans ce cas il recontacte l'équipe de pilotage pour plus d'informations.
- En cas d'acceptation, l'équipe de pilotage après négociation avec le client établie le cahier de charge qui doit être validé et par le chef de projet et par le client.

2. Dès réception du cahier de charge par le chef de projet, il commence à identifier les équipes et leur affecter les taches.
3. Trois (03) équipes sont identifiées :
    - Collecte de besoin.
    - Conception.
    - Implémentation.
4. Le projet se lance par le lancement du travail parallèle des équipes suivant la démarche adoptée.

III.4.2.1.b. Diagramme d'activités général

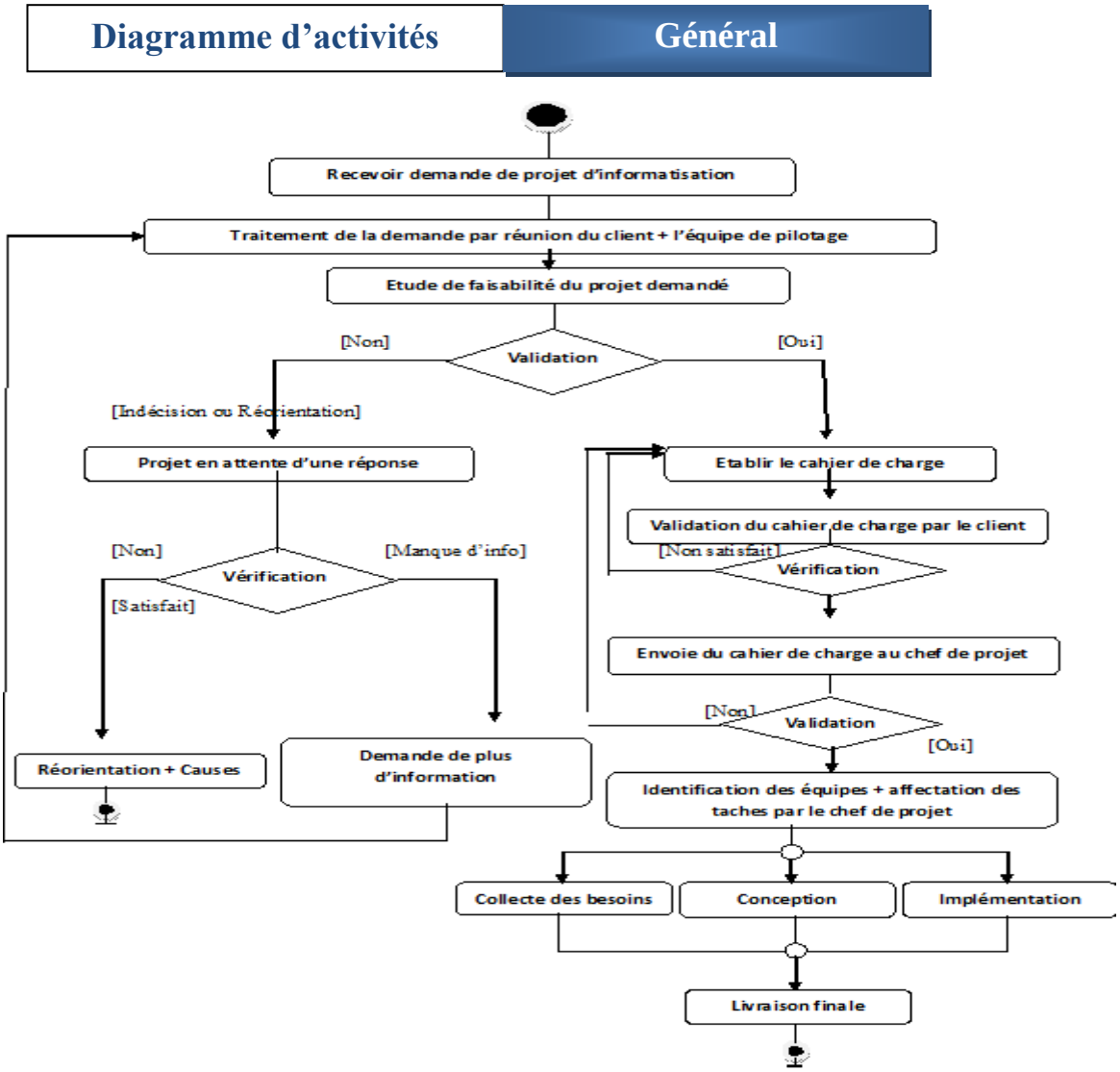


Figure 11: Diagramme d'activités général décrivant le raisonnement de conception de l'outil

III.4.2.2.a. Scénario « Etablir cahier de charge »

C'est une tache réalisée par l'équipe de pilotage ou le client est toujours impliqué. Celle tache consiste en la récupération des besoins du client puis restructuration et finalisation. Le cahier de charge va comporter les critères suivants : (Délai, Cout, Budget) et les besoins des utilisateurs.

NB :

- Le délai global de projet est divisé par le chef de projet en sous délais et chaque équipe est dès le début informé du délai dont elle ne doit pas dépasser.
- Dans le cas de dépassement de délai de réalisation de projet prévu dans le cahier de charge, une réunion sera établie par le chef de projet avec l'équipe pour gérer et régler le problème. L'outil ne gère pas ce dépassement.

III.4.2.2.b. Diagramme d'activités « Etablir cahier de charge »

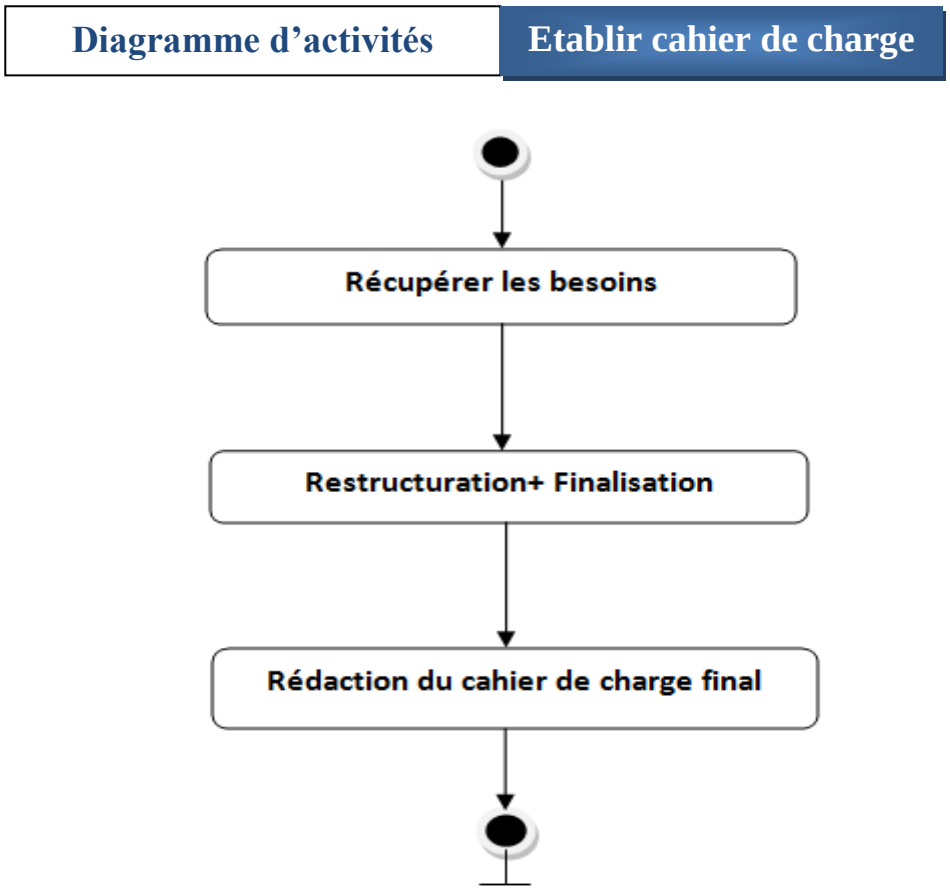


Figure12: Diagramme d'activités de la fonction «Etablir cahier de charge»

III.4.2.3. Scénario «validation du cahier de charge par le client»

1. Apres la rédaction du cahier de charge, le client doit le valider.
   - ***Si*** satisfait le cahier de charge sera envoyé au chef de projet.
   - ***Sinon*** rétablissement du cahier conformément aux remarques faites par le client.
2. Une deuxième validation du cahier de charge doit se faire par le chef de projet.
   - ***Si*** satisfait il entame l'identification des équipes du développement.
   - ***Sinon*** Rétablissement du cahier de charge.

III.3.2.4.a. Scénario « Identification des équipes »

Cette identification se fait par le chef de projet suivant les étapes si après :

- Choisir les employés adéquats.
- Former les équipes (des équipes en binôme généralement suivant le principe de l'agilité).
- Affecter les taches.

NB :

- Les membres de chaque équipe sont informés par leur désignation, de même les autres équipes sont au courant des employés désignés dans les autres équipes.

