RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINSTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ AMMAR TELIDJI - LAGHOUAT
Faculté des Sciences
Département de Mathématique et Informatique

MÉMOIRE DE MASTER

Domaine : Mathématique et Informatique

Filière : Informatique

Option : Système d'Information & Décision

Thème

Conception et Implémentation d'un Outil d'aide À la Collecte Des Besoins Par Maquettage

---

Présenté par :

BESSEKHOUAD YOUCEF FARTAS BACHIR

Soutenu devant le jury composé de :

Mme A.BELABASSI	Président	U. Amar TELIDJI, Laghouat
Mr. L.BENSAAD	Examinateur	U. Amar TELIDJI, Laghouat
Mr. Y.GUELLOUMA	Examinateur	U. Amar TELIDJI, Laghouat
Mme. B.KERROUCHE	Encadreur	U. Amar TELIDJI, Laghouat

Année / ...../2012-2013

Dédicace

Avec un énorme plaisir, un cœur ouvert et une immense joie, que je dédie mon travail à mes très chers, respectueux et magnifiques parents qui m'ont soutenus tout au long de ma vie ainsi à mes frères, ma sœur et sans oublier mes fidèles amis. À toute personnes qui m'ont encouragé ou aidé au long de mon parcours universitaire.

Youcef BESSEKHOUAD

Dédicace

Je dédie ce modeste travail à mes très chers parents qui m'ont toujours apporté leur soutien, leur dévouement, leurs encouragements, et mon appris à ne pas baisser les bras dans la vie. Que dieu vous bénisse et vous garde.

Mes dédicaces vont également à mon cher frère et ma chère sœur, et à toute ma famille.

À tous mes chers amis Mohammed.S, Mohammed.D, Noureddine.E, Youcef.B, Yacine.B, Mohammed.B, Elhadj.B, Tefik.S ; Pour tous les instants inoubliables que j'ai passés, et que je vais les passés avec vous.

À mon binôme Youcef BESSEKHOUAD que je remercie pour tous ses efforts.

À tous ceux qui m'aiment et ceux que j'aime,

À toi . . .

Bachir FARTAS

Remerciement

En préambule à ce mémoire, Nous tenons tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce modeste travail. En second lieu, nous tenons à remercier notre encadreur Mme KERROUCHE Badra, pour l'orientation, la confiance, la patience qui ont constitué un apport considérable sans lequel ce travail n'aurait pas pu être mené au bon port. Qu'il trouve dans ce travail un hommage vivant à sa haute personnalité. Nous tenons à exprimer nos sincères remerciements à tous les professeurs qui nous ont enseigné et qui par leurs compétences nous ont soutenu dans la poursuite de nos études. Nous tenons aussi à exprimer nos reconnaissances envers Mr Belhadj TIRICHINE qui a eu la gentillesse de lire et corriger ce travail. Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail Et de l'enrichir par leurs propositions. Enfin, 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 modeste travail. Merci à tous et à toutes.

Resumé

Notre travail est inclus dans la partie ingénierie des besoins (IB) qui est la partie la plus important dans le développement de n'importe quel système informatique. Ce travail consiste à proposer une démarche de collecte des besoins fondée sur le concept but-scénario où le scénario est illustré par des maquettes et sur le principe de la méthode O* (orienté objet et évènement). Cette démarche permet d'enrichir le créneau de recherche en IB et ainsi offrit à l'ingénieur des besoins et à l'utilisateur des outils qui leur permettant d'assurer une bonne collecte des besoins et par conséquent un système de qualité.

Mots clé : Système d'information, Ingénierie des besoins, Fragment de besoin, But, Scénario, Maquette.

Abstract

Our work is included in the part of requirements engineering (RE) which is the most important part in the development of any information system. this work consist to propose a process of gathering requirements based on the concept goal-scenario where the scenario is illustrated by mock-up and the principle of the method O* (object-oriented and event). This approach permit to enrich the search space in requirement engineering and offer to the engineer of requirements and the user tools that it permits him to ensure a good requirement collecting and consequently a good quality of system

Keywords : Information system, Requirement ingenierie, Requirement chunck, Goal, Scenario, Mock-up.

Table des matières

Table des figures

Liste des tableaux

Liste des Abréviations

IB : Ingénierie des besoins

FB : Fragment de besoin

MERISE : Méthode d'étude et de Réalisation Informatique pour les Systèmes d'Entreprise.

SCRAM : Scenario Requirements Analysis Method

CREWS : Cooperative Requirements Engineering With Scenarios

GAB : Guichets Automatiques Bancaires

KAOS : Knowledge Acquisition in autOmated Specification

UML : Unified Modeling Language

MOF : Meta Object Facility

CASE : Computer Aided Softw are Engineering

OMG : Objet Management Group

Introduction

Au années 1960, L'évolution des méthodes de développement à commencer par un simple codage de chaque tâche toute seule, après quelques années les chercheurs en développement des systémes d'informations ont ajouté un concept qui s'appelle la conception afin de regrouper et organiser les données.

Puis, en 1976 Bell et Tayer ont fait observer que l'inadéquation des fonctionnalités du système aux besoins des usagers, l'incohérence, l'incomplétude et l'ambiguïté des documents de besoins avaient une influence majeure sur la qualité du logiciel final [1], Ce qui a conduit à l'apparition de ce qu'on appelle l'ingénierie des besoins (IB) ,qui est un processus d'identification de buts assignés au système envisagé et de les transformer en contraintes et exigences imposées au système et aux agents qui assurent le fonctionnement de ce système. Elle peut être aussi vue comme un processus qui permet de transformer une idée floue en spécification précise des besoins servant de support à la spécification du système et de ses interfaces avec l'environnement. Au cours de l'IB, les besoins attachés au système sont découverts, négociés, validés, spécifiés et répertoriés dans des documents de spécification des besoins afin d'être utilisé par les concepteurs du système.

Enquêtes et chiffres

Plusieurs enquêtes ont confirmé la gravité des dégâts causés par l'insuffisance de qualité des documents et des exigences, parmi ces enquêtes :

1. Une enquête menée auprès de 800 projets conduits dans 350 compagnies américaines par le Standish Group (2003) et présentée dans deux rapports, intitulés « Chaos »et « Unfinished Voyages », a prouvé que 31% des projets sont annulés avant même d'être terminés [1]. En 1995, cela a coûté 81 milliards de dollars aux compagnies américaines. Ce même rapport montre que 50% d'entre eux n'avaient que partiel-

lement réussi dans le sens où ils avaient n'ecessité des budgets et des délais très fortement majorés. La mauvaise qualité des documents des exigences constitue 47% des causes d'échecs citées. Ce pourcentage est distribué de la façon suivante :

- manque de participation des utilisateurs (13%).
- besoins mal exprimés ou incomplets (12%).
- besoins changés entre le début et la fin du projet (11%).
- besoins qui manquent de réalisme (6%).
- objectifs peu clairs (5%).

2. Une enquête de grande envergure effectuée en Europe auprès de 3800 organisations dans 17 pays différents a conclu que : les principaux problèmes sont liés à la spécification des exigences (>50% des réponses) et à la documentation des exigences (50%) [1]. Christel & kang (1992) ont classé les problèmes qui se posent lors de léxtraction des exigences en dix points, répertoriés en trois classes différentes [1] :

   (a) Problèmes d'étendue du système étudié.
   - Les frontières du système sont mal définies.
   - Des informations non nécessaires sont fournies.

   (b) Problèmes de compréhension
   - Les utilisateurs ont une idée incomplète de leurs besoins.
   - Les utilisateurs connaissent mal les possibilités et contraintes des systèmes proposés.
   - Les analystes ont une faible connaissance du domaine.
   - Lútilisateur et lánalyste parlent des langages différents.
   - Il est facile dómettre des informations.
   - Il peut exister des conflits de points de vue entre différents utilisateurs.
   - Les besoins sont souvent vagues et non mesurables.

   (c) Problèmes de volatilité des exigences
   - Les exigences évoluent au cours du temps.

à travers ces enquêtes en peut déduire límportance de l'IB et son rôle majeur dans la production de systèmes de qualité qui répondent aux besoins de l'utilisateur.

Problématique

Existe-il des méthodes qui permettent à língénieur des besoins de pénétrer au sein du désordre des besoins de lútilisateur et de collecter le nécessaire ?

Objectif

Notre objectif est d'introduire une méthode qui conduit vers une bonne collecte des besoins, c'est à dire :

1. Faciliter et fiabiliser le travail de l'ingénieur et de l'utilisateur.
2. Avoir des besoins fiables dans les meilleurs délais.

Organisation du mèmoire

Notre document est organisé comme suit :

Introduction

Ce chapitre sert à expliciter l'importance de l'IB et son rôle majeur dans le monde des systèmes.

Chapitre 1 : Etat de l'art

Ce chapitre permet de récapituler et de donner une idée sur les travaux réalisés sur le sujet de la collecte des besoins.

Chapitre 2 : Conception

Ce chapitre permet de lister toutes les étapes de conception de l'outil de collecte des besoins.

Chapitre 3 : Implémentation

Ce chapitre permet de lister tous les outils utilisés pour l'implémentation et de donner une idée sur l'outil de la collecte des besoins (structure) et la façon d'utiliser cet outil (démarche à suivre).

Conclusion

Ce chapitre permet de récapituler les quatre chapitres précédents et de spécifier les perspectives qui peuvent faire l'objet dúne autre recherche.

Chapitre 1

État de l'art

1.1 Introduction

Sous l'effet de pousser la roue de développent vers l'avant, un état de l'art est incontournable dans le but d'avoir une idée sur les travaux connexes afin de localiser leurs inconvénients et lacunes pour les diminuer ou bien les illuminer définitivement si c'est possible, sans oublier leurs avantages et atouts pour les soutenir et les conserver.

1.2 Les approches traditionnelles

Dans les méthodes héritées des années 80 telles que MERISE, l'IB est une partie intégrante de l'étape d'analyse aboutissant à la construction d'une représentation abstraite des données, des traitements et des interfaces en suivant une approche de modélisation conceptuelle. Leur objectif principal est de décrire ce que le système doit faire, c'est à dire ses fonctionnalités dans un schéma conceptuel [2].

Dans ces approches traditionnelles, l'IB fondée sur la modélisation conceptuelle est centrée sur la question QUOI à laquelle on répond par le cycle <acquisition, abstraction, validation> qui est schématisé dans figure 1.1 [2].

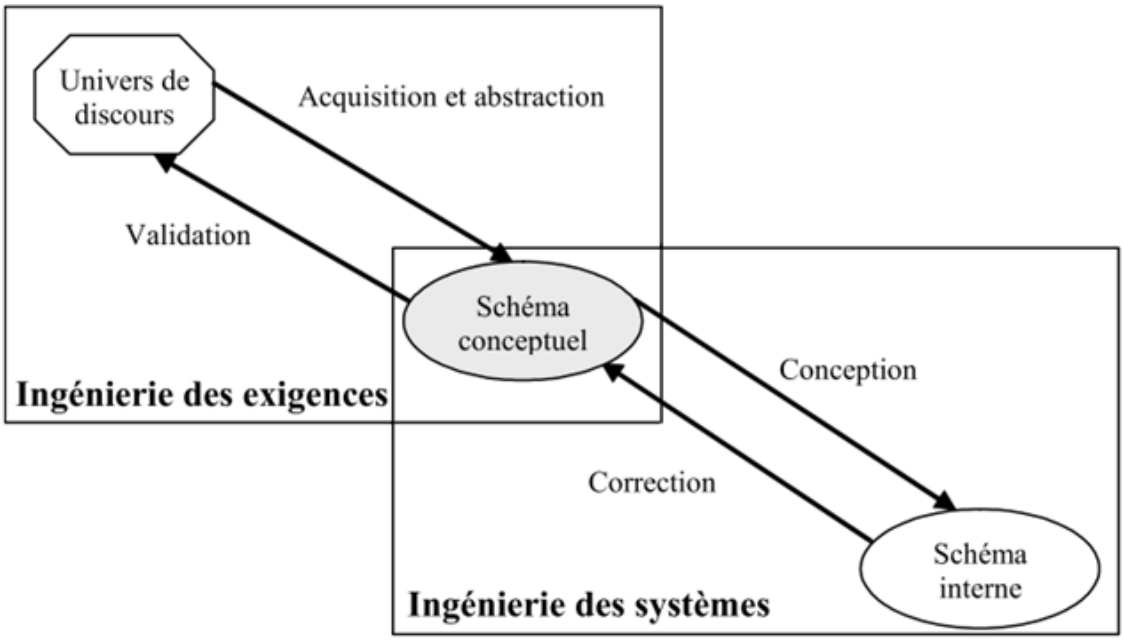

FIGURE 1.1 – Le cycle <acquisition, abstraction, validation> [2]

Explication du schèma :

La tâche d'acquisition de connaissance de domaine et des besoins s'appuie sur un cahier des charges et des interviews. Elle permet l'abstraction de la connaissance acquise et la spécification des fonctionnalités attendues du système, Le schéma conceptuel résultant représente un support de validation des besoins par des techniques telles que le maquettage et le prototypage.

Un centrage exclusif sur le QUOI inhibe les vraies questions du POURQUOI et aboutisse à la production de la spécification conceptuelle d'un système inadapté aux réelles attentes de ses usagers. Par ailleurs, le schéma conceptuel est l'expression d'une solution fonctionnelle et non des besoins à l'égard du système. L'absence d'une expression des besoins eux-mêmes ne peut que rendre difficile la validation par les parties concernées. En outre la pratique du cycle précédent repose sur des hypothèses qui ne semblent plus valides aujourd'hui [2], et qui sont les suivantes :

1. Les fonctionnalités d'un système sont stables, elles n'évoluent que peu dans le temps.
2. Les besoins relatifs à un système sont donnés au départ.
3. La validation des besoins peut se faire en référence aux fonctionnalités du système c'est-à-dire à l'aide du schéma conceptuel.

Dans les années 1980/90, ces hypothèses étaient valides mais elles ne le sont plus au-

jourd'hui. En réponse aux pressions économiques et à l'émergence constante de nouvelles technologies, les organisations changent plus rapidement que par le passé. En conséquence, ce que les utilisateurs attendent du système évolue bien plus rapidement qu'auparavant. Leurs besoins ne sont donc pas stables. Il est donc nécessaire de se doter de moyens permettant d'établir un lien conceptuel entre les buts et les besoins qui en résultent (réponse au POURQUOI), et les spécifications fonctionnelles du système qui les supportent (réponse au QUOI) [2].

1.3 Ingénierie des besoins

La plus ancienne définition de l'IB est donnée par les deux chercheurs « Ross »et « Schoman »qui la définit comme suite : « *requirements definition is a careful assessment of the needs that a system is to fulfil. It must say why a system is needed, based on current and foreseen conditions, which may be internal operations or an external forces. It must say what system features will serve and satisfy this context. And it must say how the system is to be constructed* »[2].

En d'autres termes, le principal intérêt de l'IB est de déduire les buts du contexte organisationnel du système à développer ce que permet de comprendre les raisons qui justifient son développement. Alors l'IB doit s'intéresser sur le pourquoi et poser la question suivante : **pourquoi le système doit être développé ? à quelles fins ? Et pour quel objectif ?** Et aider les parties prenantes du projet à y répondre. Afin de permettre de découvrir les besoins correspondants à la mission et aux objectifs de l'organisation, si l'on veut éviter de développer des systèmes techniquement parfaits mais inutilisés parce qu'inadaptés aux besoins réels de leurs utilisateurs.

Puis l'IB va s'occuper du quoi et de poser la question suivante : **Qu'est-ce que le système doit faire ? Et quelle est sa fonctionnalité ?** Afin de déterminer les fonctionnalités que le système doit mettre en œuvre pour la satisfaction de ces buts et d'identifier les contraintes qui restreignent la mise en œuvre de ces fonctions.

Ces buts, fonctions et contraintes constituent les besoins qui doivent ultérieurement être convertis en une spécification précise permettant le développement du système.

L'évolution des besoins au cours du temps et la répercussion de leur évolution sur la spécification du système doit aussi être prise en compte.

La Figure 1.2 empruntée à Axel van Lamsweerde [2] schématise cette vue de l'IB centrée sur les questions POURQUOI (WHY) et QUOI (WHAT) ainsi que la mise en correspondance des réponses inhérentes à l'une et à l'autre.

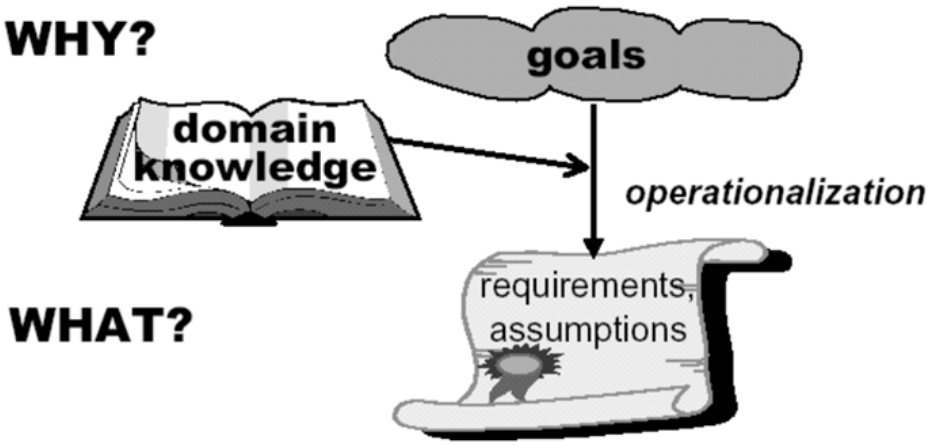

FIGURE 1.2 – Le fondement de l'ingénierie des besoins [2]

1.3.1 Le processus d'IB

Selon 'Nuseibeh' et 'Easterbrook' (2000), le processus d'IB inclut les six phases suivantes : Elicitation, Modélisation, Analyse, Spécification, Validation et Gestion des exigences [1].

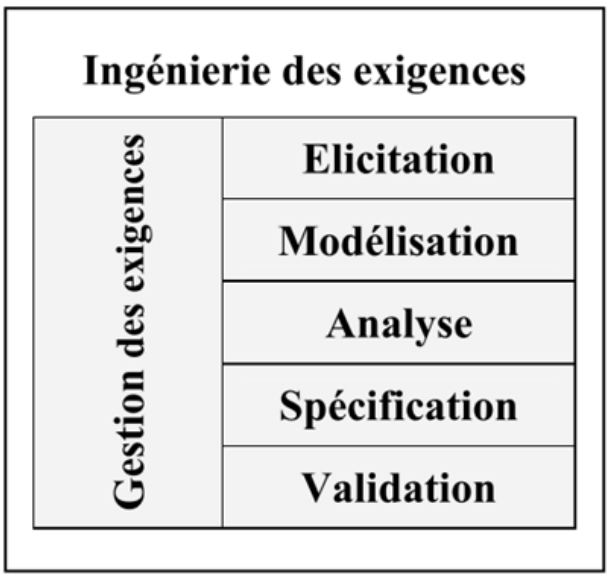

FIGURE 1.3 – Les phases de l'ingénierie des exigences [1]

Explication :

- **Elicitation :** consiste à collecter, capturer, découvrir et développer les exigences à partir d'une variété de sources y compris les parties prenantes humaines.
- **Modélisation :** « c'est la construction d'une description et d'une représentation abstraite d'un système pour les besoins de l'interprétation »selon 'Nuseibeh' et 'Easterbrook'. Divers aspects d'un système qui peuvent être modélisés pour avoir une compréhension approfondie du contexte ou de l'environnement (exemple : la modélisation de données, modélisation comportementale).
- **Analyse :** Cette phase se focalise sur l'examen, la compréhension des exigences élicitées et de vérifier leur qualité en termes d'exactitude, de complétude, de clarté et de consistance.
- **Spécification :** est l'enregistrement et la documentation des exigences dans le but d'être utilisées par les parties prenantes, et en particulier, par les développeurs. Elle consiste à établir la liste finale des exigences en les organisant suivant des catégories. Les exigences selon différentes perspectives doivent être intégrées dans le but d'avoir une vision commune du système.
- **Validation :** est la confirmation de la qualité des exigences et de leur conformité aux besoins et désirs des parties prenantes. Selon la norme EIA-632 [1], la validation est focalisée sur la vérification de la version finale du document des exigences pour détecter les conflits, les omissions et les déviations.
- **Gestion des exigences :** Cette phase est exécutée tout au long du processus d'IB. Elle consiste à suivre l'évolution et le changement des exigences, à faire la traçabilité et le contrôle des différentes versions de ces exigences durant tout le processus d'IB.

1.3.2 Comment aboutir à des besoins fiables ?

Afin d'avoir des besoins fiables, il est nécessaire de prendre en compte les quatre principes suivants [3] :

1. **Bien définir la cible des utilisateurs :** identifier les différentes populations utilisatrice en fonction de leur profil et de leurs attentes. et pour chacune de ces populations Identifier leurs représentants.

1. norme EIA-632 : c'est une norme qui complète les processus techniques de définition du système en couvrant la réalisation des produits jusqu'à leur mise en service

Remarque : il est toujours préférable de privilégier un contact direct avec le client pour comprendre les motivations (Les administrateurs métiers des applications existantes ne doivent pas être les seuls interlocuteurs).

2. **Varier les modes de collectes :** définir et organiser les modes de collecte d'information les plus adaptés au contexte du projet, des moyens et des délais :
   - **Recherche interne :** rechercher les informations sur les besoins et attentes des utilisateurs à partir des données disponibles (activité des systèmes existants, base de réclamations, ...).
   - **Interview :** identifier et collecter les besoins généraux ou spécifiques par profil.
   - **Groupe de travail :** composer des groupes de travail représentant un département ou une fonction pour creuser des besoins identifiés en interviews.
   - **Questionnaires :** mesurer les attentes des utilisateurs ou l'importance et la performance d'une fonction par une série de questions distribuée sous un support (exemple : papier).

   Remarque : quel que soit le mode de collecte, il est toujours intéressant de croiser les questions sur un mode positif et négatif (exemple : Quelle est votre perception sur cette fonction ? - Quelle est votre perception si cette fonction n'est pas couverte ?)

3. **Catégoriser les besoins :** catégoriser les besoins selon le taux de satisfaction attendu par rapport au niveau de couverture (modèle Kano) [4].

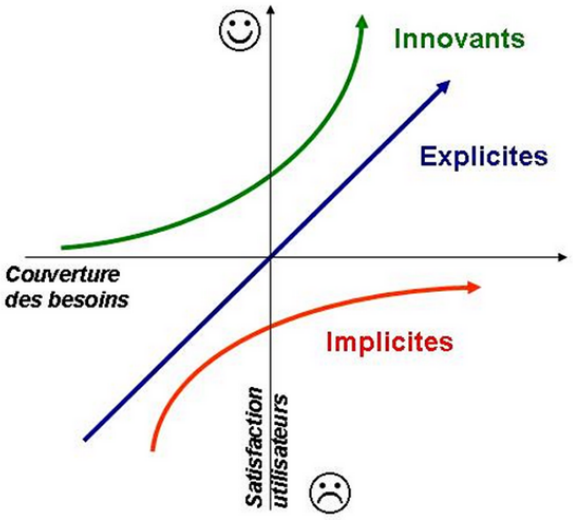

FIGURE 1.4 – Courbe du modèle Kano [4]

Explication de la figure :

- **Les besoins implicites :** ce sont des besoins qui sont tellement évidents pour l'utilisateur qu'il ne les exprime pas et leur absence amène une grande insatisfaction. Il s'agit de détecter le bon niveau de service afin de satisfaire l'utilisateur, sans faire de sur-qualité.
- **Les besoins explicites :** ce sont des besoins ouvertement exprimés par l'utilisateur. La satisfaction ou l'insatisfaction de l'utilisateur croît de façon linéaire avec la performance. Il s'agit d'améliorer ces performances continuellement afin de maximiser la satisfaction.
- **Les besoins latents ou innovants :** ce sont des besoins non exprimés par l'utilisateur car il n'en a pas conscience ou n'imagine même pas que ce besoin puisse être couvert. Lorsque ces besoins sont remplis, l'utilisateur est enthousiaste et l'effet de levier peut fortement augmenter la courbe de satisfaction.

4. **Prioriser les besoins :** les contraintes de budget et de délais des projets nécessitent de prioriser les besoins collectés. Cette priorisation s'inscrit dans la démarche suivante :
   - **Synthèse des besoins :** regroupement des besoins par types puis une analyse par segment de population est effectuée.
   - **Convergence :** identification des fonctions nécessaires pour répondre aux attentes.
   - **Pondération :** évaluation des fonctions proposées par rapport aux bénéfices business attendus et à la satisfaction des utilisateurs.
   - **Finalisation :** élaboration de scénarios, priorisation des fonctions et validation du système cible.

   Remarque : il faut se concentrer sur l'essentiel en laissant la possibilité au projet de prendre en compte des besoins innovants ou pas en fonction des priorités.

1.3.3 Empêchements et obstacles de l'IB

Parmi les obstacles qui empêchent le déroulement du processus d'IB les suivants [2] :

1. Le champ est large, c'est à dire l'objectif à atteindre n'est pas seulement un logiciel mais il comprend aussi l'environnement organisationnel dans lequel il doit opérer. Celui-ci est complexe car il intègre les êtres humains, les machines et les

logiciels, alors le système doit être considéré selon de multiples points de vue : socio-économiques, physiques, opérationnels, évolutifs. . .etc.

2. Il faut prendre en compte les aspects non fonctionnels qui sont des aspects liés à la sécurité, la performance, la robustesse, la facilité d'utilisation, l'interopérabilité, etc. du système qui ont la particularité d'être souvent en conflit les uns avec les autres.
3. L'existence de nombreuses parties prenantes. Selon les ingénieurs des besoins les parties prenantes d'un système sont : les clients, les experts du domaine, les utilisateurs, les commanditaires, les ingénieurs de maintenance et les ingénieurs d'application où chacune d'elle a son champ de compétence et de connaissance, sa propre culture, ses propres points de vue et intérêts. ce qu'il conduit à des conflits.
4. Les spécifications des besoins peuvent être entachées aux nombreuses erreurs. Certaines d'entre elles peuvent être terriblement pénalisantes et avoir des effets désastreux sur les étapes suivantes du développement et/ou sur la qualité du produit final.
5. Le processus de l'IB couvre de multiples activités qui sont inter-reliées.

1.4 Les différentes sources des besoins

Les exigences d'un système étant découvertes à partir de son environnement, qui se compose de trois parties : le monde du sujet, le monde de l'usage et monde du système [1].

- **Le monde de l'usage :** il décrit la façon dont le système est utilisé ainsi que l'ensemble des activités, tâches, procédures, interactions etc. accomplies par les agents afin d'atteindre les objectifs de l'organisation. C'est à dire identifier les intentions, souhaits et buts des utilisateurs du futur système.
- **Le monde du sujet :** il contient les connaissances du domaine pour lequel le futur système doit apporter de l'information et génère des exigences qui reflètent les lois du domaine. Ce monde décrit les objets du monde réel qui doivent être représentés dans le système.
- **Le monde du système :** Il permet de déterminer les spécifications du système et dans lequel les besoins issus des deux autres mondes doivent être formulés et documentés. Ce monde détient aussi les représentations des entités, événements,

processus des mondes du sujet et de l'usage ainsi que leur transformation en spécifications techniques et implémentations logicielles.

Tous ces mondes sont en interaction comme le montre la Figure 1.5. Les besoins issus du monde de l'usage sont capturés par la « relation intentionnelle »et la « relation d'adéquation à l'usage ». Les besoins imposés par les connaissances du domaine sont capturés par la « relation d'adéquation au domaine »[1].

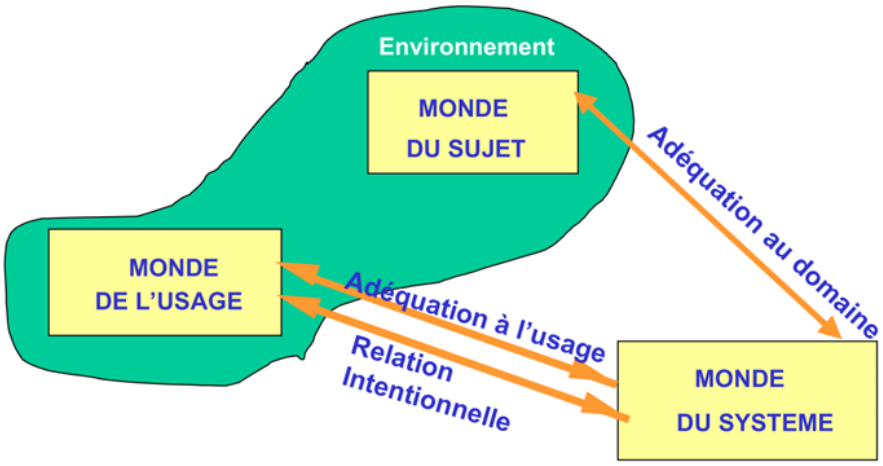

FIGURE 1.5 – Les interaction entre les trois mondes (usage, sujet, système) [1]

Dans ce qui suit on va illustrer les trois approches de l'IB qui se basent sur les deux notions (but et scénario) et pour chacune on va associer un exemple d'une méthode utilisant cette approche, ces approches sont :

1. Les approches dirigées par les buts (exemple : i*).
2. Les approches dirigées par les scénarios (exemple : SCRAM, ScenIC).
3. Les approches dirigées par le couple buts-scénarios (exemple : CREWS).

1.5 Approches dirigées par buts

1.5.1 Que veut dire un but ?

Un but est définit comme « un objectif à réaliser en utilisant le futur système »[5]. C'est à dire un objectif que le futur système doit atteindre par la coopération des agents avec le système.

1.5.2 Principe de l'approche dirigée par buts

Les approches conventionnelles centrées sur le QUOI formalisent les contraintes imposées aux opérations et aux objets du système mais ne disent pas POUQUOI ces contraintes sont imposées et si elles sont suffisantes pour garantir que les objectifs que l'organisation place dans le système seront satisfaits ou non. « Yue »a été le premier à argumenter en faveur l'introduction de modèles de buts dans le document de spécification des exigences afin d'introduire un critère de complétude des exigences où une spécification des exigences est complète si elle permet de satisfaire le but qu'elle affine [1].

Les buts sont formulés à différents niveaux d'abstraction, allant de buts stratégiques de haut niveau (tels que 'Fournir un service de retrait d'argent' pour un réseau de Guichets Automatiques Bancaires (GAB)) à des buts techniques de bas niveau (tels que 'Absorber la carte de crédit après trois tentatives de saisie de code incorrectes' pour le GAB). Les modèles de buts se basent sur la décomposition d'un but en sous buts dans des graphes de réduction ET/OU inspirés de l'intelligence artificielle qui dirige le processus d'IB [2].

Une réduction "ET" associe un but B à un ensemble de sous buts B1,B2,...,Bn qui doivent tous être satisfaits pour que B le soit.

Une réduction "OU" associe un but B à un ensemble B1,B2,...,Bn de sous buts tels que la satisfaction de l'un d'entre eux assure la satisfaction de B.

Ce mécanisme de réduction est fondamental pour assurer le passage du POURQUOI (les buts stratégiques de haut niveau) au QUOI (les buts de bas niveau qui s'opérationnalisent en objets et en opérations du SI).

Les graphes ET/OU précédents comportent des buts que l'on dit 'durs' (hard goals) et qui font référence à des fonctions du système, ils sont donc fonctionnels par opposition aux buts 'mous' (soft goals) qui font référence à des qualités attendues des fonctions. Ces buts mous sont les buts non fonctionnels qui sont utilisés pour raisonner sur les besoins qualitatifs imposés au système [2].

Dans une relation de satisfaction, un sous but contribue positivement ou négativement à la satisfaction du but père dans une certaine limite plutôt que de manière absolue.

1.5.3 Avantages des approches dirigées par buts

Plusieurs approches dirigée par buts telle que (KAOS, i*) ont été utilisées dans plusieurs projets et voici ce qu'elles ont apporté comme avantages [2] :

- Elles permettent de vérifier et d'assurer la complétude de la spécification des besoins. Celle-ci est complète si l'on montre que l'ensemble des buts peut être satisfait par l'ensemble des besoins de la spécification.
- Elles facilitent l'explication des exigences aux parties prenantes : les buts notamment de haut niveau fournissent les raisons de développement du système. Un besoin n'existe que s'il y a un but qui justifie sa présence dans la spécification.
- Une bonne exploration des alternatives de conception : le système à développer peut fonctionner et interagir avec son environnement de multiples façons. Le mécanisme de réduction des buts et le graphe ET/OU qui en résulte aide les ingénieurs d'IB à expliciter de nombreuses alternatives et à raisonner pour déterminer celle qui semble la mieux appropriée à la situation du projet.
- Structurer la collecte des besoins : en soutenir du mécanisme de réduction des buts.
- Etablir les liens de traçabilité : le graphe de réduction des buts est un moyen d'établir les liens qualifiés de pré-traçabilité, ce qui assurent la relation entre les objectifs organisationnels et les besoins à l'égard du système sont utiles pour propager un changement des premiers dans les seconds.
- Identifier et gérer les conflits : les différents acteurs du processus d'IB apportent des points de vue intéressants sur le système à développer ; on sait que ces points de vue peuvent être divergents et générer des conflits. Il semble que les buts aident l'explicitation des conflits et à leur résolution.

1.5.4 Inconvénients des approches dirigées par buts

Comme tout travail humain, ces approches ont quelques inconvénients non négligeables qui sont :

- Il n'existe pas de définition unifiée du concept de but [5].
- Les approches existantes n'indiquent pas :
  - Comment un but doit être formulé et ce qu'il doit exprimer [5].
  - Si un but est adapté aux besoins ou non [5].
- Le risque d'avoir un changement de buts.

1.5.5 Exemple d'une approche dirigée par buts

La méthode i* pour l'élicitation des exigences

La méthode i* fournit une description de l'organisation du travail en terme de relations de dépendances entre acteurs, elle considère que les acteurs ont la liberté d'action dans les contraintes sociales (inter acteurs) qui sont appelées dépendances stratégiques [1].

i* est fondée sur les sept concepts suivant [1] :

Acteur : la notion d'acteur modélise une entité ayant des buts stratégiques et des intentions. Un acteur représente un agent physique (une personne, un animal, une voiture) ou un agent logiciel, il peut avoir un rôle ou une position. Un rôle est une caractérisation abstraite du comportement d'un acteur dans un certain contexte, alors qu'une position représente un ensemble de rôles, typiquement joué par un seul acteur.

But : un but représente un intérêt stratégique d'un acteur. Ces buts peuvent être fonctionnels ou non fonctionnels.

Croyance : les croyances sont utilisées pour représenter les connaissances de chaque acteur sur le monde (l'environnement).

Plan : un plan représente une manière ou un ensemble d'actions à réaliser pour satisfaire un but.

Ressource : une ressource représente une entité physique ou informationnelle recherchée par un acteur et fournie par un autre acteur.

Capacité : la capacité représente l'aptitude d'un acteur à définir, choisir et exécuter un plan pour réaliser un but, dans un environnement donné.

Dépendance : une dépendance entre deux acteurs indique qu'un acteur dépend d'un autre acteur pour atteindre un but, exécuter un plan ou délivrer une ressource.

Processus de i*

L'élicitation des exigences dans i* est un processus composé de deux étapes [1] :

- La première étape consiste à éliciter des exigences dites préliminaires en utilisant un diagramme de dépendance stratégique.
- La deuxième étape a pour objectif l'élicitation des exigences finales en utilisant un diagramme de raisonnement.

1. Les exigences préliminaires :

   Dans i*, les intentions sont modélisées en buts, une analyse orientée but est introduite pour déterminer les exigences fonctionnelles et non-fonctionnelles du futur système. Les exigences préliminaires sont supposées impliquer des acteurs sociaux qui dépendent les uns des autres pour atteindre des buts, et accomplir des tâches. i* utilise le diagramme de dépendance stratégique pour décrire le réseau des relations de dépendances sociales parmi les acteurs, de même les diagrammes de raisonnement pour l'analyse et la réalisation des buts.

2. Les exigences finales :

   Durant l'analyse des exigences finales, le système futur est décrit dans son environnement d'exécution à travers les fonctions et les qualités pertinentes. Le système est représenté par un ou plusieurs acteurs ayant des dépendances avec d'autres acteurs. Ces dépendances définissent toutes les exigences fonctionnelles et non-fonctionnelles du futur système.

1.6 Approches dirigées par scénarios

Les acteurs de l'organisation ont parfois des difficultés à énoncer à priori les objectifs attendus du système ; ceux-ci ne sont facilement explicités qu'après avoir bien compris ce que doit faire le système. Même si l'objectif basé sur le raisonnement est tout à fait approprié pour l'ingénierie des exigences, les objectifs sont parfois difficiles à obtenir. La difficulté à manipuler des notions abstraites comme les buts a été reconnue par des études en sciences cognitives [2].

1.6.1 Définition du scénario

Un scénario est une description partielle ou complète des interactions entre un usager et le système pour accomplir une tâche spécifique. Jacobson (1992) a inventé le terme de cas d'utilisation pour les scénarios et plus tard, il l'introduit dans UML. Par conséquent, ce terme est largement utilisé aujourd'hui. Les scénarios ont souvent décrit une information par un exemple, par contre ce qu'est voulu est de la généraliser sous forme d'un modèle pour l'utiliser dans l'ingénierie des systèmes. Les scénarios ont été recommandés comme un moyen d'améliorer et de valider les exigences et ils ont été préconisés comme un moyen

efficace de la communication entre les développeurs et les parties prenantes [6].

Les scénarios ont différents contenus. Ils peuvent décrire [2] :

- des activités, actions ou événements actuels ou futurs,
- des objets mis en jeu par le système,
- le cadre de travail actuel ou imaginé par les acteurs,
- des contraintes de qualité attendues du système,
- des informations sur l'organisation tels que sa structure, ses départements, ses groupes, ses agents,...etc,
- les intervenants dans le processus de l'IB, leurs caractéristiques, leurs vues et leurs aspirations.

Un scénario permet d'appréhender des réalités plus faciles à décrire que les buts. Les scenarios sont de trois types : descriptifs, exploratoires et explicatifs. Ils permettent de connaître les acteurs du système et les différents éléments qui le composent tels que :

- les évènements, les processus,...etc.
- le pourquoi du monde réel.
- les fonctionnalités désirées pour le système à développer.

L'approche par scénario met en valeur le lien établi entre les exigences et le système désiré, projeté dans l'avenir, ce qui contribue à identifier les besoins, mais le problème c'est le nombre des scénarios nécessaires pour dire qu'on a capturé suffisamment d'exigences.Les scénarios sont capturés comme des récits de texte, des croquis et des médias informels. Les scénarios sont exprimés à trois niveaux d'abstraction différents [2] :

1. **Niveau instance :** au niveau instance, un scénario utilise des noms réels (le client « Dupont ») ou des descriptions d'événements avec des valeurs de paramètres réels (arrivée de la commande 812, le 20/09/99 à 17h33).
2. **Niveau type :** les scénarios exprimés au niveau type n'utilisent pas d'entités du monde réel mais des types d'entités. Ils ne font pas référence à Dupont mais à la notion de client. L'exécution d'un scénario exprimé au niveau type est un scénario du niveau instance.
3. **Mixte :** un scénario mixte comporte des parties au niveau type et d'autres au niveau instance.