III.4.2.4.b. Diagramme d'activités « Identification des équipes »

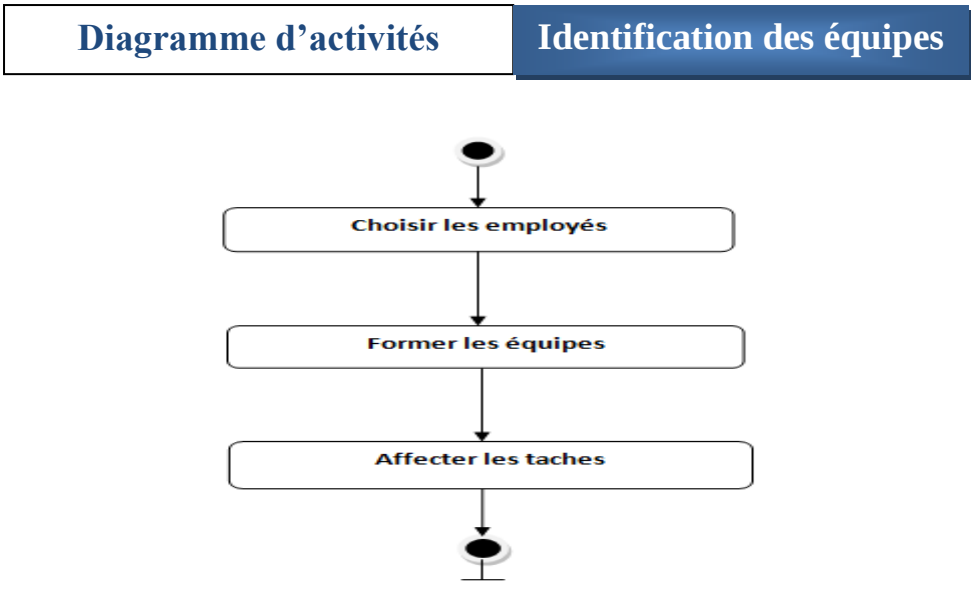


Figure13: Diagramme d'activités de la fonction « Identification des équipes »

III.4.2.5.a. Scénario « collecte des besoins »

1. Apres réception du cahier de charge par le chef de projet comme par le client, L'équipe de collecte commence son étude et analyse les besoins par rapport au sujet.
    - ***Si*** manque d'information => Demande de plus d'information au chef de projet, a son tour il contacte l'équipe de pilotage qui aussi contacte le client pour accomplir le manque.
    - ***Sinon*** rédaction de la première maquette et établissement du cas d'utilisation + rédaction de la documentation (rapport) puis envoie au chef de projet.
2. Le chef de projet doit valider chaque maquette et cas d'utilisation.
    - ***Si*** non satisfait refaire les maquettes et cas d'utilisation.
    - ***Sinon*** Affichage des maquettes et cas d'utilisation dans l'espace commun.
3. Une réception éventuelle des observations, critiques, ou commentaires depuis n'importe quel acteur du système.
4. Le chef de projet dans son espace de travail traite toutes les observations et critiques publiées dans l'espace commun, il prend en considération ce qui est important et néglige le reste.
    - ***S***'il existe des observations pouvant entrainer des changements, le chef de projet renvoie ces mêmes observations à l'équipe de collecte pour revoir des nouvelles maquettes et cas d'utilisation adaptés aux changements. L'affichage dans l'espace virtuel se fait une seule fois et chaque acteur ne peut poster qu'un seul commentaire ou observation.
    - ***Sinon*** Envoyer les maquettes et les cas d'utilisations validés a l'équipe de conception.

III.4.2.5.b. Diagramme d'activités « collecte des besoins »

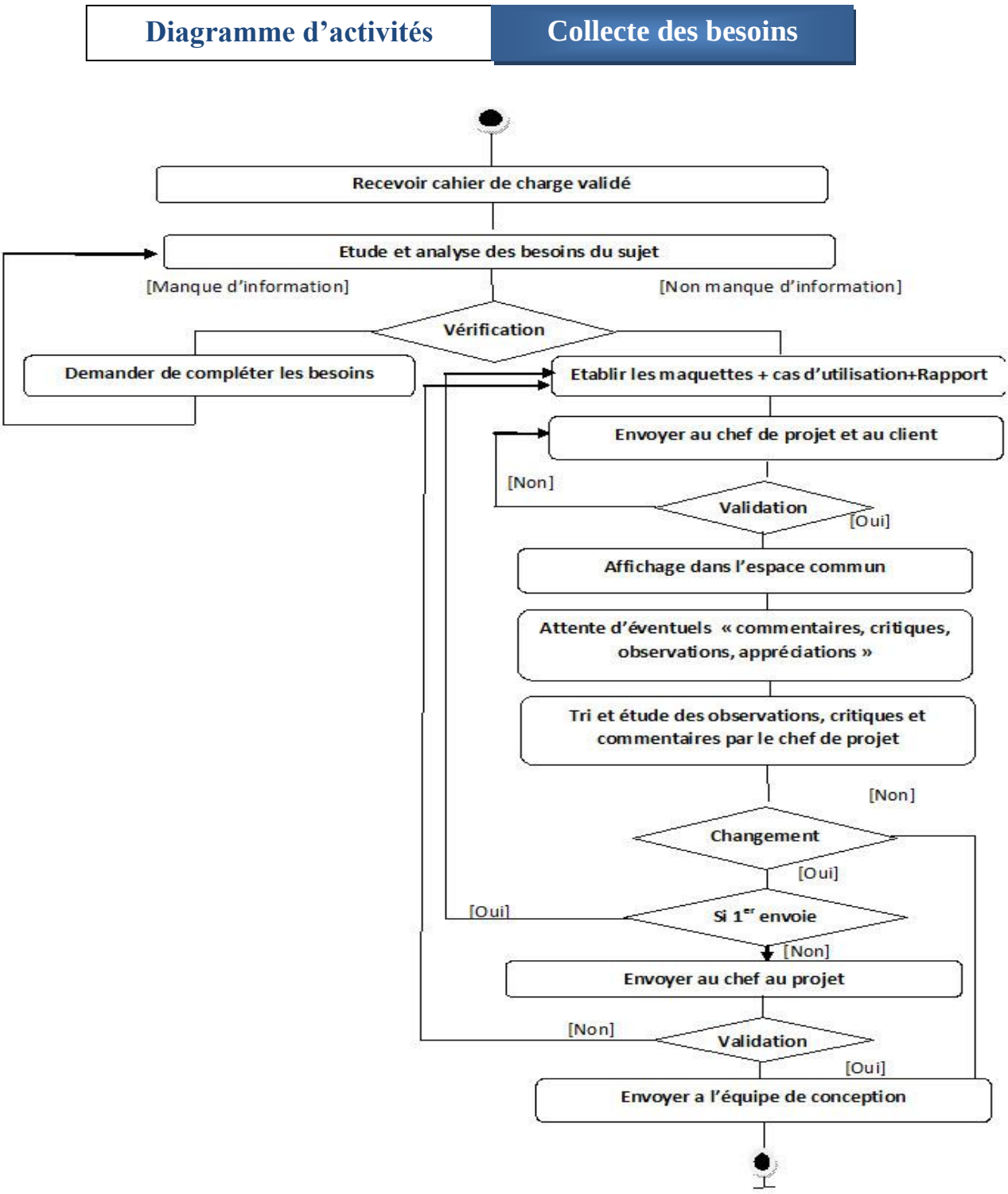


Figure14: Diagramme d'activités de la fonction « Collecte des besoins »

III.4.2.6.a. Scénario « conception »

1. L'équipe de conception reçoit les maquettes et les cas d'utilisation validés.
2. Suivant la démarche :

   A partir des cas d'utilisation l'équipe de conception génère :

   - Le diagramme de séquence système.
   - Le diagramme de classes participantes.

   → Ces deux diagrammes seront affichés dans l'espace virtuel pour d'éventuels critiques ou changements.

A partir des maquettes l'équipe de conception génère :

- Le diagramme de navigation.

→ Ce diagramme sera affiché dans l'espace virtuel pour d'éventuels critiques ou changements.

A partir du diagramme de séquence et celui de classes participantes l'équipe de conception génère :

- Le diagramme d'interaction.

→ On n'a pas besoin de poster ce diagramme virtuel car il est généré des diagrammes déjà validés.

A partir des diagrammes : navigation, interactions et classes participantes l'équipe de conception génère :

- Le diagramme de classes de conception.

→ On n'a pas besoin de poster ce diagramme virtuel car il est généré des diagrammes déjà validés.