Les scénarios sont exprimés dans différentes notations [2] :

- **Informelles :** les scénarios informels sont décrits à l'aide du langage naturel, de vidéos, etc. Ils sont utilisés lorsque les utilisateurs rejettent les notations formelles ou semi formelles.
- **Semi-formelles :** les scénarios semi-formels utilisent des notations structurées comme des tableaux ou des scripts.
- **Formelles :** les scénarios formels sont décrits à l'aide de langages basés sur des grammaires régulières ou des diagrammes d'états. Ils peuvent être utilisés pour simuler le fonctionnement futur du système et juger des réactions des utilisateurs.

1.6.2 Démarche

Un processus typique pour l'IB basé sur les scénarios est composé de cinq étapes :

- acquisition des scénarios,
- génération de la spécification,
- vérification de la spécification,
- génération d'un prototype,
- et enfin validation du prototype avec les utilisateurs.

En premier lieu, l'analyste commence par acquérir les scénarios des utilisateurs. Puis, une spécification qui décrit le comportement dynamique du système est synthétisée à partir des scénarios. Ensuite, l'analyste vérifie la spécification dans le but de détecter et corriger des scénarios incohérents et/ou incomplets. La phase suivante consiste en la génération d'un prototype du système à partir de la spécification. Finalement, la cinquième étape a pour rôle de valider le prototype avec les utilisateurs. En cas de scénarios invalides détectés lors de la troisième étape ou de la dernière étape, l'analyste retourne à la première étape et recommence les différentes étapes jusqu'à ce que tous les scénarios soient valides [7].

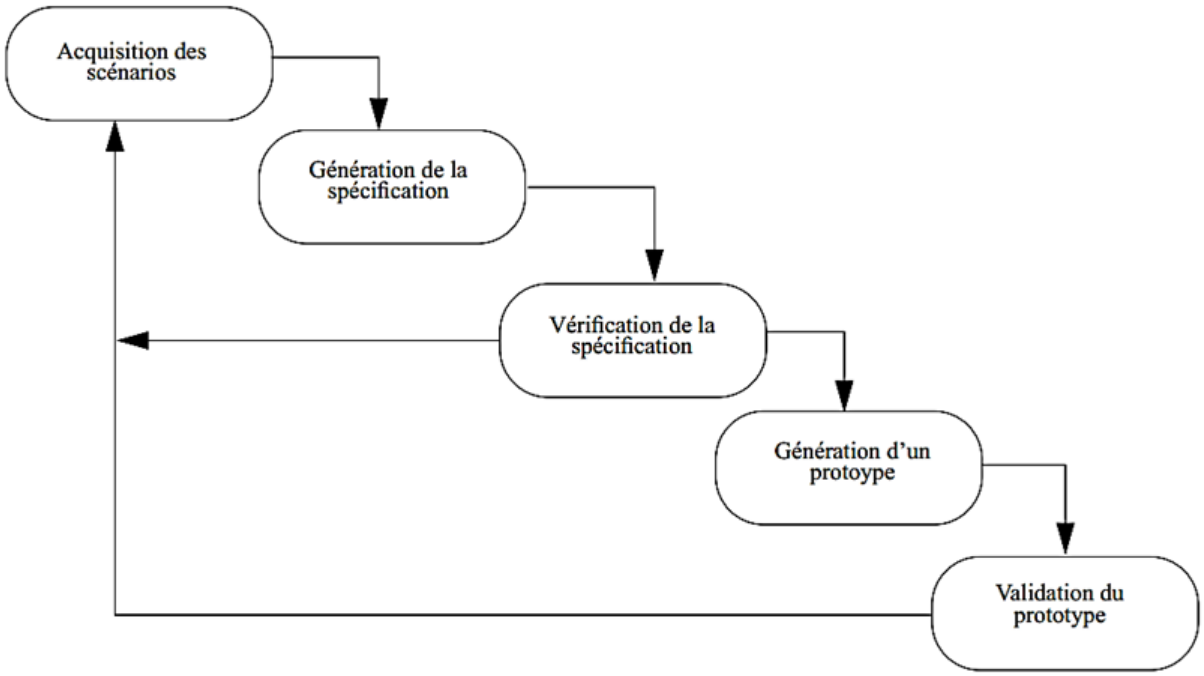

FIGURE 1.6 – Processus d'ingénierie des besoins basé sur les scénarios [7]

1.6.3 L'utilisation des scénarios

Plusieurs interprétations différentes sont apparues allant de scénarios comme base pour générer des cas d'utilisation, les descriptions de système pour aider à comprendre les implications du système.

Les scénarios ont été utilisés avec des notions différentes. Une distinction importante est compris entre les scénarios qui font partie de la spécification du système requis, par rapport à des scénarios qui sont capturées à partir de l'expérience réelle (par des exemples réels) et ne proviennent pas d'une conception du système. Le premier point de vue de scénario a été adopté pour le cas d'utilisation dans la littérature orientée objet, dans lequel un cas d'utilisation exprime une vision projetée d'interaction de l'utilisateur avec le système conçu [8]. Deux méthodes importance on IB baser sur scénarios, la méthode de ScenIC et SCRAM.

1.6.4 Exemple

ScenIC

Peu de méthodes ont présenté la façon d'utiliser des scénarios dans le processus de l'analyse des besoins et de validation. L'une des exceptions est the Inquiry Cycle de Potts qui utilise le scénario scripts pour identifier les obstacles ou les problèmes dans l'analyse des besoins orienté par but. Cette méthode propose un schéma (voir figure 1.7) des connaissances liées au scénario composée de buts, objectifs, tâches, obstacles et acteurs. Les scénarios sont composés d'épisodes et d'action menés par les acteurs, qui sont généralement des humains, mais peuvent être aussi des machines.

Les buts sont classés soit en état de réalisation, état de maintenance ou à éviter, tandis que les obstacles empêchent les objectifs d'être atteint, ou inhibent la réussite des tâches. Le procédé de la méthode se déroule dans un cycle d'exprimer des scénarios dans un format semi-structuré, en critiquant et en inspectant les scénarios dans walkthroughs (procédure pas à pas) qui conduit au raffinage des exigences [9].

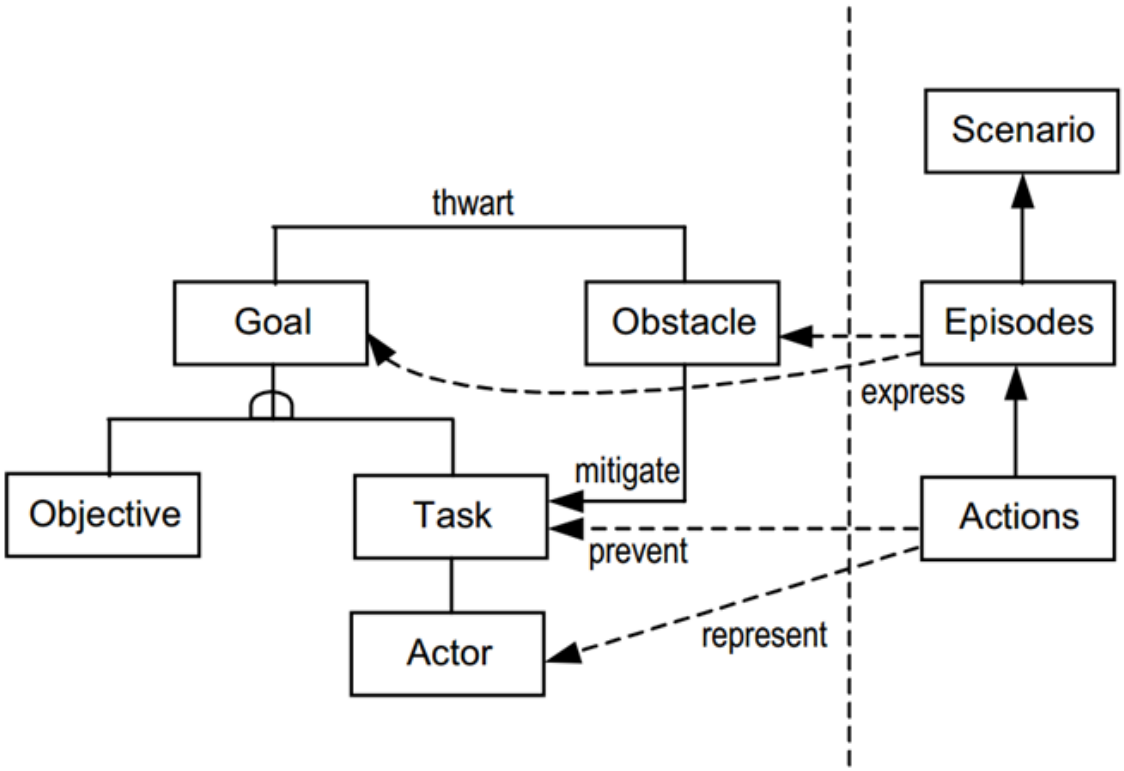

FIGURE 1.7 – Schéma des connaissances liées au scénario d'après Potts [9]

Les directives sont données pour le formatage des scénarios narratifs et l'identification

des buts, des actions et des obstacles. Les épisodes du Scénario sont évalués pour voir si les objectifs peuvent être atteints par les tâches du système, si les acteurs peuvent mener bien les tâches, s'il y a des obstacles qui empêchent les acteurs d'exécuté les tâches, etc. Avec cette manière de dépendances entre les buts, les tâches, les acteurs et les ressources on peut vérifier et s'assurer que le système réponde à ses exigences [9].

SCRAM

La seconde méthode d'analyse des besoins est la méthode SCRAM qui recommande une combinaison entre prototypes, scénarios, Logique de conception et des manifestes de validation.

En SCRAM, les scénarios sont utilisés avec les premiers prototypes afin de susciter les exigences en réaction à une conception préliminaire. L'approche est basée sur l'hypothèse que la technique d'intégration fournit le meilleur moyen pour améliorer l'IB et que la participation des utilisateurs à la conception est la meilleure façon d'obtenir un effet efficace pour la validation des besoins. Une autre motivation est d'utiliser des scénarios comme moyen de situer la discussion sur la conception, de sorte que des nouvelles exigences peuvent être obtenues par le raisonnement sur les problèmes posés par les scénarios décrivant un contexte d'utilisation [9].

Cette approche fusionne l'élicitation et la validation des scénarios en fournissant le contexte pour que l'utilisateur évalue la conception qu'il a lui-même présenté sous forme d'un scénario. Les étapes de SCRAM sont illustrées dans la figure 1.8.

Cette approche fusionne l'élicitation et la validation des scénarios en fournissant le contexte pour que l'utilisateur évalue la conception que lui-même la présenter par un scénario. Les étapes du procédé SCRAM sont illustrées dans la figure suivante [9] :

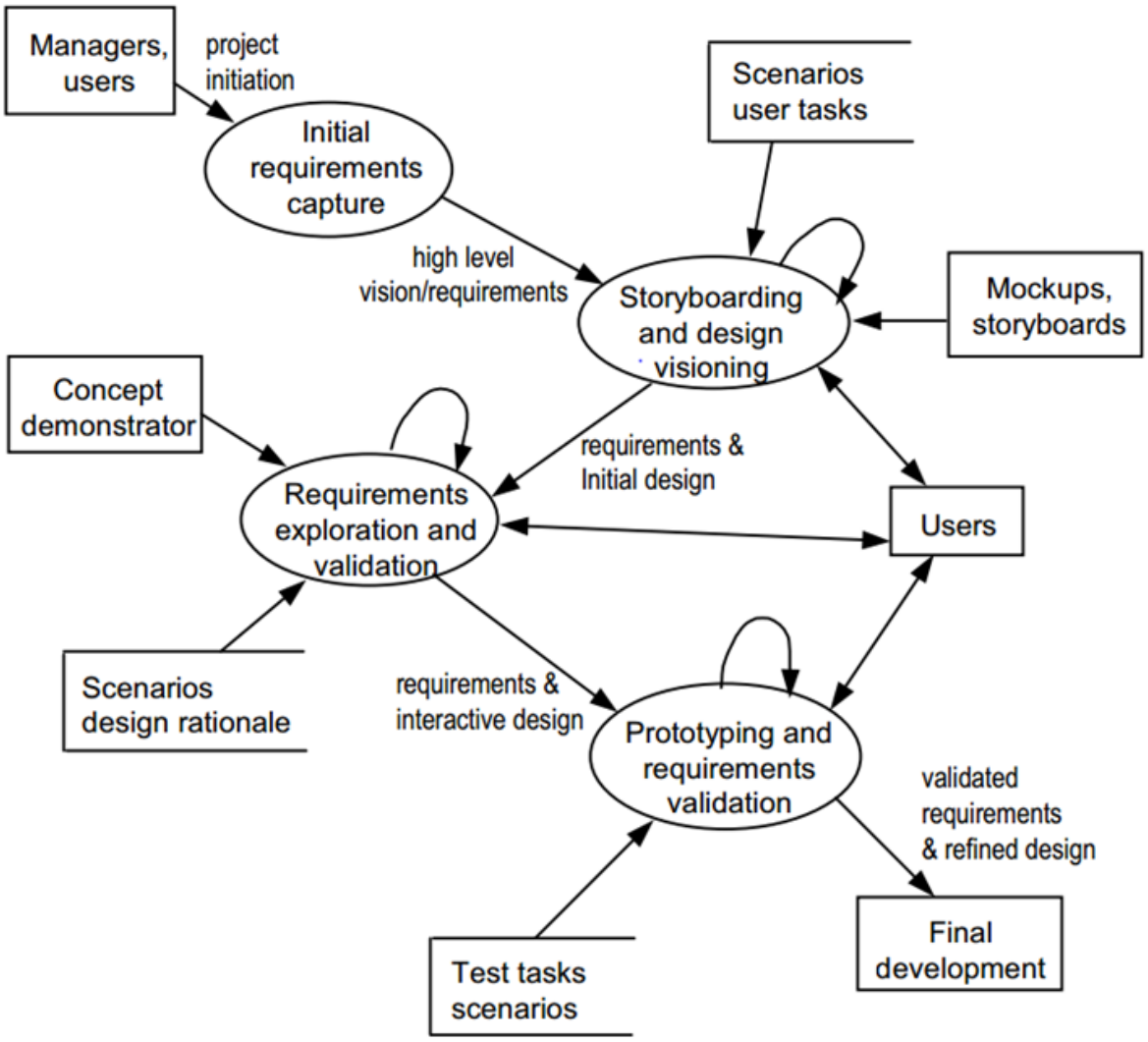

FIGURE 1.8 – Processus de la méthode SCRAM [9]

La méthode contient quatre phases :

1. **Initiale exigences capturé et le domaine familiarisation :** c'est réalisé par des entretiens classiques et des faits techniques pour gagner suffisamment de renseignements pour développer un démonstrateur du premier concept. En pratique ce prend 1-2 visites de clients.
2. **Storyboard et vision de la conception :** cette phase crée une première vision du système requis qui est expliqué aux utilisateurs en storyboard (un document présente le scénario d'un système par figure) pour obtenir des commentaires sur la faisabilité.
3. **Exigences d'exploration :** elle utilise les manifestes et les premiers prototypes pour présenter plus des détails de la conception aux utilisateurs de sorte que la

conception puisse être critiquée et que les exigences soient validées.

4. **Prototypage et validation des exigences :** cette phase développe plus de prototypes entièrement fonctionnels et continue d'affiner les exigences jusqu'à l'élaboration d'un prototype jugé acceptable par tous les utilisateurs [9].

1.6.5 Avantage

1. **Point de vue utilisateur :** un scénario considère toujours un système du point de vue de l'un de ses utilisateurs. Cet avantage est fondamental lors de la validation des besoins. Un Scénarios donne aux utilisateurs une sensation de participation réelle au développement du système.
2. **Spécifications partielles :** les Scénarios sont un moyen naturel pour la rédaction de spécifications partielles. Chaque scénario capture une séquence d'interactions utilisateur-système représentant une transaction ou un fonctionnement du système du point de vue d'un utilisateur.
3. **Facilité de compréhension :** les Scénarios simplifient à la fois l'élicitation et la validation des exigences, parce que la notion d'interaction d'utilisateur-système se révèle être une façon naturelle de comprendre et de discuter les exigences à la fois pour les utilisateurs et les ingénieurs.
4. **Cycles Courts de feedback :** la combinaison de la capacité de traiter chaque fonction utilisateur séparément dans un scénario et la façon dont l'utilisateur représente les besoins dans les scénarios, permettent des cycles courts de feedback entre les utilisateurs et les ingénieurs.
5. **Base de test du système :** l'interaction des séquences capturées dans les scénarios sont également une base idéale pour définir un test du système [6].

1.7 Approches dirigées par le couple buts-scénarios

Les buts sont difficiles à extraire à cause de l'absence d'une définition unifiée du concept but et l'approche orientée but ne donne pas une formulation explicite du concept but, les scénarios sont plus facile à utiliser que les buts. Cependant les scénarios utilisent des exemples et des illustrations concrètes, ce qui conduit à des descriptions des

besoins restreints et nécessite une généralisation afin d'obtenir des besoins complets [5]. Comme solution aux limites et aux difficultés de l'approche par but et à celle par scénario des approches basé le couplage de ces deux concepts ont été proposé comme l'approche CREWS-L'Ecritoire.

1.7.1 L'approche CREWS-L'Ecritoire

L'approche CREWS-L'Ecritoire est développée au sein du projet CREWS (Cooperative Requirements Engineering With Scenarios) en basant sur la notion du fragment de besoin.

Qu'est-ce un Fragment de besoin ?

Un fragment de besoin est un couple de deux notions but et scénario <B,Sc>,dont le quel le but est intentionnel et le scénario est opérationnel pour aboutir à une définition du FB qui est « une manière de réaliser un but au moyen d'un scénario ».

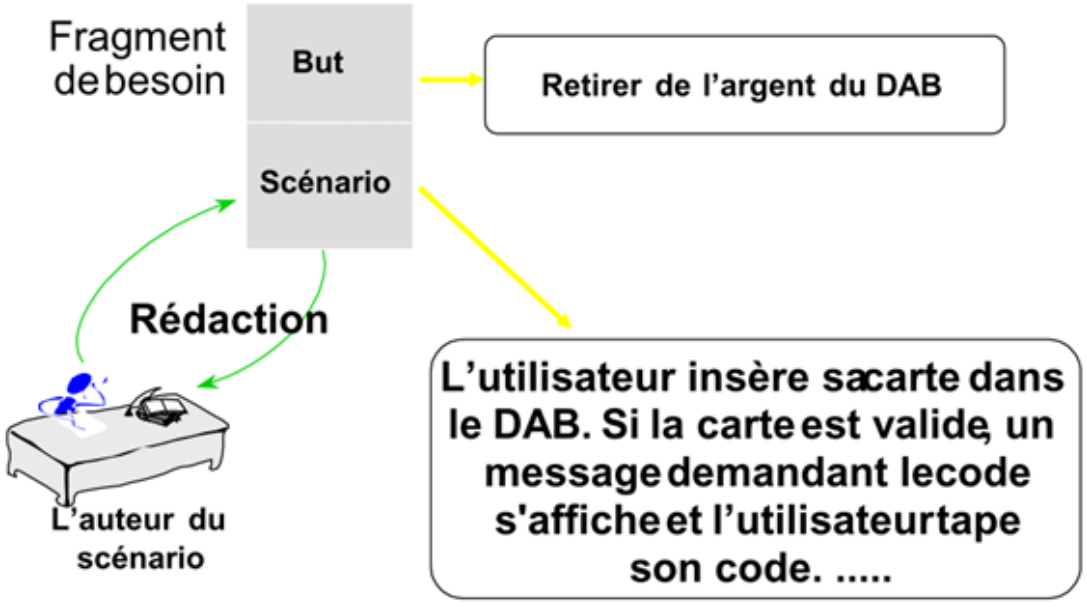

FIGURE 1.9 – La notion du fragment de besoin [5]

Dans la figure 1.9, en voit que le but est une expression littérale qui exprime ce que l'utilisateur veut obtenir à l'aide du système. Pour lever une partie du caractère flou d'un but, l'approche s'appuie sur une formalisation du but fondée sur une adaptation de l'approche linguistique [5].

Le concept de but

Le but est exprimé comme une clause avec un verbe principal et plusieurs paramètres, où chaque paramètre joue un rôle différent par rapport au verbe. Un exemple d'un objectif exprimé dans cette structure est la suivante : (Fournir) $_{verbe}$ (de l'électricité) $_{objet}$ (du producteur PPC) $_{source}$ (à nos clients) $_{bénéficiaire}$ (en utilisant le réseau PPC) $_{moyen}$ (d'une façon normale) $_{manière}$ [5]
Basant sur cette structure :

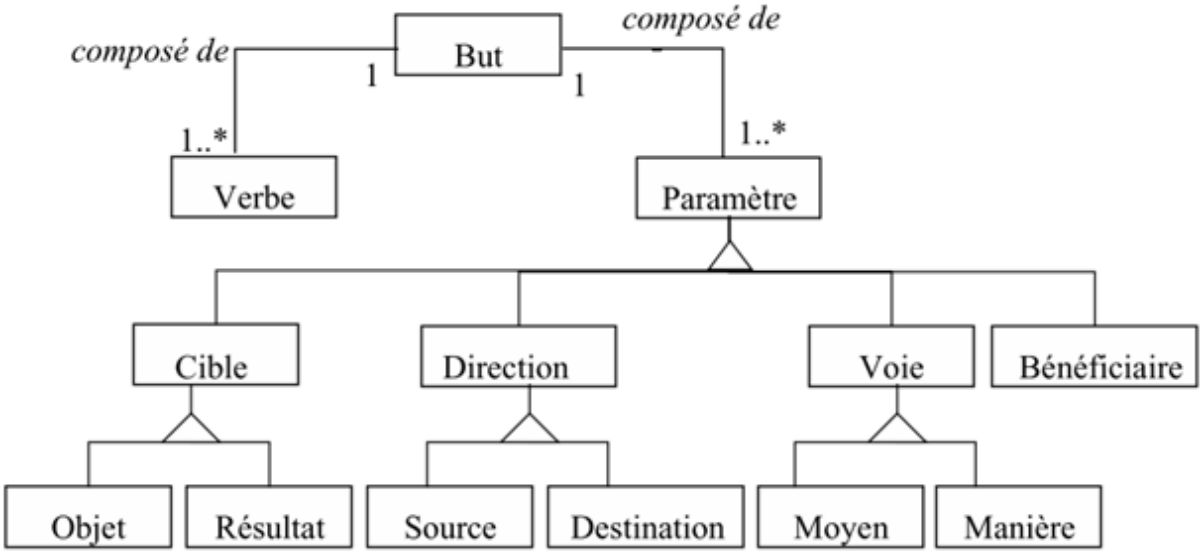

FIGURE 1.10 – la structure du but selon UML [5]

Le concept scénario

Il est composé d'une ou plusieurs actions, une action est une interaction d'un agent à un autre. La combinaison des actions dans un scénario décrit une trajectoire unique. Un scénario est caractérisé par un état initial et un état final. Un état initial fixé à un scénario définit une condition préalable pour que le scénario puisse être déclenché. Un état final définit un état atteint à la fin du scénario. Par exemple, le scénario « retirer de l'argent à partir du GAB dans le cas normal »ne peut se déclencher que si les deux états « l'utilisateur a une carte »et « le GAB est prêt »sont vrais. Il faut distinguer entre les scénarios normaux et exceptionnels. Le premier conduit à la réalisation de son objectif associé tandis que le second échoue dans la réalisation des objectifs. Par exemple, le scénario associé au but « retirer de l'argent à partir du GAB en saisissant trois fois un

code erroné »est du type exceptionnel avec les états finaux « l'utilisateur n'a pas une carte », « l'utilisateur n'a pas de l'argent »et « le GAB est prêt »[5].

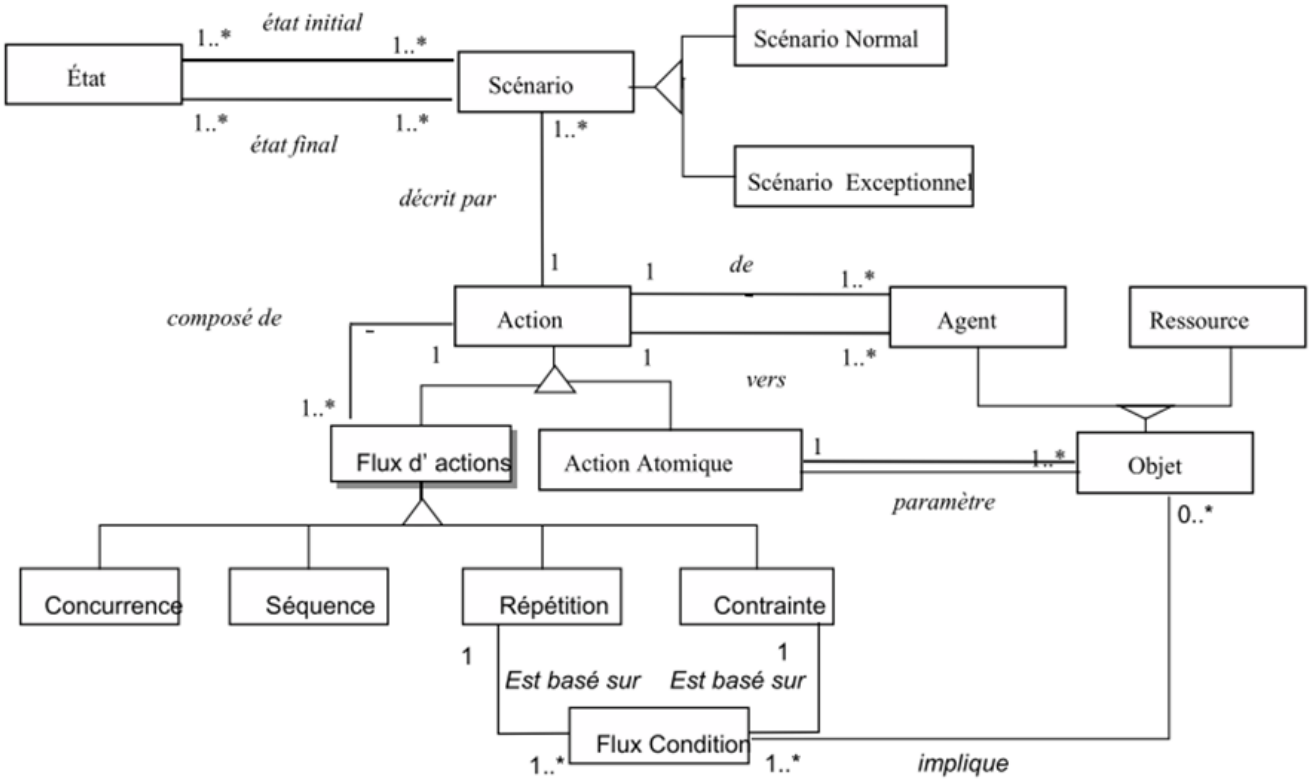

FIGURE 1.11 – La structure du scénario selon UML [5]

La hiérarchie des Fragments de besoins

Les Fragments de besoins sont reliés entre eux par trois relations nommés composition, alternatifs et affinement. La composition et l'alternatif présentent une structure horizontale par les liens « OU »et « ET »alors que la troisième relation présente une structure verticale par le lien appelé « AFFINE PAR ». Ces trois relations se retrouvent à différents niveaux d'abstraction : comportement, fonctionnel et physique [5].

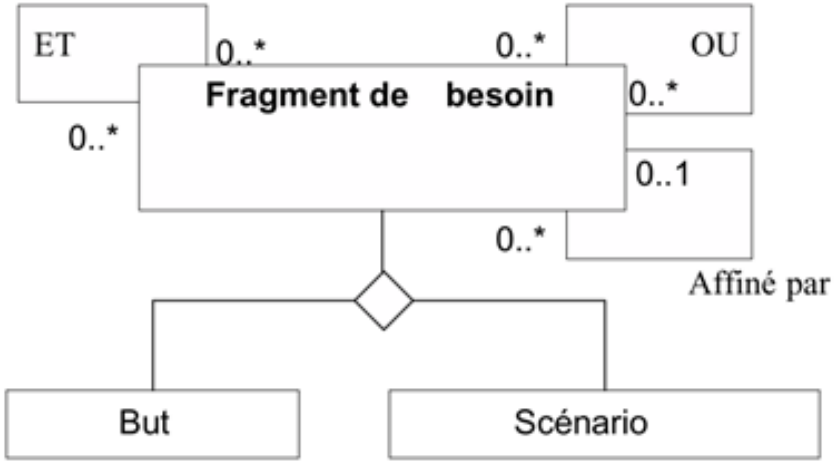

FIGURE 1.12 – La structure d'un fragment de besoin [5]

Le processus de l'approche CREWS-L'Ecritoire

Dans l'approche CREWS-L'Ecritoire, on introduit dans le processus de construction trois propriétés : l'itération, la systématique et la bidirectionnelle. Lorsqu'il y a un but à atteindre, un scénario est écrit pour définir la manière dans lequel on atteindre ce but. Le scénario une fois relié au but, est analysé pour la découverte de nouveaux buts (voir la figure 1.13).

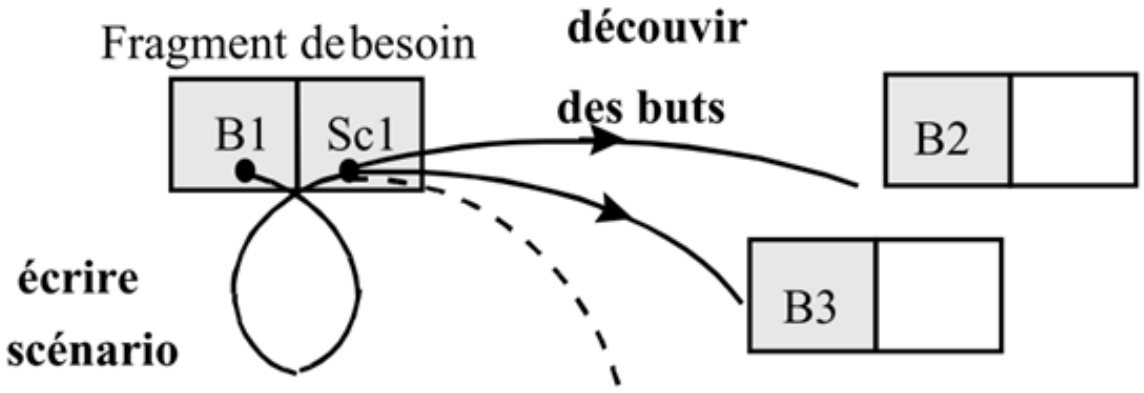

FIGURE 1.13 – Le processus de l'approche CREWS-L'Ecritoire [5]

Le processus est donc incrémentale et itératif : les fragments de besoin sont découverts au fur et à mesure des itérations du processus. Chaque itération comporte deux activités

principales qui se font d'une façon semi-automatique pour découvrir et documenter les besoins selon un mouvement bidirectionnel entre les deux concepts but et scénario :

1. Ecrire, vérifier et conceptualiser un scénario associé à un but.
2. Découvrir de nouveaux buts par analyse du scénario.

Le processus est conduit par une équipe qui comporte un ingénieur des besoins qui joue le rôle de facilitateur et un groupe des experts du domaine appelé Auteur de Fragment de Besoin (AFB).

L'écriture des scénarios Les scénarios sont écrits en langage naturel, l'écriture d'un scénario est une activité complexe où l'on peut distinguer trois sous-activités : (1) écrire le scénario, (2) vérifier et compléter les actions du scénario à l'aide de patrons sémantiques, et (3) conceptualiser le scénario [5].

1. ***Ecrire le scénario***

   Cette activité commence par l'écriture d'un texte en langage naturel décrivant une manière possible d'atteindre un but. Pour réaliser cette activité l'utilisateur est guidé par deux types de directives : Les directives du contenu et les directives du style.

   Les directives du style définie la formulation du texte du scénario, tandis que les directives du contenu se base sur le contenu du scénario. Par exemple, étant donné le but « retirer de l'argent à partir du GAB dans le cas normal », l'écriture du scénario permet d'obtenir la description suivante :

états initiaux : 'l'utilisateur a une carte' , 'le GAB est prêt '.
L'utilisateur insère une carte. Le GAB vérifie la validité de la carte, si elle est valide, un message demandant le code est affiché. L'utilisateur tape son code sur le clavier du GAB.Si celui-ci est valide alors, un message demandant le montant est affiché à l'utilisateur. L'utilisateur introduit le montant dans le GAB.Si celui-ci est disponible, alors le GAB éjecte la carte à l'utilisateur. Si un reçu est demandé. Le GAB imprime un reçu et il délivre les billets à l'utilisateur.
états finaux : 'l'utilisateur a une carte' , 'le GAB est prêt ', 'l'utilisateur a de l'argent'

2. ***Vérifier et compléter les actions du scénario à l'aide des patrons sémantiques***

   Les patrons sémantiques permettent d'identifier et de compléter les paramètres manquants dans une action. Un paramètre manquant est exprimé par un point d'interrogation. Prenons l'exemple de l'interaction suivante : « L'utilisateur insère une carte ».

   Le verbe 'insérer' est un verbe de communication et l'interaction est donc exprimée par le patron instancié suivant :

   Communication (insérer) [Agent : 'l'utilisateur' ; Objet : 'une carte' ; Source : 'l'utilisateur' ; Destination : ' ?']

   Le paramètre 'Destination' est inconnu et donc exprimé par un point d'interrogation ' ?'. Ceci permet de détecter l'absence du paramètre et de compléter l'interaction par l'ajout de celui-ci (le GAB). L'interaction est complétée ainsi on obtient la description suivante : « L'utilisateur insère une carte dans le GAB ». Ce mécanisme de vérification est automatique dans l'approche. La vérification des actions du scénario permet d'obtenir un texte où les modifications sont indiquées en gras.

3. ***Conceptualiser le scénario***

   La conceptualisation d'un scénario permet de transformer un texte en langage naturel en un texte semi-structuré, ceci facilite la lisibilité des scénarios et rend possible l'application de la deuxième activité du processus qui est la découverte des buts.

   L'exemple suivant montre un scénario conceptualisé et couplé au but dans un FB :

Retirer de l'argent du GAB au moyen d'une carte dans le cas normal	états initiaux : 'l'utilisateur a une carte' , 'le GAB est prêt '. 1. l'utilisateur insère une carte dans le GAB 2. le GAB vérifie la validité de la carte 3. si la carte est valide, alors 4. un message demandant le code est affiché par le GAB à l'utilisateur 5. l'utilisateur tape son code sur le clavier du GAB 6. si le code est valide, alors 7.un message demandant le montant est affiché par le GAB à l'utilisateur 8. l'utilisateur introduit le montant dans le GAB 9. si le montant est disponible, alors 10. le GAB éjecte la carte à l'utilisateur 11. si un reçu est demandé par l'utilisateur au GAB, alors 12. le GAB imprime un reçu à l'utilisateur 13. le GAB délivre les billets à l'utilisateur états finaux : 'l'utilisateur a une carte' , 'le GAB est prêt ', 'l'utilisateur a de l'argent'.

Découvrir des buts Une fois le scénario est conceptualisé, l'approche propose d'exploiter les actions de celui-ci afin de découvrir de nouveaux buts. L'approche propose trois types de découverte des buts : Alternatifs, complémentaires ou affinant le fragment de besoin en cours. Chacun de ces types s'appuie sur un ensemble des règles. Une règle est composée d'un but de règle et d'un corps de règle. Le but décrit l'objectif de la règle, et le corps fournit la description et les directives permettant de guider l'ingénieur des besoins pour mener à bien l'application de la règle [5].

Exemple : (***Règle C1***)

- **But :** Découvrir (à partir de FB <B, Sc>)$_{\text{Source}}$ (les buts complémentaires du but B)Résultat (en analysant les interactions du scénario Sc)Manière
- **Corps :**

  Etape 1 : Dans le scénario Sc identifié les objets qui correspondent à des ressources physiques

  Etape 2 : Pour chaque ressource construire les paires d'interactions (Consomma-

tion, Production)

Etape 3 : Pour chaque paire incomplète (soit avec l'interaction de consommation manquante ou avec l'interaction de production manquante) suggérer de nouveaux buts

Etape 4 : Sélectionner les buts et les nommer

Etape 5 : Le(s) but(s) sélectionné(s) est (sont) complémentaire(s) à FB.

MAP CREWS-L'Ecritoire

Si les scénarios et les buts sont bien définis en tant que produits obtenus à partir d'un processus d'IB, toutes les approches n'indiquent pas quoi faire, quand faire ou même comment faire pour les obtenir. Il est largement admis qu'un tel guidage devrait être fourni sous la forme d'un modèle de processus. Et comme solution la démarche de l'approche est modélisée par un MAP.

Le MAP est un modèle de processus vu comme un panel de processus à partir duquel, en se basant sur des directives de guidage et par la sélection dynamique et le choix de la prescription particulière qui est la mieux adaptée à la situation rencontrée [5].

Le MAP capture toutes les activités associées à la démarche de l'approche CREWSL'Ecritoire (voir la figure 1.14) telles que la formalisation des buts, l'écriture, la conceptualisation et la complétude des scénarios, la découverte des buts et la documentation des fragments des besoins [5].