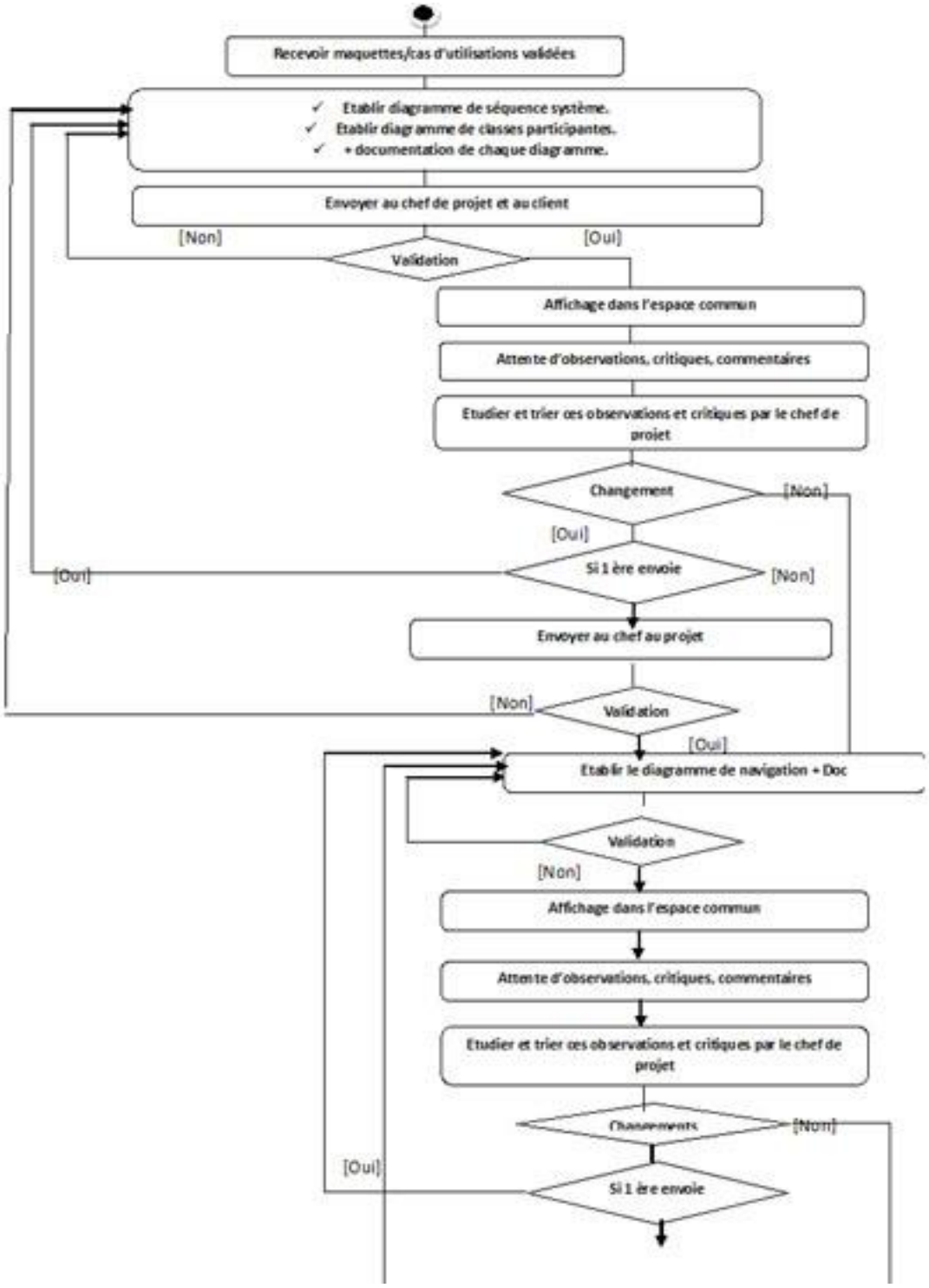
NB :

- Chaque diagramme élaboré par l'équipe de conception est accompagné par une documentation (rapport).
- Chaque diagramme fourni doit être obligatoirement validé par le chef de projet.
- Attente d'éventuels (commentaires, critiques, observations ou appréciations) de la part des différents acteurs du système pour tous les diagrammes sauf les diagrammes interaction et classes de conception.
- Si nécessité de changement ce dernier entraine des modifications.

- L'affichage des diagrammes dans l'espace virtuel se fait une seule fois lors du premier établissement des diagrammes (séquence système, classes participantes, navigation).

III.4.2.6.b. Diagramme d'activités « conception»

Diagramme d'activités | **Conception**



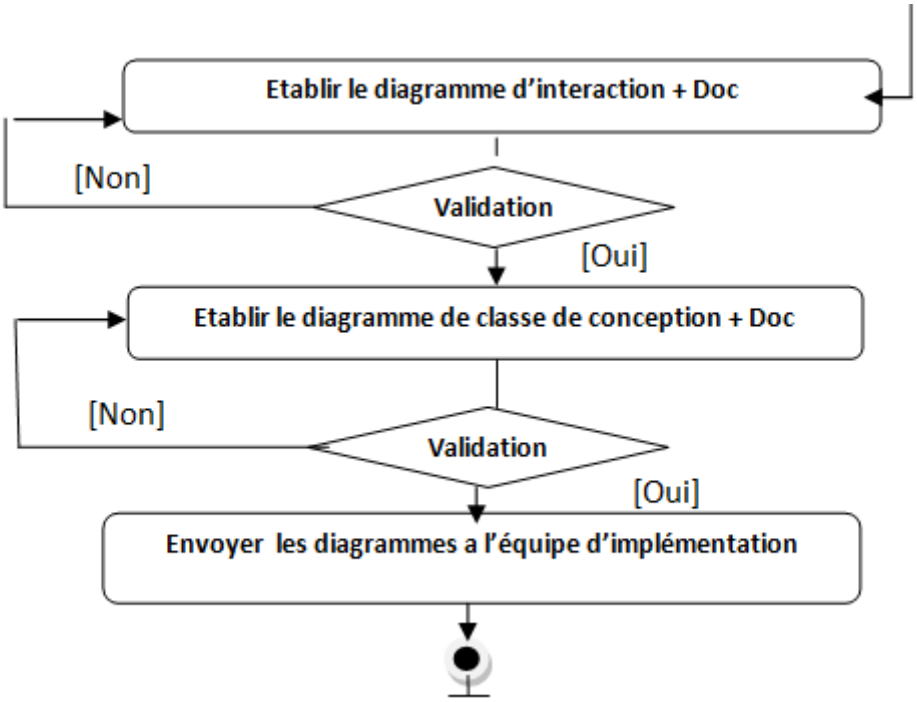


Figure15: Diagramme d'activités de la fonction « Conception »

III.4.2.7.a. Scénario «Implémentation »

- L'équipe d'implémentation reçoit de la part de l'équipe de conception le modèle conceptuel par module (fonction) validé.
- Lancement de la programmation (passage des diagrammes au code) par fonction.
- Tester chaque fonction (module) a part. Le test se fait avec le client.
- Rédaction de la documentation au fur et a mesure.
- Vérification par le chef de projet et le client:
  - Si satisfait(s) alors fin de réalisation.
  - Sinon Retour a la programmation.

- Intégration des modules implémentés testés et validés chacun a part pour aboutir au produit final.

NB :

- Le client n'a pas le droit de voir ou commenter le code. Il intervient simplement dans la phase des tests des modules et d'intégration.

III.4.2.7.b. Diagramme d'activités «Implémentation»

Diagramme d'activités | **Implémentation**

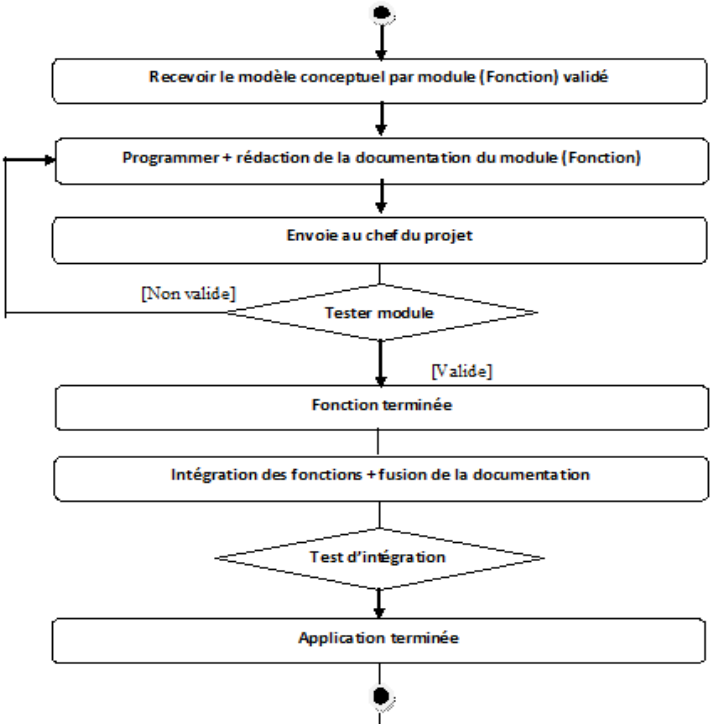


Figure16: Diagramme d'activités de la fonction « Implémentation »

III.4.2.8.a. Scénario «Livraison finale »

1. Le chef de projet récupère le produit complet.
2. Envoie du produit final a l'équipe de pilotage qu'a son tour l'envoie au client pour finaliser la livraison.
3. Etablir contrat de maintenance et de formation.
4. Payement et facturation.