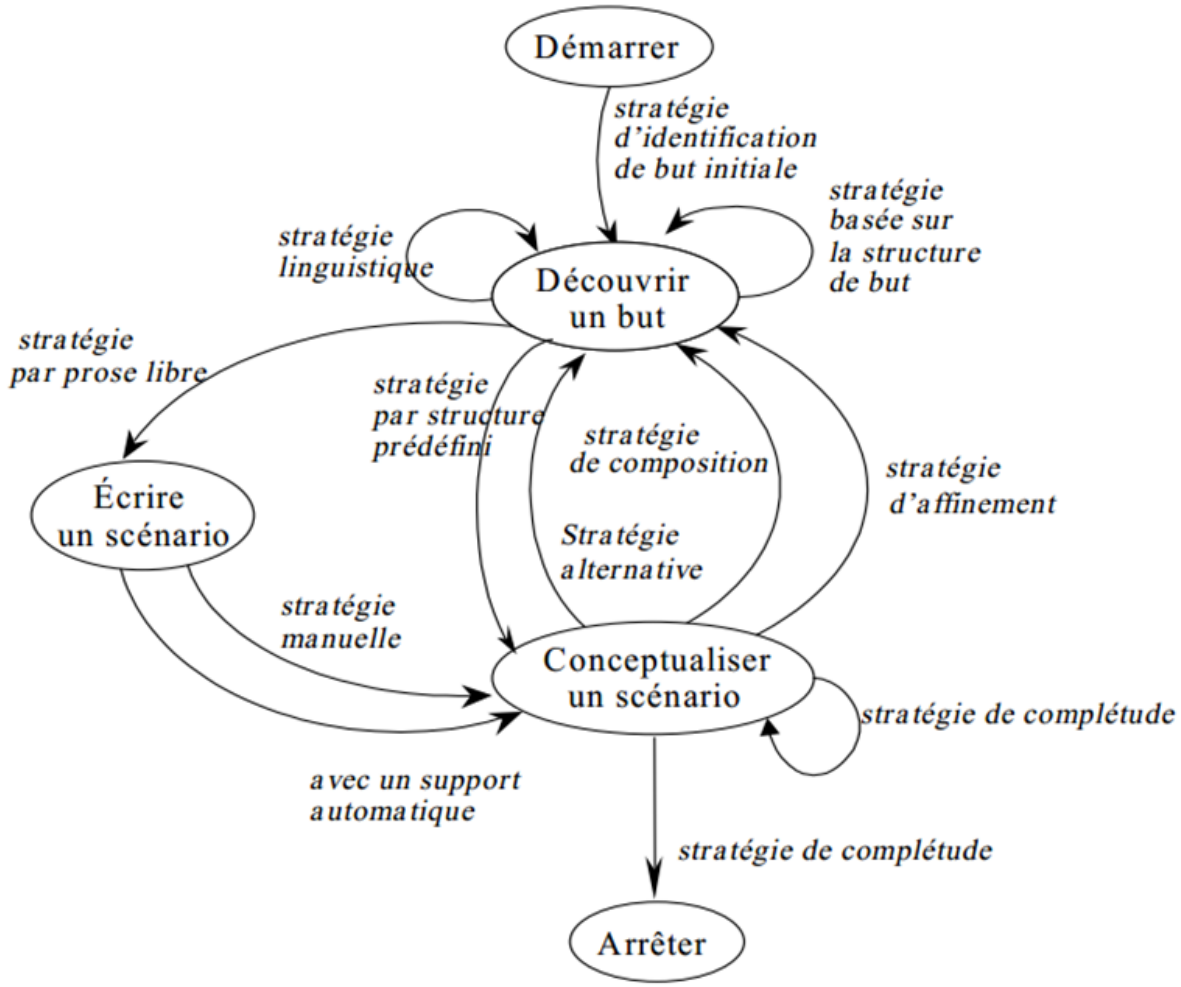

FIGURE 1.14 – Le MAP de CREWS-L'Ecritoire [5]

1.8 Conclusion

A partir de ce qu'on a vu dans ce chapitre sur les approches déjà réalisée sur l'IB, en peut conclure que l'IB représente un domaine de recherche très riche et très vaste dont il reste pas mal de choses à développer pour les gens qui veulent fiabiliser le processus de la collecte des besoins afin d'aider avant tout l'ingénieur dans son travail et sans oublier les utilisateurs qui représentent l'un des ressources du processus afin d'acquérir les bons intentions .

En revanche, les approches réalisées sur le domaine de l'IB n'ont pas encore standardisé leur processus c'est-à-dire qu'elles ne sont pas applicables partout, mais pour certains cas qui respectent certaines conditions elles peuvent s'appliquer.

Chapitre 2

Conception

2.1 Introduction

Un système d'information est un système organisé de ressources, de personnes et de structures qui évoluent dans une organisation et dont le comportement coordonné vise à atteindre un but commun.

Les systèmes d'information sont censés aider les utilisateurs dans leurs activités : stocker et restaurer l'information, faire des calculs, permettre une communication efficace, ordonnancer et contrôler des tâches,...etc.

La conception d'un système d'information nécessite une bonne réflexion sur l'ensemble des fonctions de l'organisme que l'on veut mettre en place. Donc Elle nécessite des méthodes qui permettent ca mise en place. Ces méthodes s'appellent « méthodes d'analyse », Se sont des méthodes qui consistent à créer une représentation virtuelle d'une réalité pour faire ressortir les points auxquels on s'intéresse.

Parmi ces méthodes on a : la méthode « MERSIE », la méthode « OOM », la méthode formelle « B »et sans oublier le langage « UML ».

2.2 Approche proposée pour l'ingénierie des besoins

Notre approche est basé sur le principe de la méthode orienté objet et événement(O*) en l'assemblant avec l'approche de l'écritoire (vue dans le chapitre précédent), sauf que l'approche de l'écritoire utilise des documents écrit afin de collecter les besoins des utilisateurs alors que notre approche fournit aux utilisateurs et ingénieurs des besoins une

plateforme sous web afin de mieux fiabiliser ce processus de la collecte des besoins. Notre approche est basée sur des maquettes qui permettent à la fois aux utilisateurs d'exprimer leurs besoins facilement, dans n'importe quelle place et n'importe quel instant, et à l'ingénieur des besoins de collecter et consulter les besoins des utilisateurs où il veut et quand il veut. En plus de tout ça notre approche offre une simple et efficace manipulation des maquettes ce qui peut avoir un impact positif sur les délais et sur la qualité des besoins collectés.

2.3 O* Une approche orientée objet et événement

2.3.1 Objectif

L'objectif de cette méthode est d'abstraire des souhaits, besoins, exigences et contraintes des usagers du futur système d'information puis la spécification conceptuelle de sa fonctionnalité pour avoir un schéma conceptuelle qui sert à [10] :

- Spécifier le comportement attendu du SI dans son environnement.
- Servir de support à sa conception technique et son implémentation.

2.3.2 Les principes de l'approche O*

C'est un modèle conceptuel dans le sens où il fournit un ensemble de concepts permettant d'exprimer la sémantique du monde réel, de telle sorte que la représentation obtenue soit proche de la manière dont l'être humain perçoit ce monde. Ce modèle s'appuie sur des principes qui sont [10] :

- **Le principe des 100% :** exigeant que les aspects statiques et dynamiques des phénomènes du monde réel et leurs relations soient représentés dans le même schéma conceptuel.
- **Le principe de conceptualisation :** selon lequel seuls les aspects conceptuels du système d'information en cours de développement doivent être considérés dans le schéma conceptuel.
- **Le principe de simplicité :** imposant que chaque concept du modèle possède une définition précise et une sémantique propre adaptée à son interprétation par des usagers du monde réel.

- **La puissance sémantique des concepts :** qui est d'autant plus grande que les concepts du modèle ont une sémantique proche du monde réel.
- **L'orthogonalité des concepts :** grâce à laquelle un seul concept du modèle est pertinent pour représenter une situation réelle.
- **Principe de localisation :** organiser la représentation autour de la notion d'objet
- **Principe de causalité :** modéliser les causes et les conséquences des changements d'états des objets

2.3.3 Le modèle O* et son langage

L'objet O*

Un objet de la méthode O* est définit comme suit « On définit l'objet comme une représentation informationnelle d'un phénomène du monde réel, pertinent en regard de la base de données que l'on veut développer. Il possède une identité et un état qui change au cour du temps »[10].

C'est à dire que c'est une projection d'un phénomène persistant du monde réel, qui est pertinent pour le système a développé.

Le schéma O*

Un schéma O* est une collection de schémas de classes qui définit les caractéristiques de comportement et de structure des objets de la classe, ces caractéristique sont [10] :

- **Propriétés :** elle permet d'établir un lien de composition où bien de référence entre un objet et une valeur ou bien entre un objet et un autre objet
- **Opérations :** une opération est une action dont l'exécution change l'état des objets d'une classe. En accord avec le principe d'encapsulation, l'opération fait partie du schéma de la classe des objets auxquels elle s'applique.
- **Graphes de transitions d'états :** est un graphe qui donne une vision locale du comportement des objets d'une classe dont les nœuds sont des classes d'états formant une partition sur les états possibles des objets d'une classe, et les arcs sont des transitions d'état possibles entre ces classes d'états.
- **Assertions :** ce sont des conditions qui portent sur les valeurs des propriétés des objets ou sur les cardinalités des propriétés multiples (liens de composition ou de référence multiples). Elle s'exprime sous la forme d'une assertion booléenne.

– **Evénements :** un événement décrit un ensemble d'occurrences d'événement de même nature. Cette description comprend la cause de la survenance d'une occurrence d'événement et son impact sur les objets de la base de données. Il y a trois types d'événements :
  - Un événement interne constate le changement d'état remarquable d'un objet de la base,
  - Un événement temporel survient à un instant déterminé à l'avance,
  - Un événement externe a pour origine un stimulus en provenance de l'environnement du système d'information.
– **Héritage :** sert à établir des relations entre des objets représentant des aspects différents d'un même phénomène du monde réel et qu'ils possèdent la même identité.

Voici une matrice qui positionne ces caractéristiques selon leurs concepts [10] :

	Aspects Statiques	**Aspects Dynamiques**
Aspects prescriptifs	Propriétés Héritage	Evénements Opérations
Aspects prescriptifs	Assertions	graphes de transitions détats

TABLE 2.1 – Représentation des concepts du modèle O* [10]

2.4 Outils de conception

2.4.1 UML (*Unified Modeling Language*)

En français c'est « langage de modélisation unifié », est un langage de modélisation graphique à base de pictogrammes. Il est apparu dans le monde du génie logiciel, dans le cadre de la « conception orientée objet ».il peut être appliqué à toutes sortes de systèmes [11].

Important : UML est une démarche, ne pas une méthode.

Historique

Voir le schéma suivant [12] :

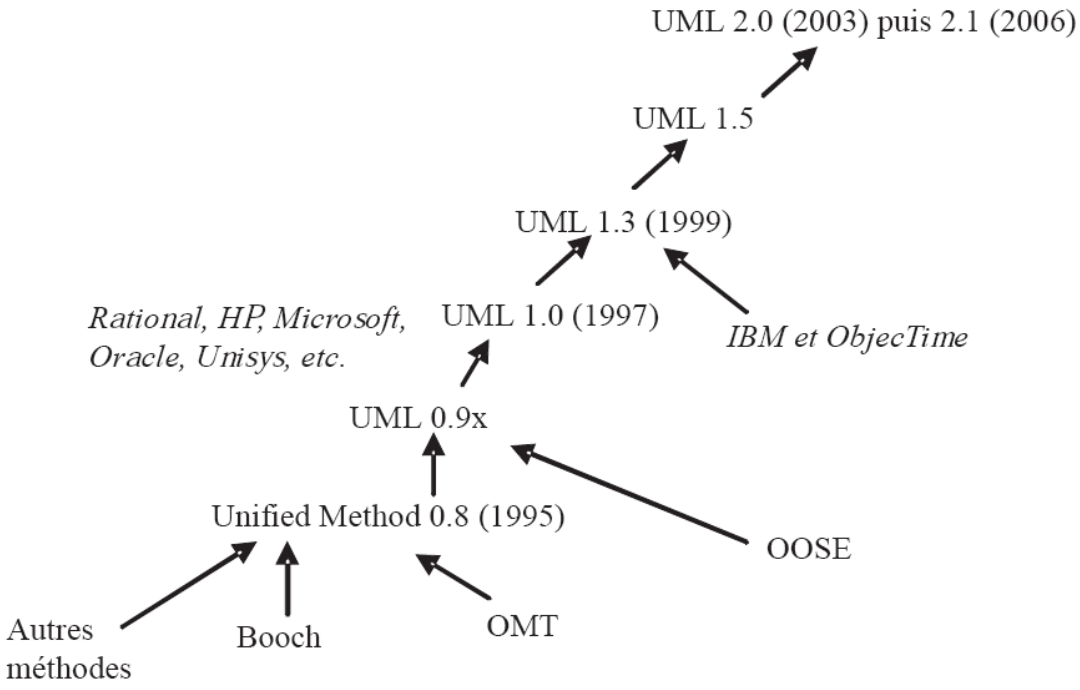

FIGURE 2.1 – Historique d'UML [12]

Explication du schéma :

A partit du schéma en déduire que UML est l'accomplissement de la fusion de précédents langages de modélisation objet :Booch, OMT, OOSE. Issu des travaux de Grady Booch, James Rumbaugh et Ivar Jackobson. UML est un standard défini par l'Objet Management Group(OMG). La dernière version diffusée par l'OMG est UML 2.3 depuis mai 2010.

Fonctionnement de l'UML

UML permet de représenter un système selon différentes vues complémentaires à l'aide des neuf diagrammes suivants :

- Diagramme des cas d'utilisation : Pour collecter les besoins des utilisateurs
- Diagramme de classes et Diagramme objet : Pour définir la structure statique du système.

- Diagramme états-transition et Diagramme d'activités : Pour définir la dynamique des objets du système.
- Diagramme de séquence et Diagramme de collaboration : Pour définir les interactions entre les objets du système.
- Diagramme de composants et Diagramme de déploiement : Pour la réalisation et le déploiement.

Chaque type de ces diagrammes possède une structure et véhicule une sémantique précise. La combinaison des différents types de diagrammes UML offrent une vue complète des aspects statiques et dynamiques du système.

Caractéristiques de l'UML

Les Caractéristiques du langage UML sont les suivantes [11] :

- UML est avant tout un support de communication performant, qui facilite la représentation et la compréhension de solutions objet.
- Sa notation graphique permet d'exprimer visuellement une solution objet, ce qui facilite la comparaison et l'évaluation de solutions.
- L'aspect formel de sa notation, limite les ambiguïtés et les incompréhensions.
- Son indépendance par rapport aux langages de programmation, aux domaines d'application et aux processus, en font un langage universel.

Remarque : la notation graphique d'UML n'est que le support du langage. La véritable force d'UML, c'est qu'il repose sur un métamodèle qui le permet de normaliser la sémantique des concepts qu'il véhicule.

Exemple : une association d'héritage entre deux classes soit représentée par une flèche terminée par un triangle ou un cercle, n'a que peu d'importance par rapport au sens que cela donne à votre modèle. Un bon langage de modélisation doit permettre à n'importe qui de déchiffrer cette intention de manière non équivoque. Le métamodèle UML a apporté une solution à ce problème fondamental. Donc UML n'est plus qu'un simple outil qui permet de dessiner des représentations mentales mais en plus que ça c'est un outil qui permet de parler un langage commun, normalisé et accessible.

2.4.2 Le MAP

Le MAP est un modèle qui fournit un support dans la sélection d'alternatives en proposant des aides méthodologiques [5]. Il est représenté comme un graphe étiqueté et dirigé qui utilise deux notions fondamentales « intention »et « stratégie »et comporte un ensemble des sections correspondant chacune à un triplé <*Intention source, Intention cible, Stratégie*>.

Une intention (représentée par un nœud) capture la notion de tâche que l'utilisateur projette d'accomplir à un moment donné du processus. Une intention est une expression en langage naturelle qui commence par un verbe auquel on associe plusieurs paramètres. Le paramètre clé dans l'expression de l'intention c'est la cible.

Une stratégie (représentée par une flèche) est une manière par laquelle une intention est réalisée. Une stratégie caractérise le flux de l'intention source (déjà réalisée) à l'intention cible (à réaliser) et la façon d'accomplir l'intention cible.

Une section s'applique sur des instances de produit. A chaque instance de produit et à chaque intention on associe une signature. Une signature associée à une intention correspond à l'état de produit que la réalisation de cette intention permet d'atteindre. L'application d'une section sur une instance de produit permet de modifier cette instance et, par la suite, de changer sa signature [5].

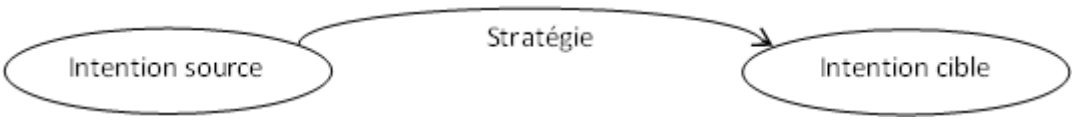

FIGURE 2.2 – Représentation graphique d'une section [5]

Utilisation du MAP

L'utilisation d'un MAP peut se résumer en trois étapes : le démarrage, la navigation et l'arrêt de l'exécution.

a) Démarrage :

L'utilisateur commence l'exécution du MAP par une section dont l'intention source est l'intention 'Démarrer'. Il choisit ensuite une intention cible parmi toutes les sections possibles (dont l'intention source est 'Démarrer'). Selon la situation de

la méthode d'origine, l'utilisateur choisit la stratégie la plus adaptée qui réalisera l'intention cible désirée.

b) Navigation dans le MAP

Chaque exécution d'une section permet à l'utilisateur de réaliser une intention générique du MAP sur une instance de produit. L'instance de produit prend ainsi la signature associée à l'intention cible. Cette intention devient donc une intention source pour cette instance de produit à partir duquel on doit naviguer pour atteindre une autre intention (cette fois-ci une intention cible). De la même manière, l'utilisateur doit tout d'abord choisir une intention cible parmi celles qui lui sont proposées par le MAP. Ensuite, il choisit une stratégie adaptée à la situation. Une fois cette intention atteinte, la navigation reprend sur le même principe.

c) Arrêt

Une fois l'utilisateur a atteint son objectif initial et qu'il souhaite arrêter l'exécution du MAP, il choisit l'intention cible Arrêter.

2.4.3 Les métamodèles

MOF (Meta Object Facility)

Le MOF est un langage de définition de méta-modèles qui contient les 4 niveaux de (méta)modélisation qui apparait dans la figure suivante [13] :

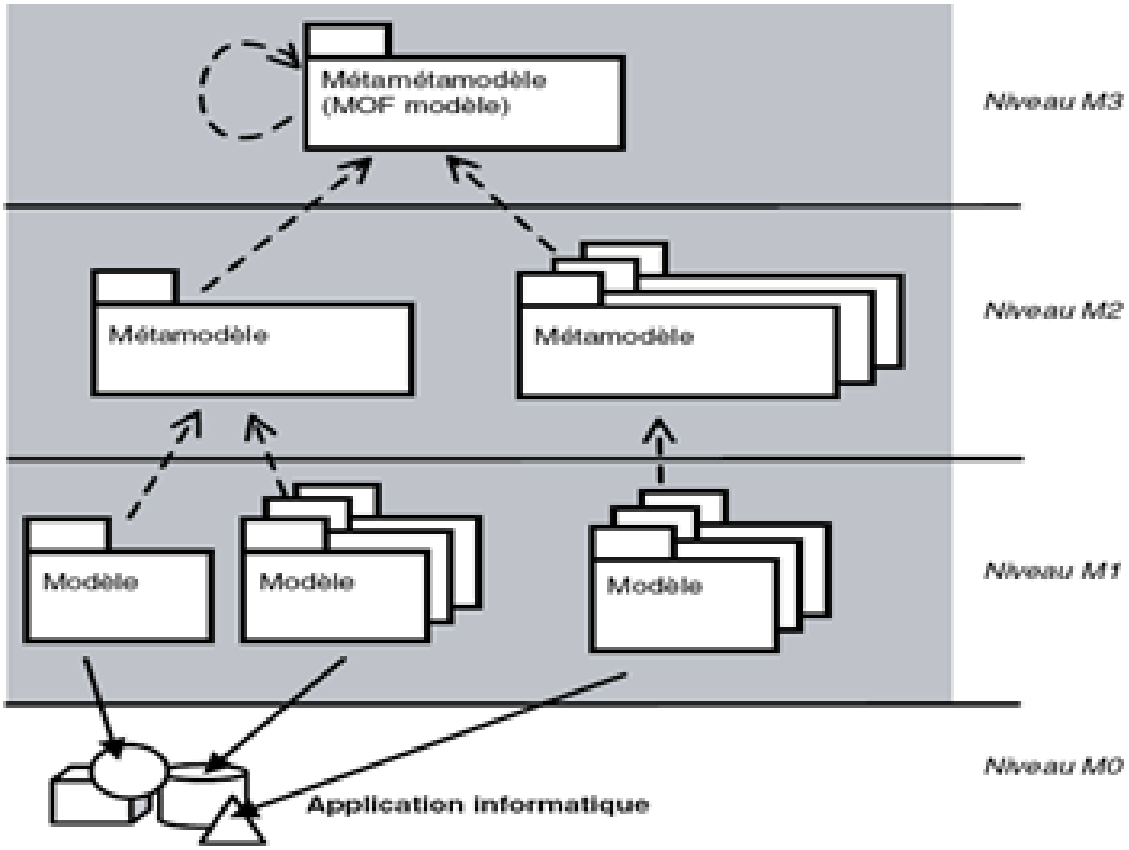

FIGURE 2.3 – Les 4 niveaux de méta-modélisation du MOF [13]

Explication du schéma :

Le MOF définit 4 niveaux de modélisation

M0 : système réel, système modélisé

M1 : modèle du système réel défini dans un certain langage

M2 : méta-modèle définissant ce langage

M3 : méta-méta-modèle définissant le méta-modèle

Remarque : Le dernier niveau M3 qui est le MOF est méta-circulaire : il peut se définir lui même

Les concepts du MOF Le MOF repose sur 4 concepts de bases qui sont [13] :

- **Class :** une métaclasse permet de définir la structure de méta-objets. Un ensemble de méta-objets constitue un modèle. Une métaclasse contient des méta-attributs et des méta-opérations.
- **DataType :** un type de donnée permet de spécifier le type non-objet d'un méta-attribut ou d'un paramètre d'une méta-opération.