III.4.2.8.b. Diagramme d'activités «Livraison finale »

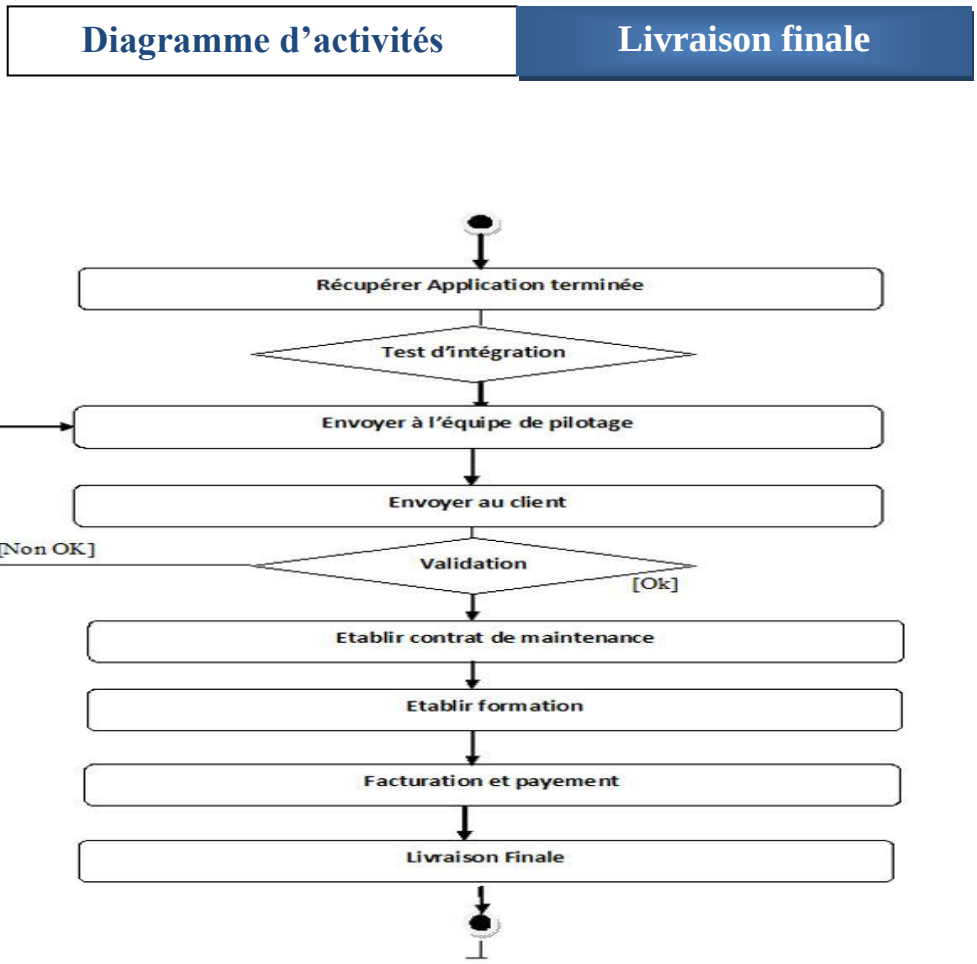


Figure17: Diagramme d'activités de la fonction « Livraison finale »

III.5. Conclusion

A ce niveau, on a modélisé les aspects informatiques du système, sans pour autant rentrer dans les détails d'implémentation.

Notre étude conceptuelle repose sur la représentation d'une suite de tâches et d'opérations effectuées par les différents acteurs du système en se basant sur le principe de la démarche citée dans le « chapitre2 ». Les diagrammes d'activités UML ont décrit le raisonnement suivi pour développer notre outil d'aide à la conception agile.

Chapitre IV

IV.1. Introduction
IV.2. Outils de développement
IV.3. Présentation de l'application
IV.4. Perspectives
IV.5. Conclusion

IV.1. Introduction

Notre outil d'aide à la conception agile est basé principalement sur les notions de la démarche UML agile et celles du travail collaboratif suivant le workflow permettant la communication et la circulation automatique de flux d'informations.

IV.2. Outil de développement

L'important des activités d'implémentation est la traduction de la conception en code source. On utilisé un outil open source pour réaliser notre outil d'aide a la conception agile.

IV.2.1. BonitaOpenSolution

BonitaOpenSolution est un système de gestion de flux de travail libre et complet. Il est composé de trois modules intégrés:

- Bonita Studio
- Bonita Forms
- Bonita User Experience

A. ***Bonita Studio*** est un éditeur basé sur la plateforme eclipse qui permet de dessiner des flux de travail conformes à la notation BPMN (Business Process Modeling Notation)[1]. Il offre des fonctionnalités avancées tel que la définition des participants aux flux, l'affectation de priorités aux tâches, la définition de variables, la connexion à des

systèmes externes ainsi que la définition et la personnalisation des interfaces utilisateur.

B. ***Bonita Forms*** génère les interfaces utilisateur requises pour l'échange de données entre participants humains.

C. ***Bonita User Experience*** fournit une interface similaire à celles des boîtes de courriel. L'interface permet à l'administrateur de superviser les processus et aux participants de consulter les tâches qu'ils doivent effectuer. Bonita Open Solution constitue donc une solution intuitive et facile d'accès pour les experts métier. Elle est distribuée sous licence GPLv2[2]. [TB]

IV.2.2. Utilisation de l'outil Bonita

Bonita Open solution peut être utilisé pour créer des flux de données complexes (workflow) comme pour d'autres usages tels que la création des feuilles de calcul … etc. c'est un outil Open source et téléchargeable sous la licence GPL.

IV.2.3. Choix de BonitaOpenSolution

Nous avons travaillé avec l'outil BonitaOpenSolution malgré qu'il ne nous offre pas toutes les fonctionnalités pour la simple raison de sa disponibilité en open source, et vu que notre université ne dispose pas d'un outil payant pouvant jouer au mieux le rôle de réalisation de notre outil d'aide à la conception agile.

IV.2.4. Base de données

Notre outil dispose d'une base de données pour garder l'historique des clients ainsi que les informations relatives a ces derniers.

---

[1] : BPMN Business Process Model and Notation est une notation graphique standardisée pour modéliser des procédures d'entreprise ou des processus métier.

[2] : GPLV2 GNU General Public License est une licence qui fixe les conditions légales de distribution des logiciels libres du projet GNU.

Le schéma de classe de cette base de données est le suivant :

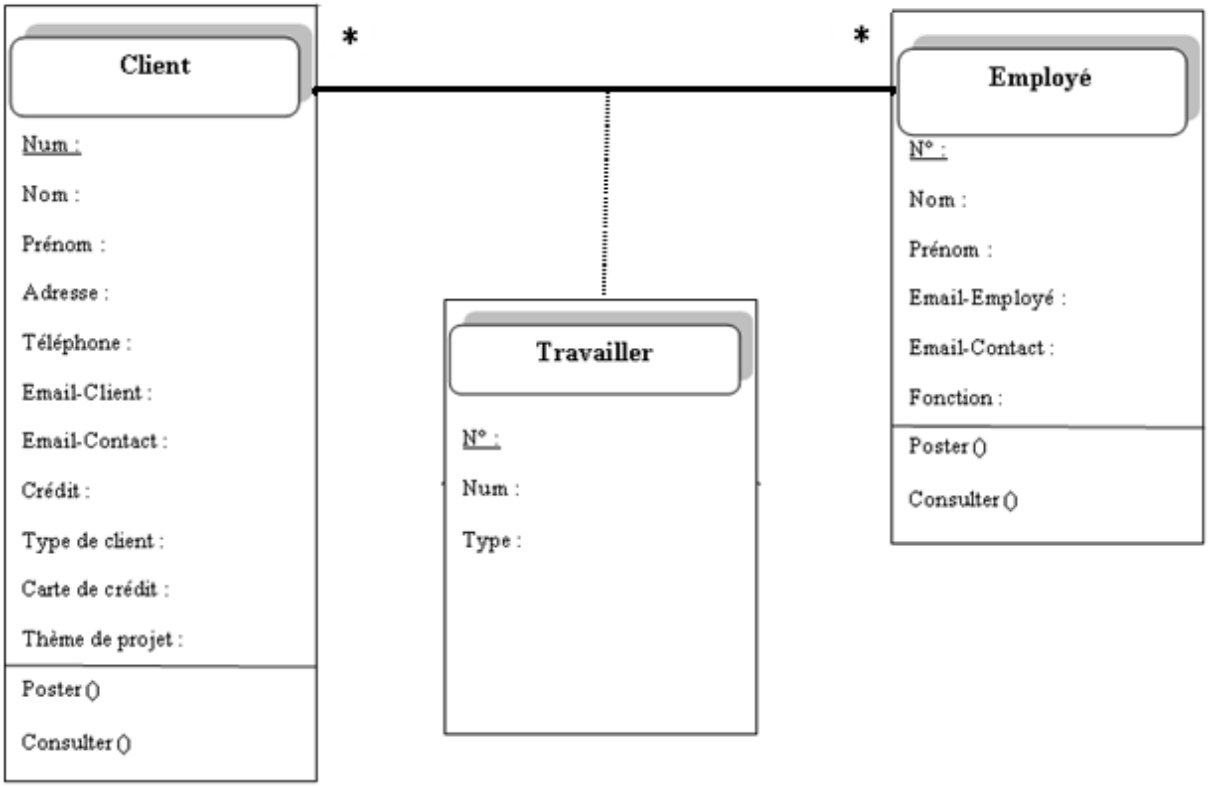


Figure19 : schéma de la base de données utilisée

IV.3. Présentation de l'outil

Le schéma suivant résume le raisonnement sur lequel est basé notre outil :

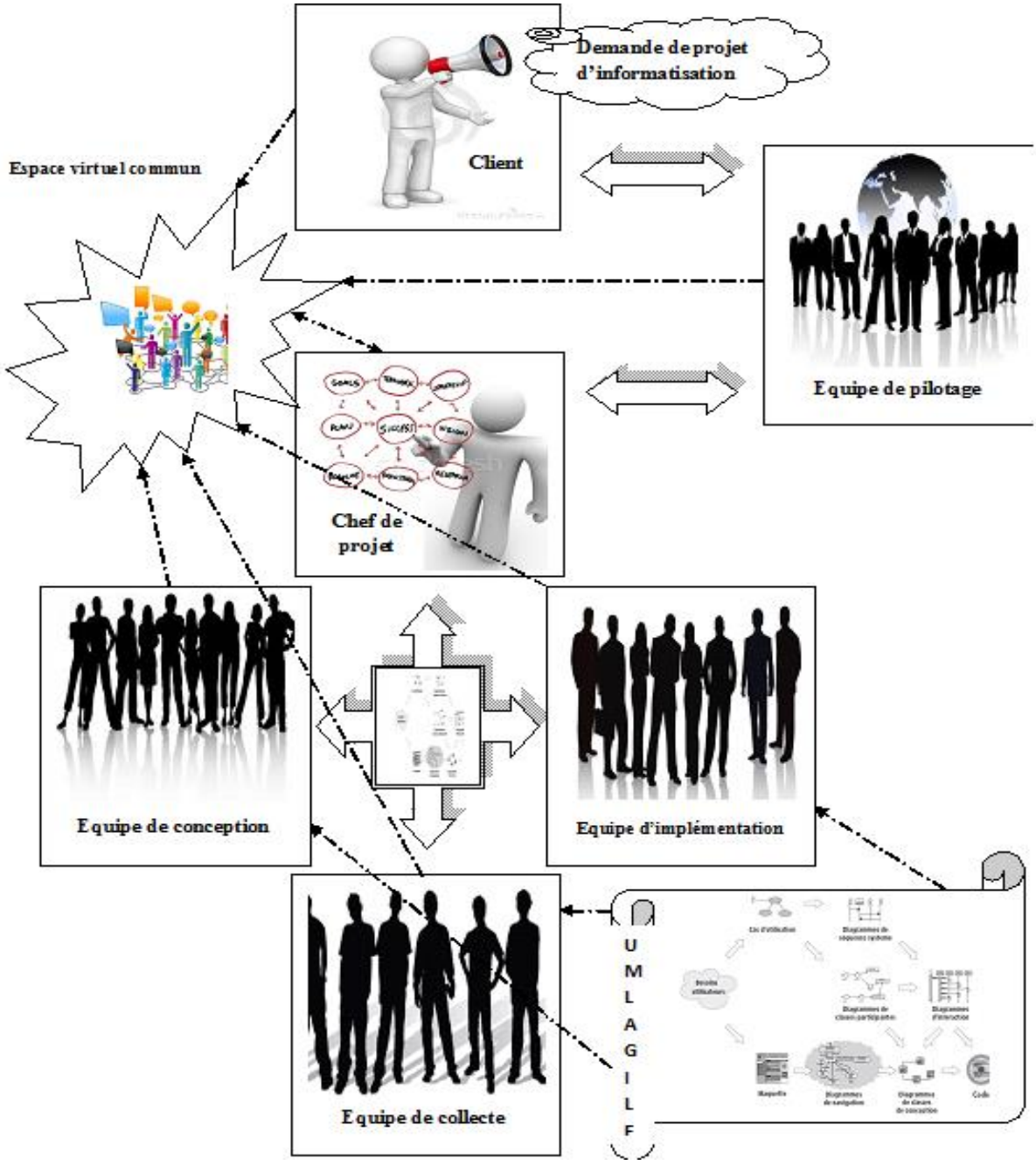


Figure 20: schéma résumant le principe de notre outil d'aide a la conception agile

Dans ce qui suit nous allons présenter un exemple de la version finale de notre outil d'aide à la conception agile en essayant d'expliquer le principe de fonctionnement.

IV. 3.1. Page d'accueil : « authentification »

L'identification de chaque acteur de notre système est un besoin et elle est loin d'être une option, puisque on ne peut pas autoriser n'importe qui à se connecter que depuis sa session. Son importance réside aussi dans la définition des opérations et des taches que l'utilisateur identifié pourra effectuer ainsi que l'ensemble des données dont il aura l'accès.

Figure21 : page authentification

IV. 3.2. Page recevoir demande d'informatisation :

La demande d'informatisation faite par le client déclenche le système, après identification de l'équipe de pilotage, un formulaire s'affiche ou le pilotage effectue une saisie des coordonnées de client ainsi que les détails du projet pour enfin l'envoyer au chef de projet.

Figure22 : page Recevoir demande d'informatisation

IV.3.3. Page « Arrivée d'un nouveau client »

Dès la connexion du chef de projet depuis sa session, une page contenant les informations sur le projet et le client s'affiche, et c'est ce qui fait que la circulation est automatique entre les acteurs.

Pour permettre au chef de projet ou de « réorienter, accepter ou demander plus d'information » l'outil génère :

Figure23: page Recevoir demande d'informatisation

NB :

Nous n'avons présenté qu'une simple partie du fonctionnement de l'outil dans ce chapitre, Le produit complet sera présenté devant le jury, et un manuel et annexe d'utilisation de notre outil sera attaché ace document.

IV.4. Quelques problèmes et Perspectives :

Notre travail constitue une partie d'un travail global visant a réaliser un outil performant d'aide a la conception agile, durant ce travail nous avons rencontré quelques problèmes et ceci revient a l'utilisation d'un outil open source qui est restreint et n'offre pas toutes les fonctionnalités au contraire d'un outil payant, certains ont été surmontés et nous indiquons d'autres en perspectives qui restent a réaliser :

IV.4 .1 Problèmes résolus :

1. **a. Problème1 :**

L'existence de plusieurs acteurs implique l'existence de plusieurs taches et formes, ce qui rend le déroulement d'exécution difficile.

1. b. Solution :

Le message de confirmation guide l'utilisateur de comprendre au mieux le fonctionnement de notre outil d'aide à la conception agile.

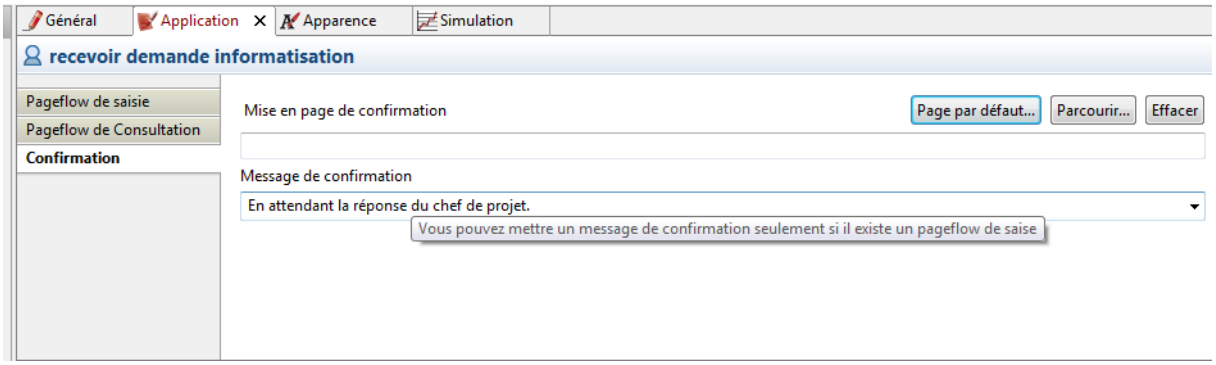


Figure24: message de confirmation en Bonita

2. a. Problème2 :

Distinction de différence entre POOL et LAN.

2. b. Solution :

Pool peut contenir plusieurs Lan, Lan est spécifier pour un seul acteur, il s'agit d'une très bonne méthode pour organiser le travail.

3. a. Problème3 :

Les données déclarées dans la partie application d'une tache ne sont pas reconnues dans tous le programme.