- **Association :** une méta-association permet de spécifier une relation binaire entre deux métaclasses.
- **Package :** un métapackage permet de regrouper sous un même espace de nommage différents éléments d'un métamodèle.

Métamodèle

Selon MOF, un métamodèle définit la structure que doit avoir tout modèle conforme à ce métamodèle. C'est à dire tout modèle doit respecter la structure définie par son métamodèle [14].

Exemple : le métamodèle UML définit que les modèles UML contiennent des packages, leurs packages des classes, leurs classes des attributs et des opérations, etc.
La figure suivante représente la relation entre un métamodèle et l'ensemble de ses modèles.

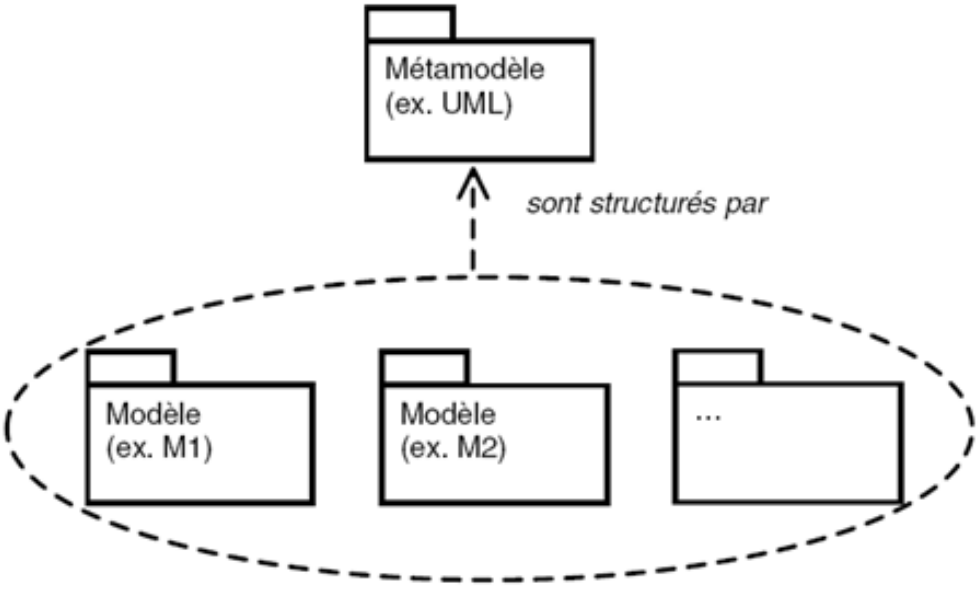

FIGURE 2.4 – La relation entre un métamodèle et l'ensemble de ses modèles [14]

Important : les métamodèles fournissent la définition des différents concepts d'un modèle où MOF les représente sous forme de diagrammes de classes [13]. Rappelons que les diagrammes de classes permettent de représenter les notions d'un domaine et leurs propriétés.

Donc l'utilisation des diagrammes de classes va permettre de définir très précisément les métamodèles et de les rendre aussi stables et productifs.

Méta-modélisation

La méta-modélisation est la modélisation des modèles à l'aide d'un langage de modélisation. Autrement dit, Représenter une méthode par la modélisation de ses parties, i.e. Ses modèles de Produit et de Processus [15].

a) Méta-modélisation du Produit :

C'est une activité qui regroupe tous les concepts et les contraintes pour construire un schéma [15].

Exemples :

- Le modèle Entité/Relation est basé sur les concepts d'entité, relation, attributs et contraintes (clefs, cardinalité).
- Chaque entité doit avoir une clef.

La représentation de l'ensemble des concepts est représentée sur un schéma appelé méta-modèle du produit.

Le modèle de produit a un rôle major dans un environnement CASE, la figure suivante nous montre ca :

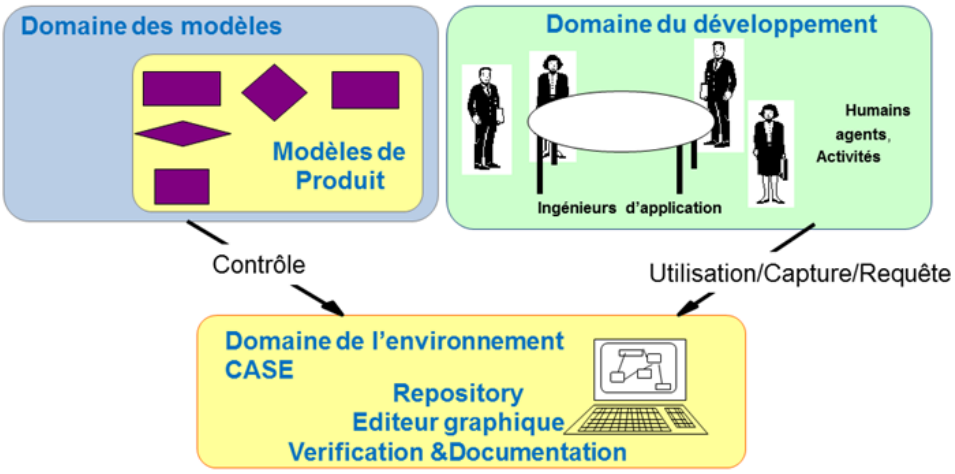

FIGURE 2.5 – Le rôle de modèle de produit dans l'environnement CASE [15]

b) Méta modélisation du Processus :

C'est une démarche qui sert à générer une liste d'étapes/activités pour la construction d'un schéma [15].

Exemples :

- La démarche de construction d'un schéma UML.
- La démarche Merise.

Le modèle de processus a aussi un rôle important dans un environnement CASE, et qui est démontré à l'aide de la figure suivante :

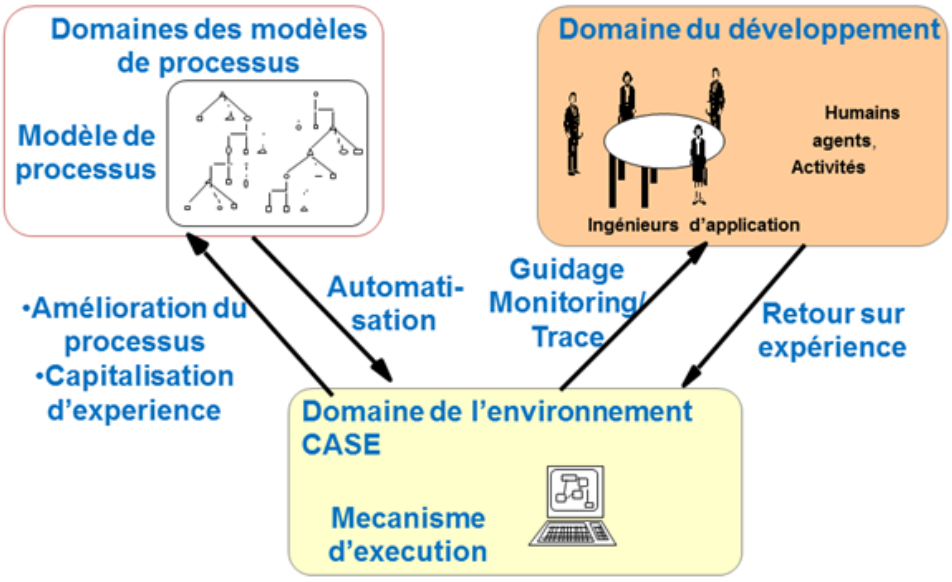

FIGURE 2.6 – Le rôle du modèle de processus dans l'environnement CASE [15]

2.5 Le méta-modèle de notre approche

2.5.1 Modèle de produit

Notre approche permet de représenter le scénario à l'aide d'une maquette, On s'est inspiré de la notion de FB utilisée dans l'approche de l'ECRITOIRE pour représenter le couple [But, Maquette].

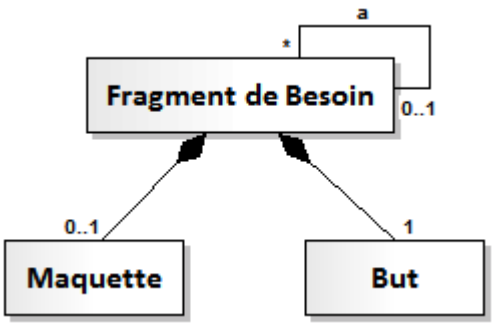

FIGURE 2.7 – Composition de fragment de besoin

Ce fragment de besoin est composé d'un but et d'une maquette assignée à ce but, l'ensemble de ces fragments de besoin sont représentés par une arborescence produite par la relation (a) qui signifie les buts et les sous buts.

FIGURE 2.8 – Arborescence des fragments de besoins

Les buts sont classés soit fonctionnels faisant référence à des fonctions du système, soit non fonctionnels référençant à des aspects liés à la sécurité, la performance, la robustesse, la facilité d'utilisation, l'interopérabilité,. . .etc. Et ayant un scénario détaillé sur l'ensemble d'interactions entre le système lui-même et ses utilisateurs.

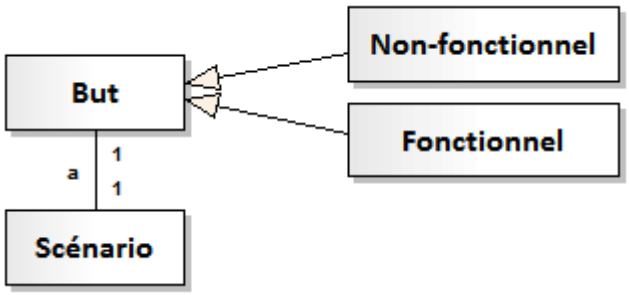

FIGURE 2.9 – Modèle de produit coté but

En se basant sur le principe de la méthode orienté objet & évènement (O*), où une maquette représente une classe et se compose d'un ensemble d'éléments qui ressemblent aux attributs d'une classe dans le schéma O*.

Ces éléments peuvent être assignés à des données et/ou à des évènements dont le quel ces évènements doivent déclencher une ou plusieurs opérations du système. Et pour chaque donnée on a un type et une taille.

Chaque donnée ou événement peut avoir une ou plusieurs assertions qui représentent aussi le prédicat dans le cas de l'évenement en méthode O*.

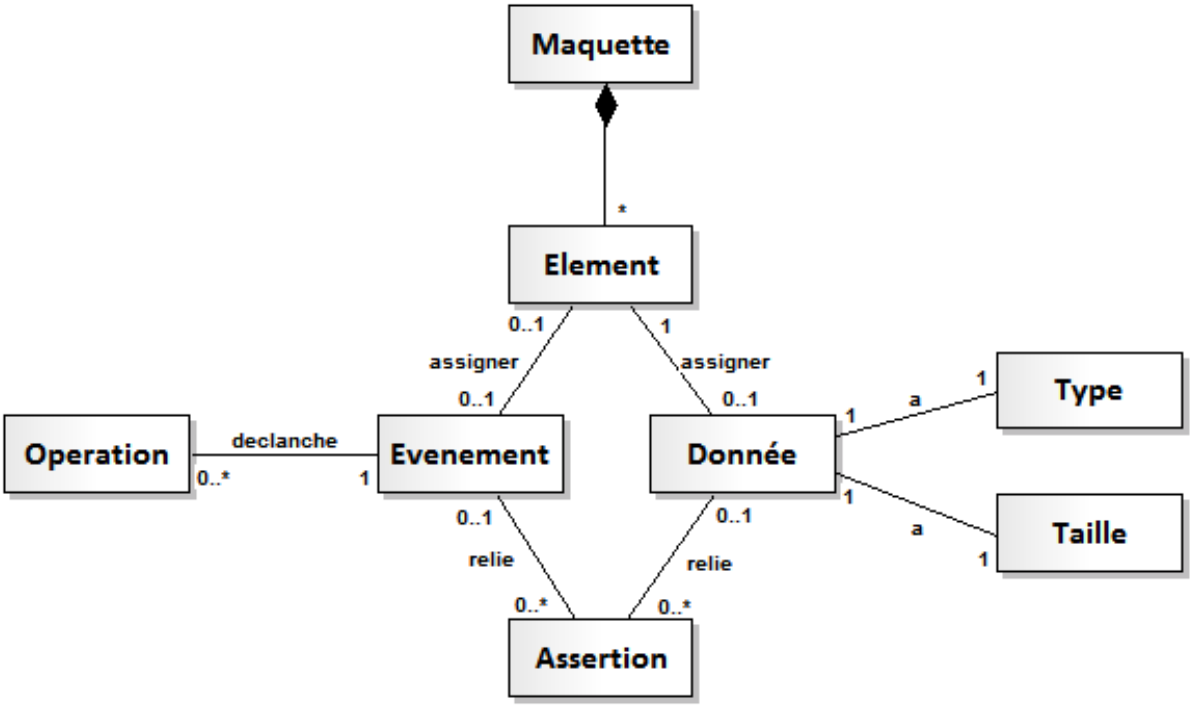

FIGURE 2.10 – Modèle de produit coté maquette

Le tableau suivant represente une comparaison entre la structure de notre approche (maquette) et la méthode orienté objet et événement (classe) :

Schéma de Classe avec la méthode O*	**Schéma de Maquette**	
classe	Maquette	
[propriétés] liste des caractéristiques	[élément]	[données]
		[évènements]
[évènements liste des évènements]	[évènements]	
[assertions] liste des contraintes statiques	[assertions]	
[opérations] liste des opérations	[opérations]	

TABLE 2.2 – Comparaison entre Schéma de Classe du méthode O* et Schéma de Maquette

La figure suivant représente le modéle de produit globale :

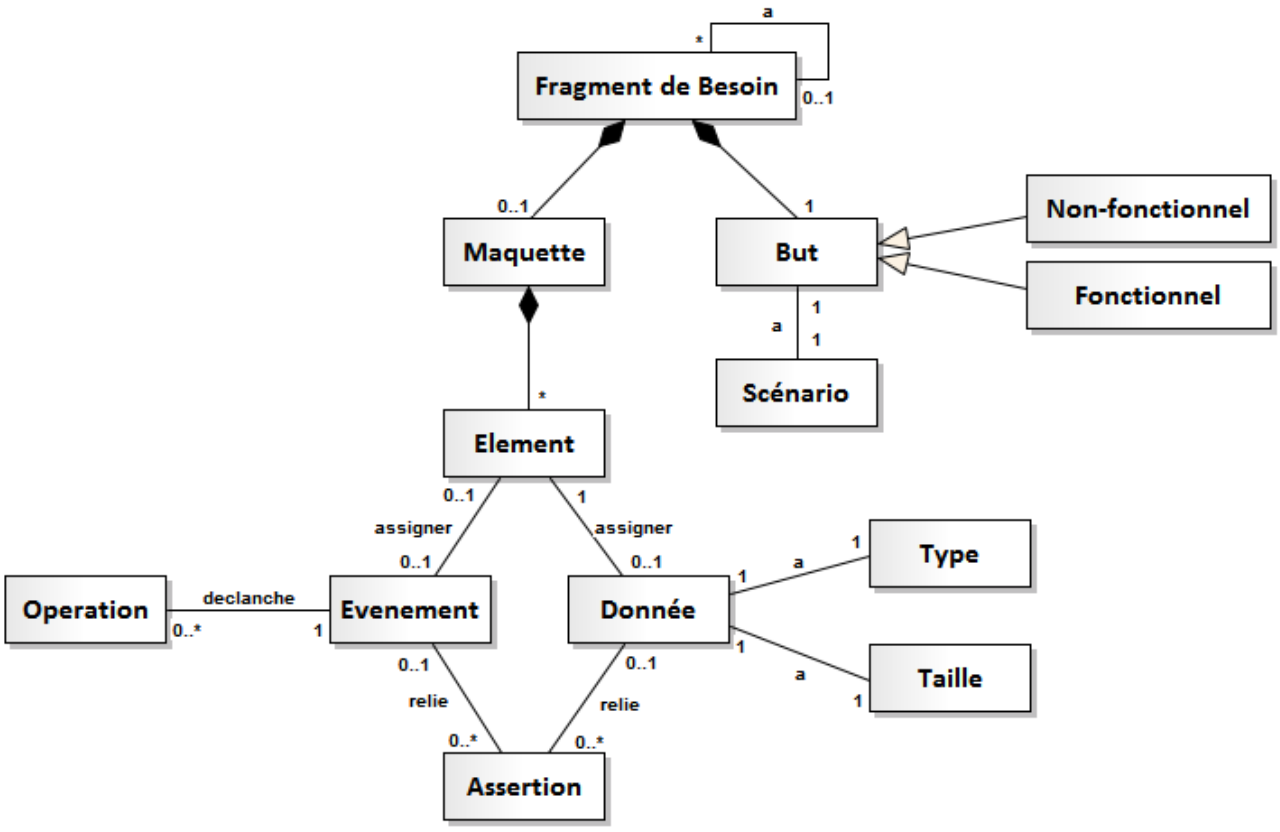

FIGURE 2.11 – Modèle de produit globale

2.5.2 Modèle de processus

La figure suivante représente le modèle de processus de notre approche :

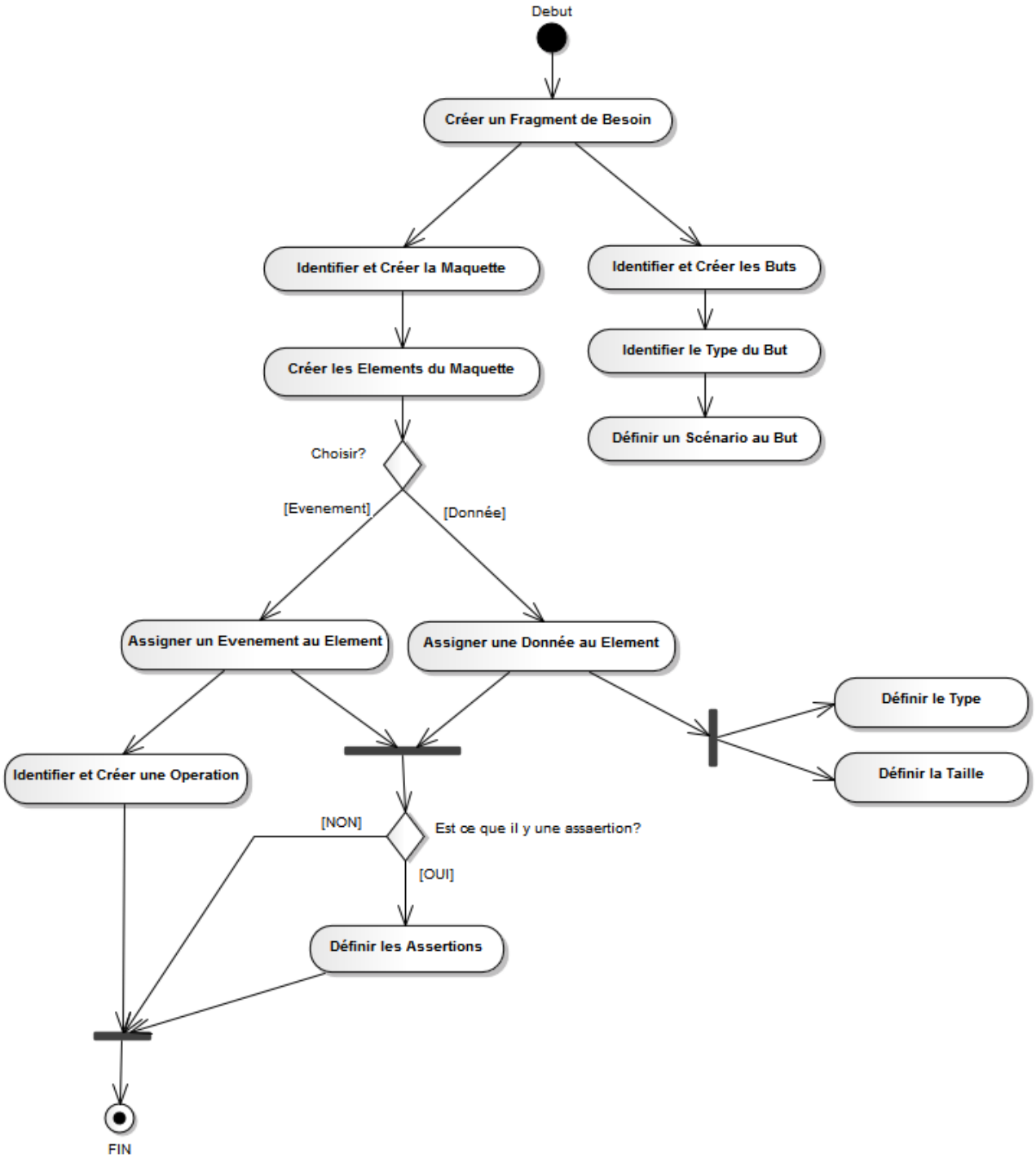

FIGURE 2.12 – Le modèle de processus de notre approche

2.6 Diagramme de classe

Le schéma conceptuel suivant contient des classes, attributs, méthodes et associations qui se rassemble pour représenter le corps de notre approche, de plus son aspect gestion qui offre et assure un meilleur fonctionnement :

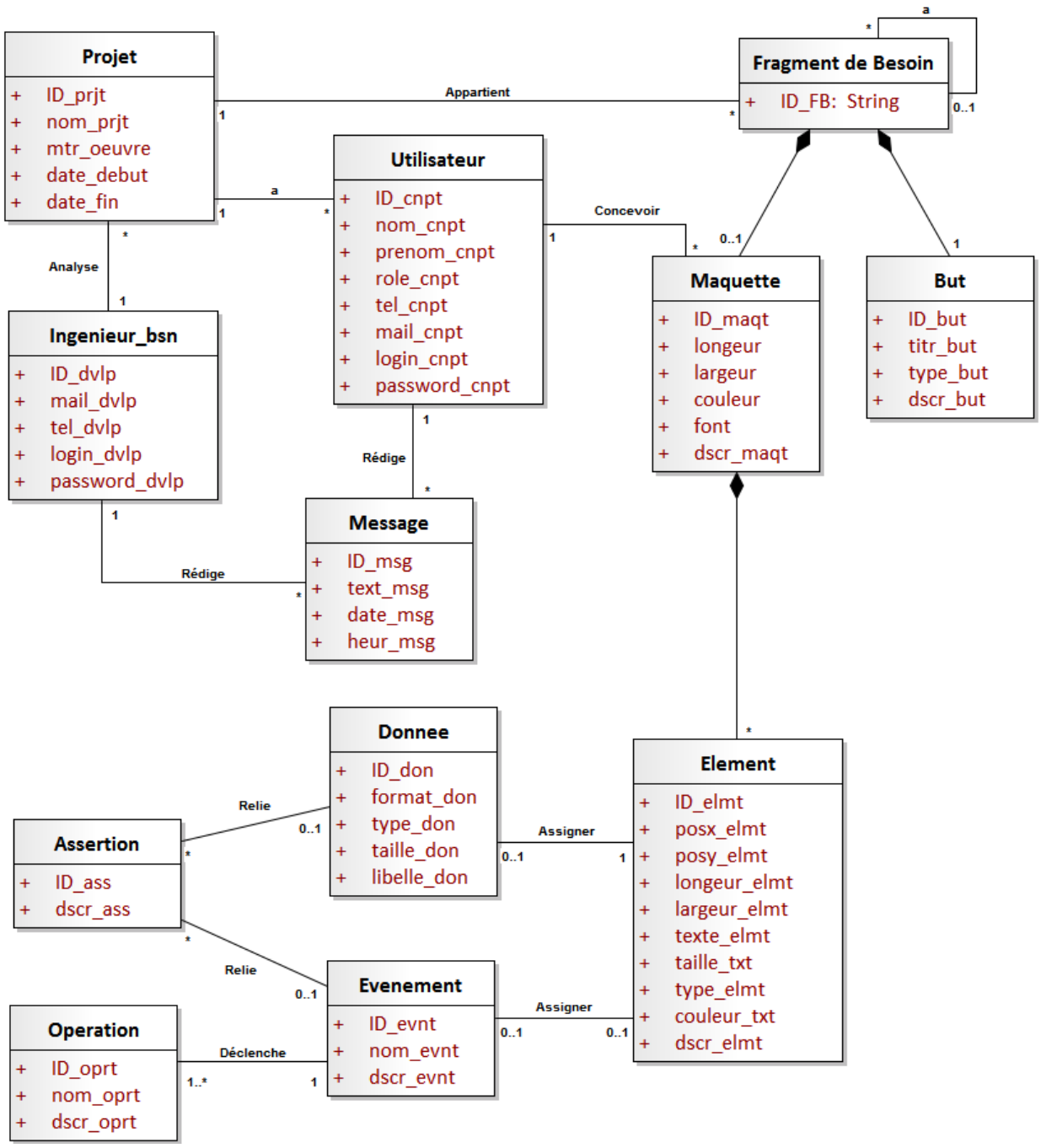

FIGURE 2.13 – Le diagramme de classe de notre approche

2.7 Le passage au relationnel

Afin d'implanter notre base de données sur un SGBD relationnel, Nous sommes obligés de migrer vers le relationnel, La figure suivante représente le modèle relationnel de notre base de données :

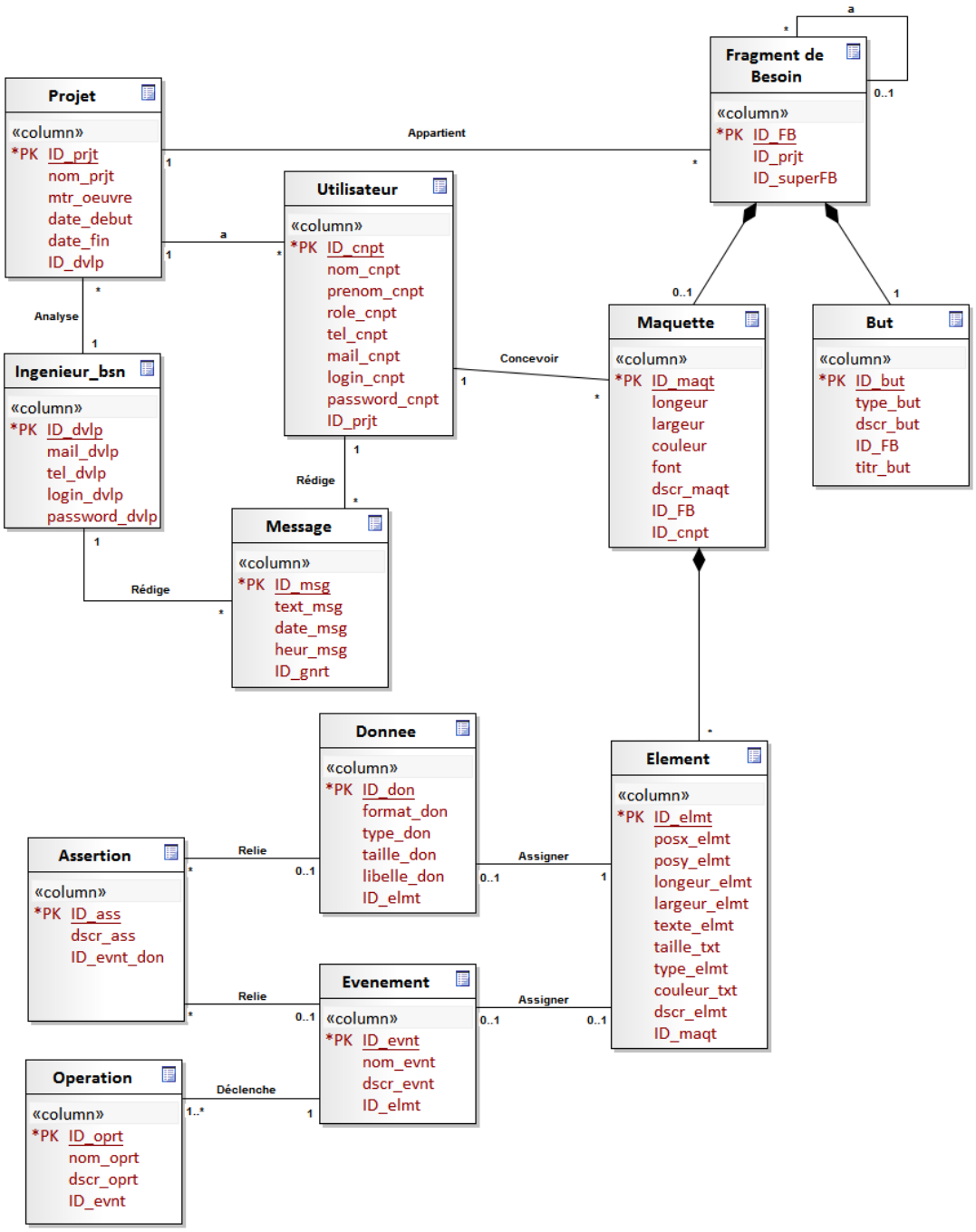

FIGURE 2.14 – Le modèle relationnel de notre approche

2.8 Démarche

Afin d'améliorer le rendement de notre outil nous avons mis en place une démarche qui est schématisé par le moyen du MAP suivant :

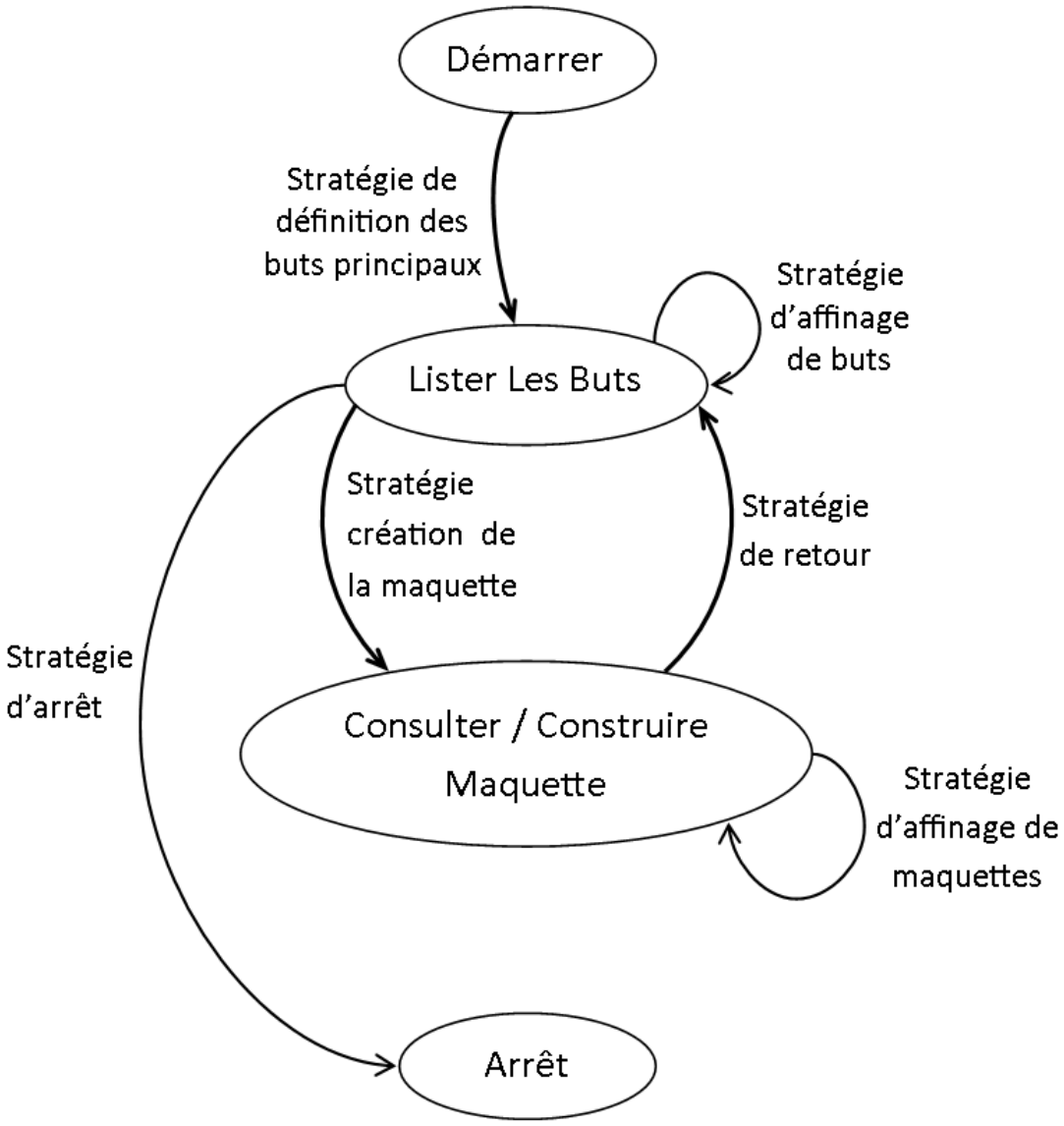

FIGURE 2.15 – La démarche de notre outil

Explication du MAP (figure 2.15) :

1. **Stratégie de définition des buts principaux :**

   Quand un ingénieur de besoin décidera de collecter les besoins d'un système, il doit réunir avec l'utilisateur principale (Exemple : directeur) pour discuter sur le fonctionnement générale du système sans entrer dans les détails. A partir de cette discussion l'ingénieur peut extraire les principaux buts (fonctionnalités) du système

à étudier.

Remarque : Car cette phase de début est très importante et très délicate (tout le reste du travail se base sur les résultats de cette phase), on a décidé de la faire manuellement.

Dans cette phase, l'ingénieur doit collecter les contacts des utilisateurs.

2. **Stratégie d'affinage de buts :**

   Après le listing des buts principaux, l'ingénieur créera des comptes pour les utilisateurs, ensuite un message qui contient un login et un mot de passe sera transmis par émail de l'ingénieur vers les utilisateurs concernés pour les donner le feu vert à travailler sur l'outil.

   Après que l'utilisateur reçoit le message, il commence à travailler en :

   – Consultant, Ajoutant, modifiant, Supprimant des buts et sous buts.
   – Consultant, Ajoutant, modifiant, Supprimant des maquettes.
   – Envoyant des messages à l'ingénieur.

   L'ingénieur à son rôle, il doit consulter ce que les utilisateurs ont fait, en :

   – Consultant, Ajoutant, modifiant, Supprimant des buts et sous buts.
   – Consultant, Ajoutant, modifiant, Supprimant des maquettes.
   – Envoyant des messages aux utilisateurs.

   Remarque : Les messages envoyés par l'utilisateur seront affichés dans la page d'accueil de l'ingénieur et le contraire.

   Quand un utilisateur effectue une modification, un message est transmis vers l"ingénieur automatiquement pour l'indiquer qu'il y a un changement et le contraire.

3. **Stratégie création de la maquette :**

   Pour qu'un utilisateur ou bien un ingénieur veut créer ou consulter une maquette, il suffit jute de cliquer sur le lien « consulter maquette »qui se retrouve au-dessus d'un but fonctionnel et ne contient pas des sous buts afin d'être dirigé vers la page « maquette »où il peut ajouter, modifier ou supprimer des éléments, événements, opérations et assertions avec des simples clics.

4. **Stratégie d'affinage de maquettes :**

   Dans la page maquette l'utilisateur ou bien l'ingénieur peut :

   – Ajouter des éléments, événements, opérations et assertions.

- Modifier les propriétés des éléments, événements, opérations et assertions.
- Définir la donnée d'un élément s'il existe.

5. **Stratégie d'arrêt :**

   Quand l'utilisateur sera satisfait c'est-à-dire qu'il a exprimé tous ces besoins, il envoie un message vers l'ingénieur pour lui indiquer qu'il a terminé. Après avoir effectué une vérification total par l'ingénieur et s'il n y a rien à modifier, il confirme la terminaison du travail par un message envoyé à l'utilisateur et le travail est terminé.

2.8.1 Recommandation et bons pratiques

Voici quelques recommandation et bons pratiques pour une meilleure utilisation :

- Il faut que l'ingénieur offre une formation à l'utilisateur sur l'usage de l'outil.
- Il faut que l'ingénieur et l'utilisateur soient d'accord sur les délais du projet.
- Il est préférable que l'ingénieur possède une copie sur les déférents documents utilisés par l'organisme (en papier ou bien électroniques).

2.9 Conclusion

Jusque-ici, on a terminé avec la partie la plus important de notre travail. On a conçu les deux métas modèles de produit et de processus. Plus que ça, on est arrivé à additionner l'aspect gestion à notre approche et de concevoir le diagramme de classe. Enfin on a clôturé par une migration vers le modèle relationnel qui va être le support de notre base de données.

Chapitre 3

Implémentation

3.1 Introduction

Notre approche sert à offrir aux utilisateurs et aux ingénieurs une plateforme sous web qui permet d'un côté aux utilisateurs d'exprimer leurs besoins et d'autre côté aux ingénieurs des besoins de collecter les besoins de leur utilisateurs. Donc cette plateforme joue un rôle majeur dans le processus de la collecte des besoins dans le but de fiabiliser ce processus et d'atteindre les résultats souhaitées dans les meilleurs délais.

3.2 Outils d'implémentation

Pour arriver à mettre en place notre plateforme, on a besoin d'utiliser les outils suivants :

3.2.1 Langages

HTML (HyperTextMarkupLanguage)

HTML est un langage à balisage qui décrit la structure logique d'un document hypertexte, Né en 1989 sous l'impulsion de « TimBernersLee »inventeur du Web. Il est basé sur SGML(StructuredMarkupLanguage), qui est une vieille norme utilisée pour la description de documents [16].

Ce langage contient des commandes implémentées par des balises pour marquer les différents types de texte (titres, paragraphe, listes...), inclure

des images, des formulaires, des liens . . .etc.

La dernière version du langage est l'HTML 5. L'extension d'un fichier HTML est (.html).

Un fichier HTML doit comporter au minimum ces 4 balises :

- <html> . . .</html> qui délimite le début et la fin du document.
- <head> . . .</head> qui représente l'entête du document.
- <title> . . .</title> qui contient le titre du document.
- <body> . . .</body> qui contient le corps du document.

CSS(Cascading Style Sheet)

Dès 1994, devant les relatives faiblesses du Html en matière de présentation, l'idée d'adjoindre des styles au Html comme dans les logiciels de traitement de texte fait son apparition. En 1995, le W3C (WWWConsortium) à réaliser une première ébauche sur les feuilles de style qui est publier en novembre 1995, ce document devient une recommandation officielle le 17 décembre 1996 sous le nom de "Feuilles de style en cascade niveau1" (Cascading Style Sheets, level 1) CSS1 [17].