3. b. Solution :

Les données doivent êtres déclarées dans le pool général pour qu'elles soient reconnues dans le programme entier (Tache : locale, Pool : globale).

4 .a. Problème4 :

Comment nous pouvons faire le contact avec l'extérieur et ; depuis l'extérieur :

Par exemple : remplissage d'une base de donnés.

Extraire des données depuis une base de données. (Access, XML)

Envoie automatique des courriers électroniques.

Exchange.

Messagerie……..

4. b. Solution :

Utilisation des connecteurs de Bonita.

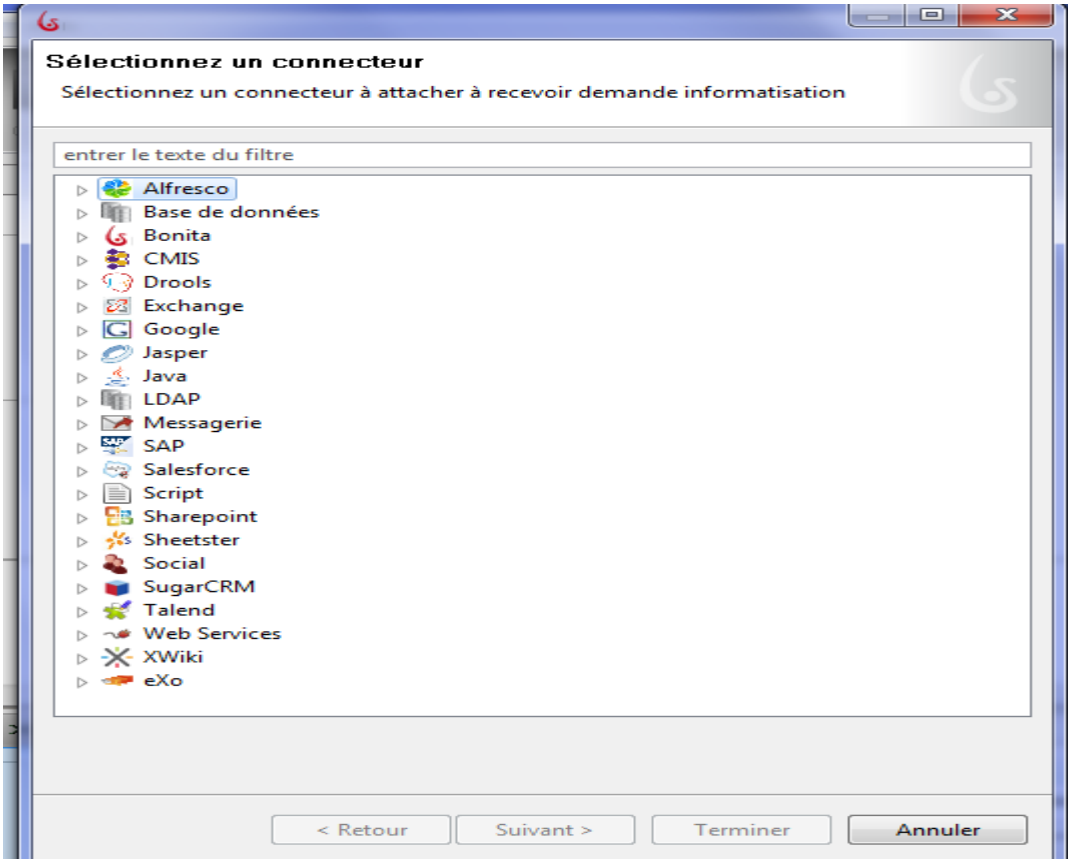


Figure25 : Utilisation des connecteurs de Bonita

Nb :

Smtp.gmail.com : envoi automatique d'un email sans revenir au compte, leport utilisé est : 465.

POS.gmail.com : recevoir des emails.

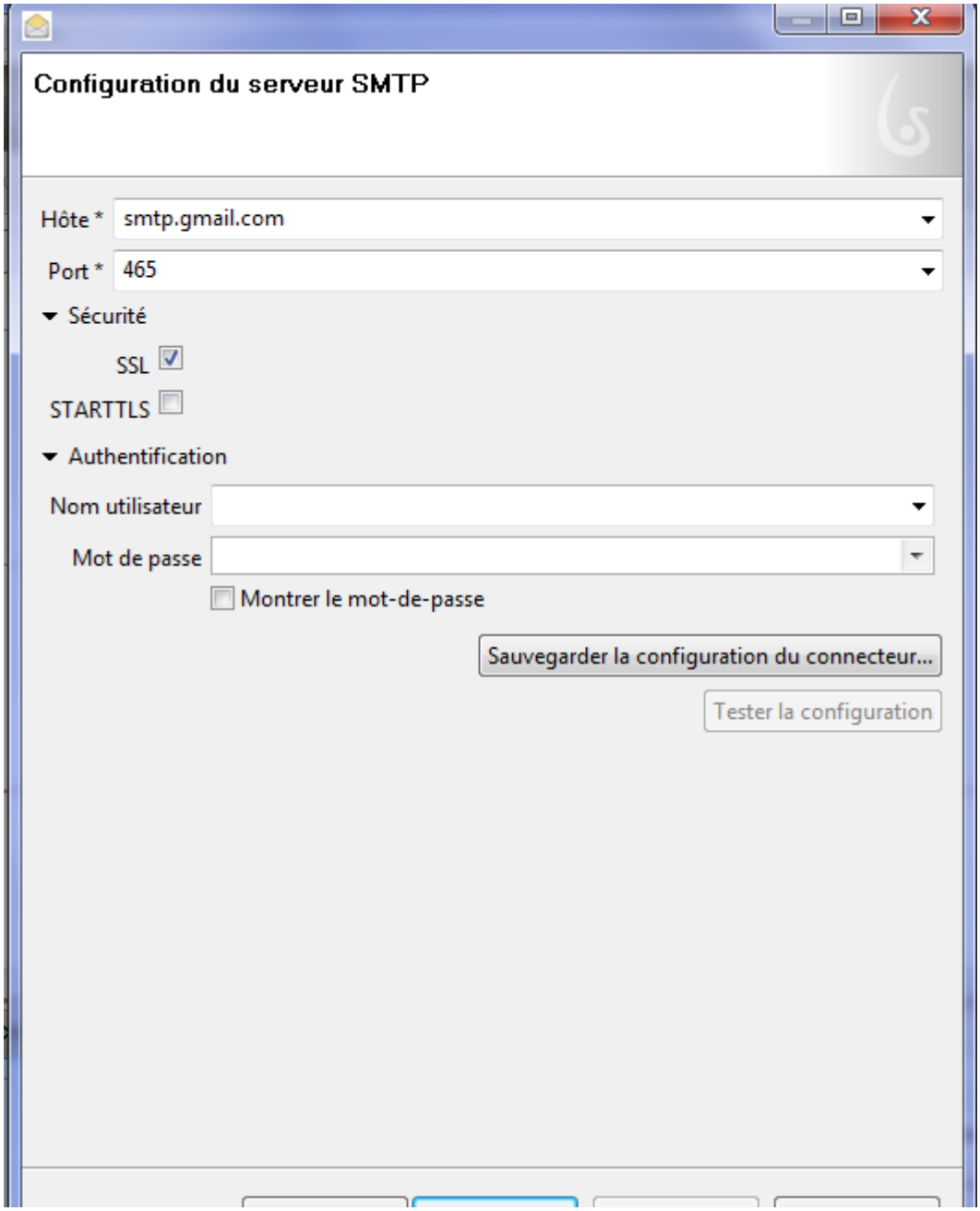


Figure 26: Configuration du serveur SMTP

5 .a. Problème5 :

Apprentissage de l'éditeur de programmation, Par exemple l'insertion des clients dans une base données Access nécessite un apprentissage de l'éditeur ***grouvy*** qui a une syntaxe très sensible aux erreurs.

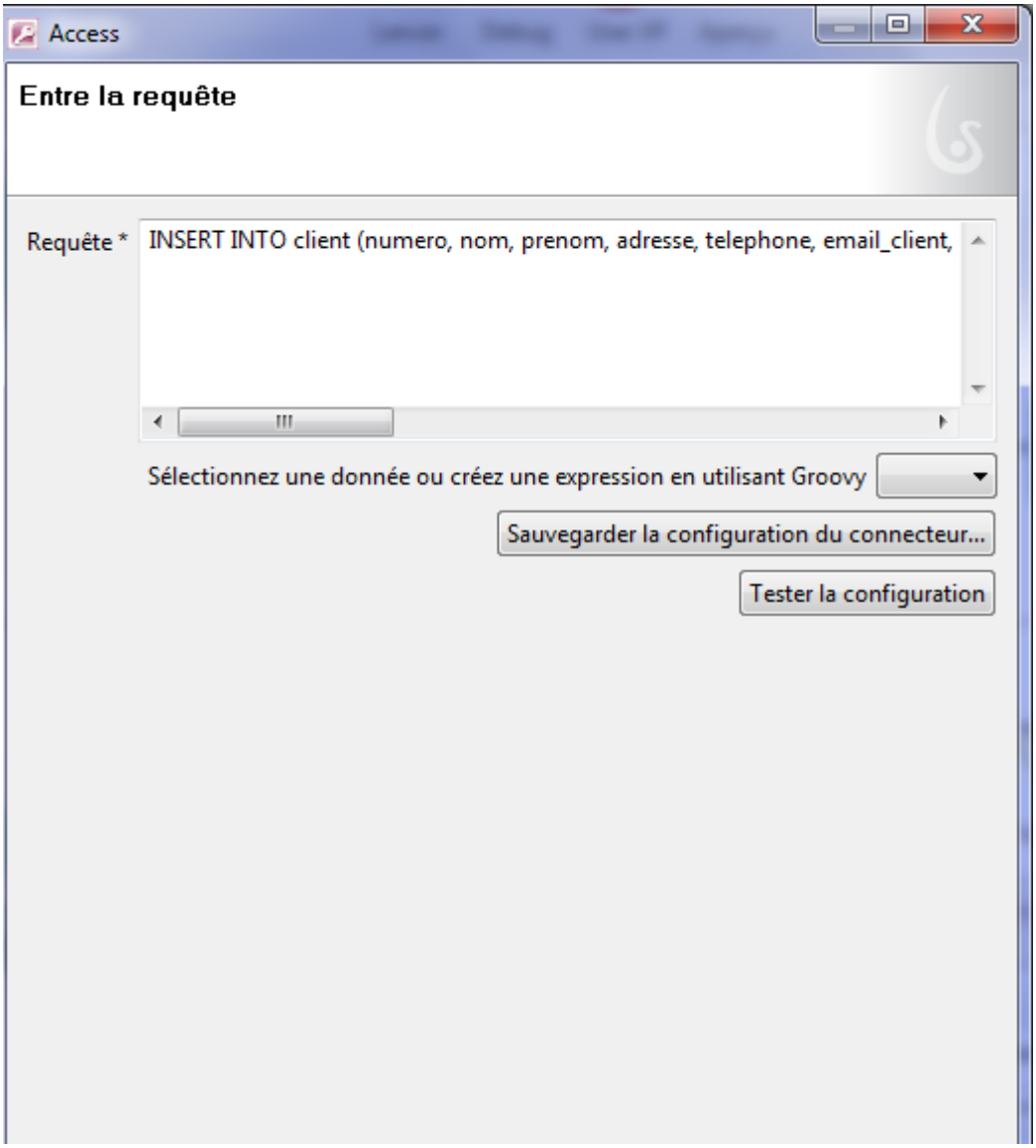


Figure27: Requête dans une base de données en Acces

5 .b. Solution :

La requête correcte : « INSERT INTO client (nom, prénom, adresse, téléphone, email client, email contact, crédit, type, thème)

VALUES ('${nom}', '${prenom}', '${adresse}', '${telephone}', '${email_client}', '${email_contact}', '${carte_de_credit}', '${type}', '${theme}') ».

On doit :

- Remplir tous les champs
- Respecter l'ordre des champs

- Faire un validateur de champ téléphone et de l'adresse émail.
- Contrôler les champs (***Obligatoire*** pour pousser l'utilisateur à remplir le champ, Accès en lecture seule s'il s'agit d'une récupération de données)

6 . a. Problème6 :

Taille de pièces jointes.

6 .b. Solution :

La taille de pièces jointes ne doit pas dépasser la taille standard ***25 Mo.***

7 . a. Problème7 :

Choix de navigateur : certains navigateurs posent problèmes comme ***Mozilla Firefox.***

7 .b. Solution :

Utilisation du navigateur ***Google Chrome.*** Car il récupère tous les champs et ne devise pas la page en deux. Et est plus rapide que tous les autres navigateurs.

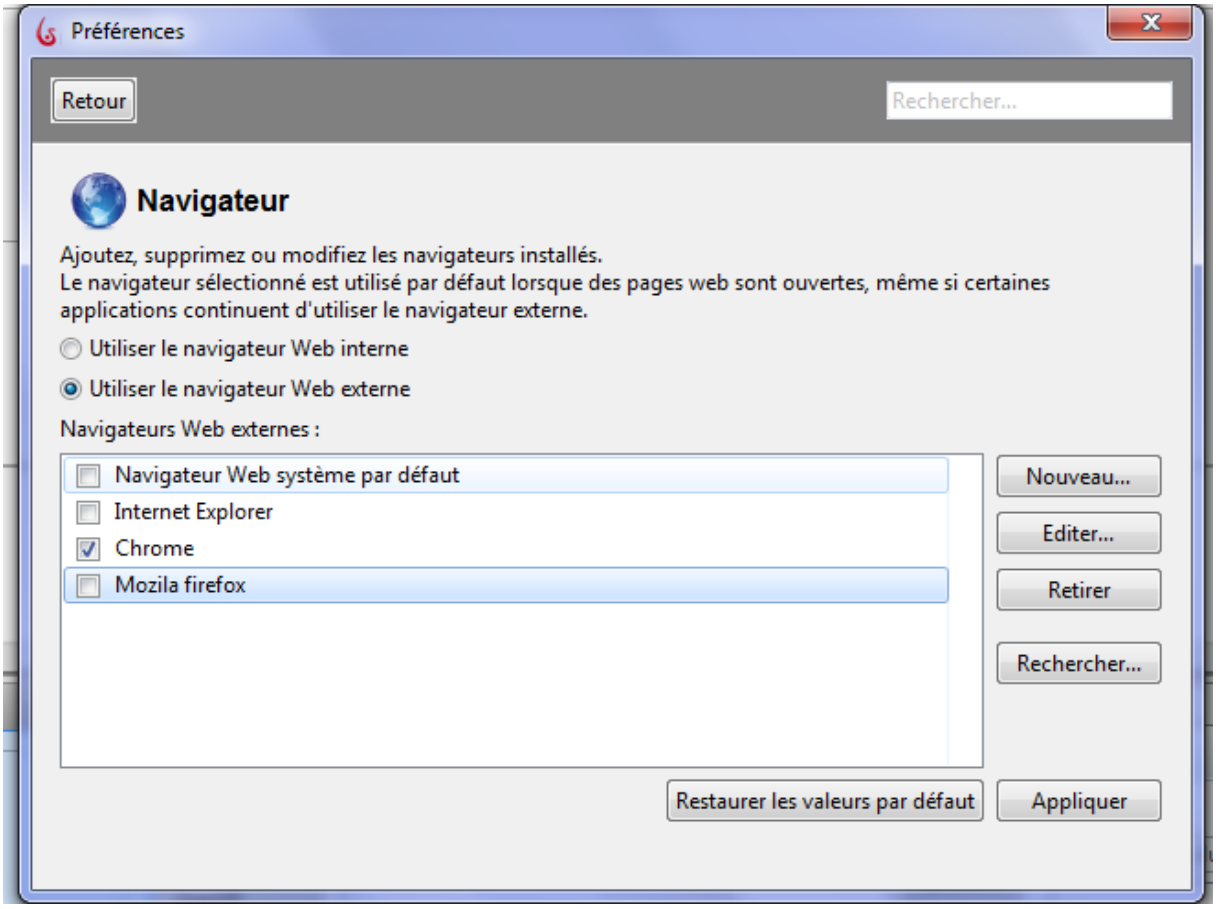


Figure28 : Choix de navigateur

8. a. Problème8 :

Problème de messagerie : L'anti virus sur ordinateur désactive certaine fonction.

8 .b. Solution :

Il faut désactiver l'anti virus.

9. a. Problème9 :

Nécessité de spécification de type de taches (humaine, service).

9 .b. Solution :

Tache humaine lorsque l'acteur est connu, Service pour une tache automatique.

IV.4 .2. Problèmes non résolus :

L'utilisation d'un outil Open source n'offre pas toutes les fonctionnalités, il reste toujours restreint par rapport à un outil payant.

Le problème majeur est le changement de versions, une version nouvelle corrige quelques erreurs dans la version précédente mais peut entrainer d'autres erreurs dans la nouvelle version en contrepartie.

- On a trouvé par expérience que les versions de ***Bonita 5.9*** et ***5.10*** n'offrent pas certaine fonctionnalités mais qui sont offertes par une version plus ancienne qui est ***5.7.***
- Ce qu'on a pu faire c'est de dupliquer le travail chaque fois qu'on se trouve dans l'obligation de changer de version pour ne rien perdre**,** on clique sur Processus→Dupliquer.

Exemple :

- Le problème d'authentification des acteurs.

L'échange dynamique de données est un problème dans la version ***5.10*** mais qui ne pose aucun problème dans la version ***5.7.***

Par contre la version ***5.7*** ne gère pas la fonction de cycle si on cherche a boucler contrairement a la version ***5.10*** qui gère les boucles

- BonitaOpenSolution est programmé en Java, Cependant chaque version de Bonita demande l'installation d'une version différente de Java sur ordinateur.

  Bonita5.7 nécessite l'installation de la version ***Java6***

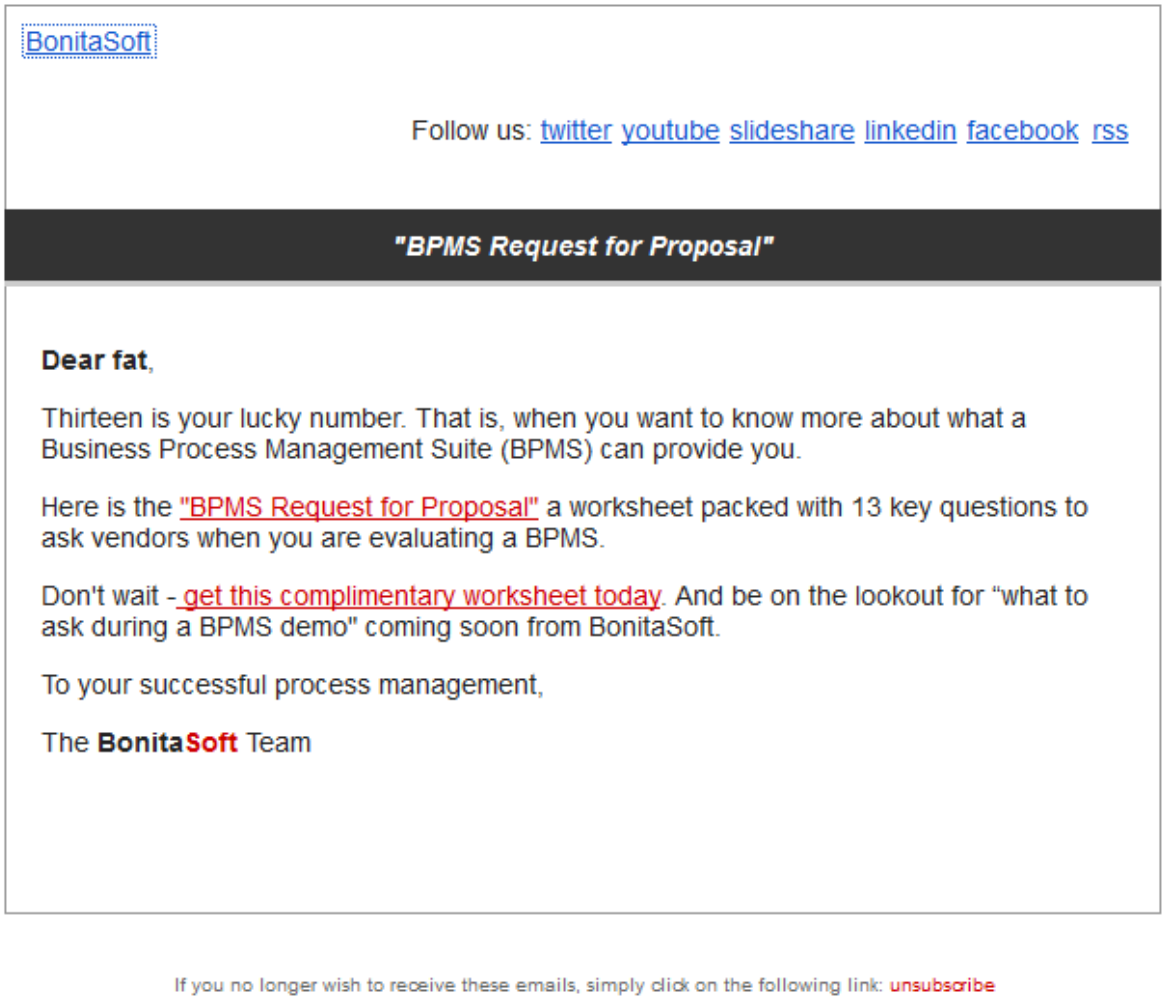
  Bonita5.10 nécessite l'installation de la version ***Java7***

- Le temps d'apprentissage important

  Exemple :
  Certains types d'affichage, ou insertion d'image dans les forms.
- Impossibilité de gérer le type de tache (Normale, Urgente).
- Notre outil ne gère pas plusieurs clients à la fois vue la complexité de l'outil open source Bonita.
- La version Bonita 5.7 n'a pas de notion de cycle, elle ne supporte pas le parallélisme. La résolution de ce problème nécessite le changement de version.

IV.4 .3. Quelques contraintes d'utilisations pour un bon fonctionnement :

- User expérience offre une boite de réception pour gérer les processus et les cas. On peut lancer une exécution depuis user XP ou accéder à l'exécution via les formes qui s'ouvre dans le navigateur.
- Pour utiliser au mieux cet environnement on doit d'abord contribuer dans le site officiel de Bonita pour être membre afin de recevoir toutes les newsletters, et être à jours des différentes évolutions de Bonitasoft solution.


BonitaSoft

Follow us: twitter youtube slideshare linkedin facebook rss

"BPMS Request for Proposal"

Dear fat,

Thirteen is your lucky number. That is, when you want to know more about what a Business Process Management Suite (BPMS) can provide you.

Here is the "BPMS Request for Proposal" a worksheet packed with 13 key questions to ask vendors when you are evaluating a BPMS.

Don't wait - get this complimentary worksheet today. And be on the lookout for "what to ask during a BPMS demo" coming soon from BonitaSoft.

To your successful process management,

The **BonitaSoft** Team

If you no longer wish to receive these emails, simply click on the following link: unsubscribe

Figure 29: Contribution au site officiel de Bonita

- Un sous Processus doit avoir un nouveau POOL.
- Pour faire le lien entre les données de différents Pools en utilise le ***Mapping automatique*** dans la mesure où les variables déclarées dans POOL ne peuvent être reconnues dans un autre.

IV.5. Conclusion

La réalisation de cet outil nous a permis de bien connaitre les principes de la notion d'agilité ainsi que celle du workflow.

A travers ce chapitre, nous avons essayé de porter l'attention sur l'utilisation de l'outil BonitaOpenSolution, et sur l'importance de l'idée de développer un tel outil d'aide a la conception agile qui est a vrai dire un sujet encore en cours de recherches afin d'aboutir a des solutions en tenant compte des besoins futurs et des natures de projets.

Conclusion Générale

C'est dans les champs particuliers de la conception et les méthodes de développement des systèmes d'information et des projets informatiques, vus à la fois comme domaine de connaissances propre et comme domaine d'application pour les outils et les méthodes informatiques, que se positionne notre réflexion.

Au cour de ce mémoire, nous avons présenté un outil qui a pour but d'aider une équipe d'informatisation à suivre une démarche agile pour la réalisation des applications dans un cadre collaboratif.

Pour aboutir au mieux à l'objectif fixé nous avons commencé par une étude conceptuelle basée sur le formalisme ***UML***, puis la concrétisation de l'outil sous un environnement Open source ***BonitaOpenSolution*** vu l'indisponibilité d'un logiciel payant.

La réalisation de ce travail est l'aboutissement d'un effort considérable ce qui a fait l'objet d'une expérience intéressante, qui nous a permis de nous familiariser plus avec la technologie ***Workflow*** et les notions d'agilité dans les méthodes de développement (méthodes agiles). Il nous a permis d'exploiter et d'améliorer nos connaissances et nos compétences dans le domaine de développement informatique et des méthodes ainsi de perfectionner notre méthodologie de recherche. Sa réalisation a été une bonne occasion pour cerner des connaissances acquises au long de notre parcours universitaire.

Ce que nous avons pu réaliser ne constitue en réalité qu'un début pour une série de travaux futurs afin d'aboutir a un outil d'aide a la conception agile rigoureux au vrai sens du mot. De plus le développement est souvent le royaume ou règne la loi du « vite fait-mal fait » surtout quand il s'agit d'un sujet de recherche pour cela vu le temps on a insisté plutôt sur une partie importante du travail demandé, chose qui nous laisse la porte ouverte pour l'amélioration de

cet outil qui reste un objectif à atteindre, cela en terme de son développement avec un outil pouvant offrir toutes les fonctionnalités chose qui nous manquait vu l'utilisation d'un outil open source.

Enfin tester l'outil dans une vraie équipe de développement afin de pouvoir évaluer l'impact des méthodes agiles sur le domaine de développement.

Bibliographie

[PR1] : **Pascal ROQUES,** «Le cahier des programmeurs, UML2 », 4ème édition, ISBN : 978-2-212-12389-0.

[AC5] : **Agnès CREPET**, « Modélisation agile & UML » ,3 décembre 2011.

[VM4]: Véronique Messager Rota Préface de Jean Tabaka Gestion de projet Vers les méthodes agiles

[SM6] : **SEMKAOUI Amal ; AKAOUCH Mohamed**, « Mémoire de Licence professionnelle » ; 2007/2008

Webographie

[EQ3]: http://www.agiliste.fr/items/agile-cest-quoi/

[UP2**]:** http://sabricole.developpez.com/uml/tutoriel/unifiedProcess/

Blogs

http://blog.valtech.fr/

http://xp.thierrycros.net/

http://scrum.aubryconseil.com/

http://www.dotnetguru2.org/proques/