Ces feuilles de style permet de séparer les données et la mise en forme pour :

- Améliorer l'accessibilité.
- Faciliter la maintenance.
- Faciliter le portage pour un autre site.
- Simplifier et aérer le code.
- Uniformisation du code entre les pages du même site.

La dernière version du langage est le CSS 3.

a) Les différents niveaux d'insertion du code css :

Le code CSS peut être placé dans [18] :

- Une balise, via un attribut.
- Un script intégré dans l'entête de la page web.
- Une feuille externe.
- Plusieurs feuilles externes.

b) La syntaxe du CSS pour une feuille externe :

```
selecteur {
propriete1 : valeur1 ;
propriete2 : valeur2 ;
...
}
```

c) Sélecteur : c'est un pattern qui sélectionne l'élément ou le groupe d'éléments sur lesquels va s'appliquer le style [18].

d) Types de sélecteur possibles :

- Une balise.
- Un nom de classe qui est introduit par(.).
- Un identifiant d'élément qui est introduit par (#).

Exemple :

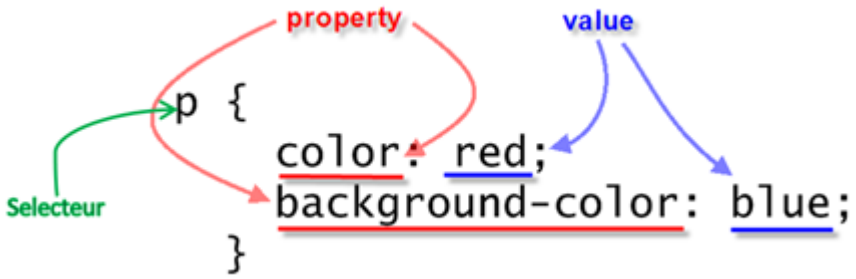

FIGURE 3.1 – Code de CSS

JavaScript

Le JavaScript est un langage de programmation de script orienté objet, il a été créé par la société « NetScapeapparu »en 1996 puis normalisé en Juin 1997 par l'ECMA (European Computer Manufactures Associations) [19].

Ce langage permet au développeur de faire :

- Petites applications : calculettes, éditions de texte, jeux, ...

– Aspect graphiques : modification d'images, gestion de fenêtres, modification locale de la page HTML, ...
– Test de validité des champs de saisies : vérifier si une zone de saisie a le bon format, ...

Le script est un programme dont le code source est inclus dans un document HTML. Ce programme est interprété et s'exécute sur la machine du client lorsque le document est chargé ou lors d'une action de l'utilisateur.
La dernière version du langage est le JavaScript 2.

a) Fonctionnement du JavaScript :

La figure suivante illustre le fonctionnement du JavaScript [20] :

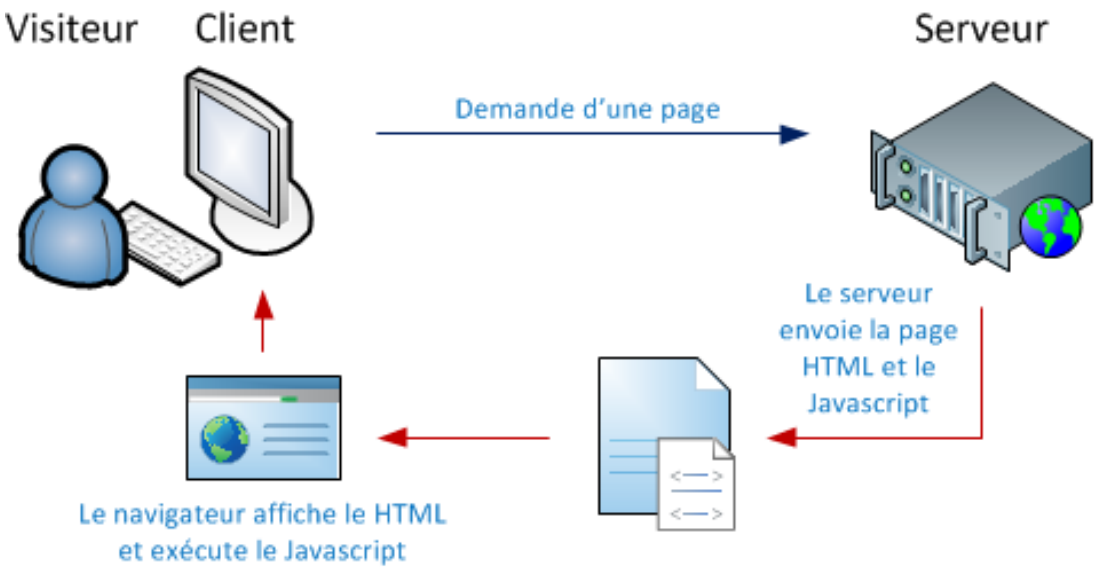

FIGURE 3.2 – Fonctionnement du JavaScript [20]

b) Les différents niveaux d'insertion du code JavaScript :

Le code JavaScript est inséré comme suit [19] :

– Soit utiliser la balise < SCRIPT>.
– Soit utiliser un fichier externe < SCRIPT SRC="fichier.js">.
– Soit dans un attribut de balise.
– Soit en gestion d'événements

Exemple :

```
<html>
<head>
<title>Glups</title>
</head>
<body>
<h1>Glups</h1>
Avec un bon navigateur
<script language="JavaScript1.2">
document.writeln("(comme "+navigator.appName+")")
</script>
on peut realiser des documents personnalises et meme dynamiques.
</body>
</html>
```

SQL (Structured Query Language)

En français c'est « Langage d'interrogation structuré ».En fait SQL est un langage complet de gestion de bases de données relationnelles. Il a été conçu par IBM dans les années 70 puis normalisé en 1986 pour qu'il devienne le langage standard des systèmes de gestion de bases de données relationnelles(SGBDR) [21].
Le SQL est à la fois :

- Un langage d'interrogation de la base (ordre SELECT)
- Un langage de manipulation des données (LMD ; ordres UPDATE, INSERT,DELETE)
- Un langage de définition des données (LDD ; ordres CREATE, ALTER,DROP),
- Un langage de contrôle de l'accès aux données (LCD ; ordres GRANT,REVOKE).

Le langage SQL est utilisé par les principaux SGBDR : Mysql, DB2, Oracle, Informix, Ingres, RDB,...

PHP

Le langage PHP a été conçu en 1994 par Rasmus Lerdorf. Ce langage de programmation permet essentiellement de construire des sites Web dynamiques et particulièrement lorsqu'ils sont reliés à une base de données.

Le plus important avantage du langage PHP est le support d'un grand nombre de bases de données comme : InterBase PostgreSQL, dBase, FrontBase, Direct MS-SQL, MySQL, IBM, ODBC, Oracle ...

Le langage PHP inclus le support des services utilisant les protocoles tels que IMAP, SNMP, NNTP, POP3 et http [22].

a) Fonctionnement du PHP :

La figure suivante nous montre le fonctionnement du système lorsque le serveur a reçu une demande par un client ou un navigateur [23].

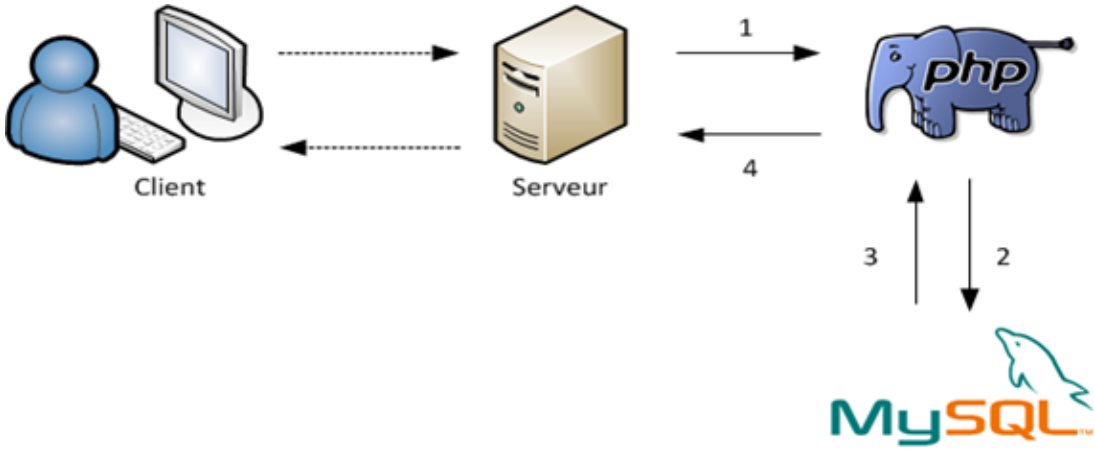

FIGURE 3.3 – Fonctionnement du langage PHP [23]

Explication :

1. Le serveur fait passer un message toujours en PHP.
2. PHP effectue les actions demandées et se rend compte qu'il a besoin de MySQL.
3. MySQL fait le travail que PHP lui avait soumis et lui répond « OK, c'est bon ! »
4. PHP renvoie au serveur que MySQL a bien fait ce qui lui était demandé.

La dernière version du langage est le PHP 5.

Remarque :

nous voyons clairement que le PHP a le rôle d'un intermédiaire entre le serveur et le SGBD(MySQL), pour cette raison des fois il est appelé PHP-MySQL [23].

b) Les différents niveaux d'insertion du code php :

Le code PHP est insérer comme suit [22] :

- En utilisant Les balises par défaut : < ?php ...Code PHP ... ?>
- En utilisant Les balises <script> <SCRIPT LANGUAGE="php">...Code PHP ...</SCRIPT>
- En utilisant Les balises d'ouverture courtes : < ? ...Code PHP ... ?>

Exemple :

```
<html>
<body>
<?php echo "Bonjour"; ?>
</body>
</html>
```

3.2.2 Technique

AJAX

AJAX est l'acronyme de « Asynchronous Javascript And XML »ce qui transcrit en Français « Javascript Et XML Asynchrones ».

a) Fonctionnement

AJAX est une philosophie qui regroupe un ensemble de technologies destinées à réaliser des mises à jour du contenu d'une page web sans un rechargement de page web visible pour l'utilisateur, la figure suivantes permet d'illustrer se fonctionnement [24] :

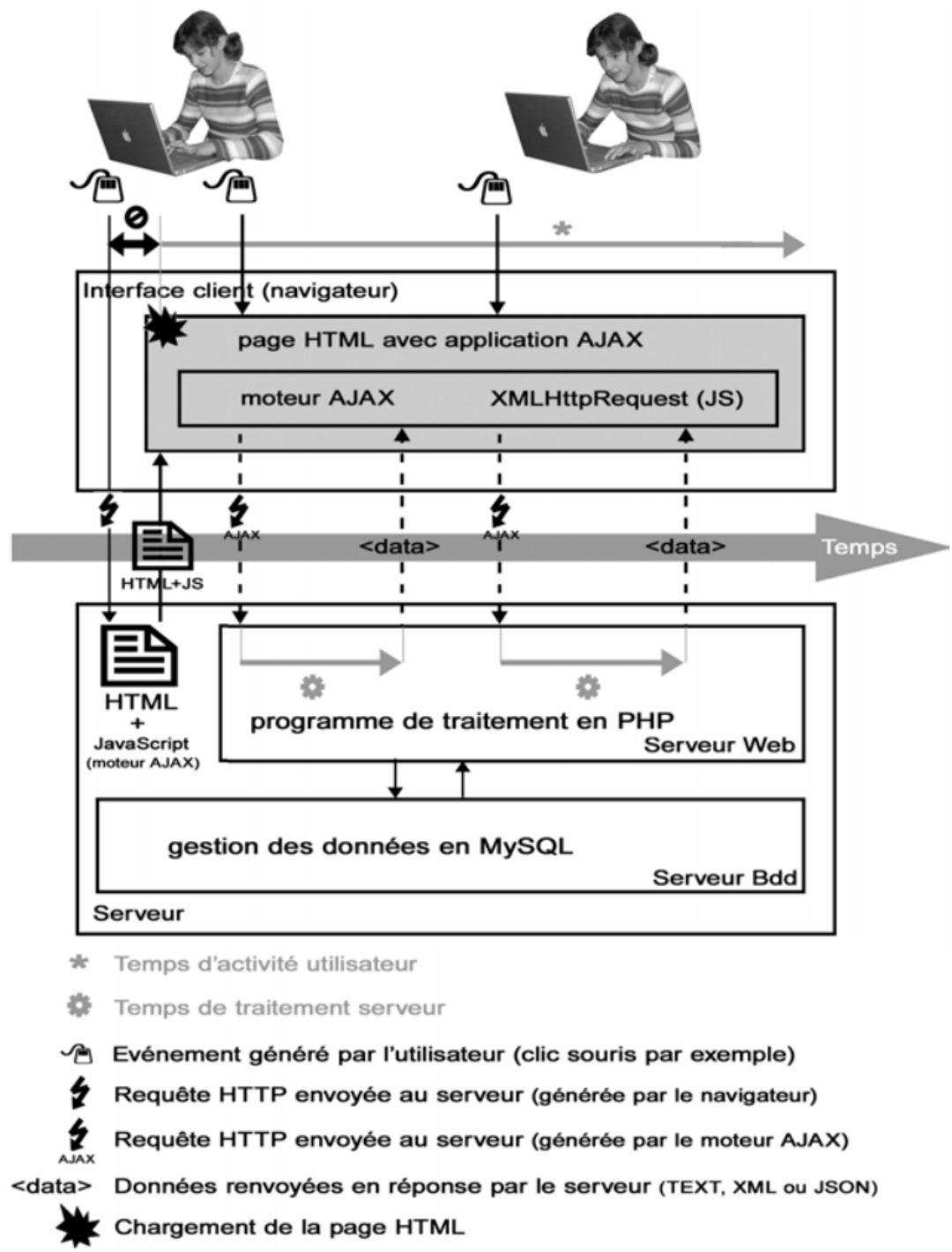

FIGURE 3.4 – Fonctionnement de l'AJAX [24]

Explication :

D'abord on commence par le langage xHTML qui va structurer le document ensuite le CSS qui va prendre en charge l'affichage de ce document. Le DOM (Document Object Modèle) qui est un standard qui définit la manière d'accéder à l'arborescence du document en le couplant avec javascript en peuvent piloter le document, XMLHTTPRequest qui est une technologie qui permet de s'appuyer sur le Protocol http et de communiquer la page web actuel avec le serveur sans rechargement de la page où les informations récupérées par le XMLHTTPRequest est du XML, avec cet échange on pourra mettre à jour certaine parties du document , enfin la Librairies Frameworks qui facilite l'application du AJAX [25].

Les technologies employées sont diverses et dépendent du type de requêtes que l'on souhaite utiliser, mais d'une manière générale le Javascript est constamment présent.

b) Les formats de données : Comme dit plus haut, l'AJAX est un ensemble de technologies visant à effectuer des transferts de données. Il existe quatre principaux formats pour le transfert des données qui sont [24] :

- Textuel est le plus simple mais il ne possède aucune structure prédéfinie, il s'agit d'une chaîne de caractères, rien de plus.
- **HTML** est un autre format simple de transfert des données, il a pour but d'acheminer des données qui sont déjà pré formatées par le serveur puis les affichés directement dans la page sans aucun traitement préalable de la part du Javascript.
- **XML** est un format qui permet de stocker les données dans un langage de balisage semblable à l'HTML, il est très pratique pour stocker de nombreuses données ayant besoin d'être formatées d'une manière que l'on peut facilement parcourir.
- **JSON** qui est le format le plus courant, il a pour particularité de segmenter les données dans un objet Javascript, il est très pratique pour de petits transferts de données segmentées.

3.2.3 Environnement de développement

WAMPSERVER

Le WAMPSERVER est un Environnement de développement qui permet de construire des applications web dynamiques et qu'il offre les outils nécessaires pour les manipuler [23].

- Le PHPMyadmin comme un serveur MySQL.
- L'Apache comme un serveur web.
- L'interpréteur de script (PHP).

Manipulation :

Une fois l'installation du Wampserver est terminée, lancer son exécution. L'icône du Wampserver va apparaitre sur la barre des tâches sous forme d'un demi-cercle blanc.

Remarque : si la couleur du demi-cercle est rouge ou jaune, cela veut dire que l'un des deux serveurs (Apache,MySQL) a du mal démarrer, ou les deux, et il faut les redémarrer.

a) Le serveur web(Apache) :

En donnant l'adresse du 'LocalHost' qu'est un une adresse IP ou un nom de domain (localhost .ex) au navigateur, puis le serveur web va lancer sa page d'accueil suivante :

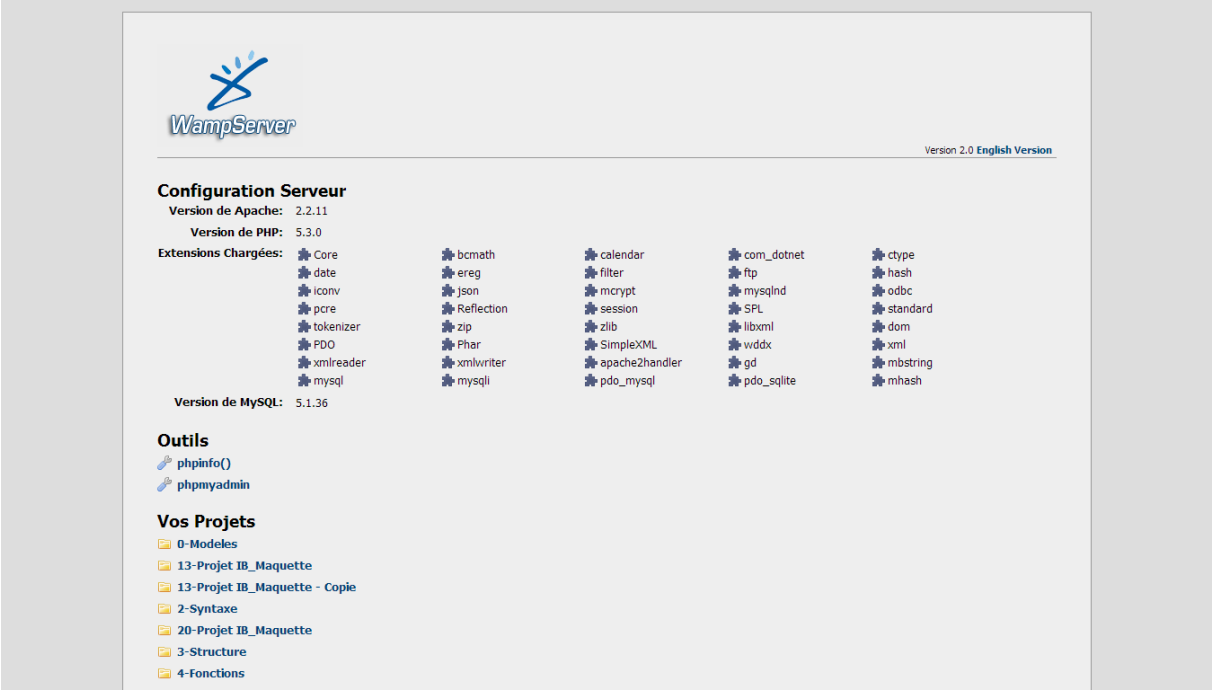

FIGURE 3.5 – La page d'accueil du serveur web

b) Le serveur MySQL : MySQL est un serveur de base de données SQL très rapide, multi-threadé, multi-utilisateur et robuste. Le serveur MySQL est destiné aux missions stratégiques et aux systèmes de production à forte charge, ainsi qu'à l'intégration dans des logiciels déployés à grande échelle.

3.3 Structure de l'outil

Notre outil est structuré en 3 parties qui sont :

3.3.1 Partie « Authentification »

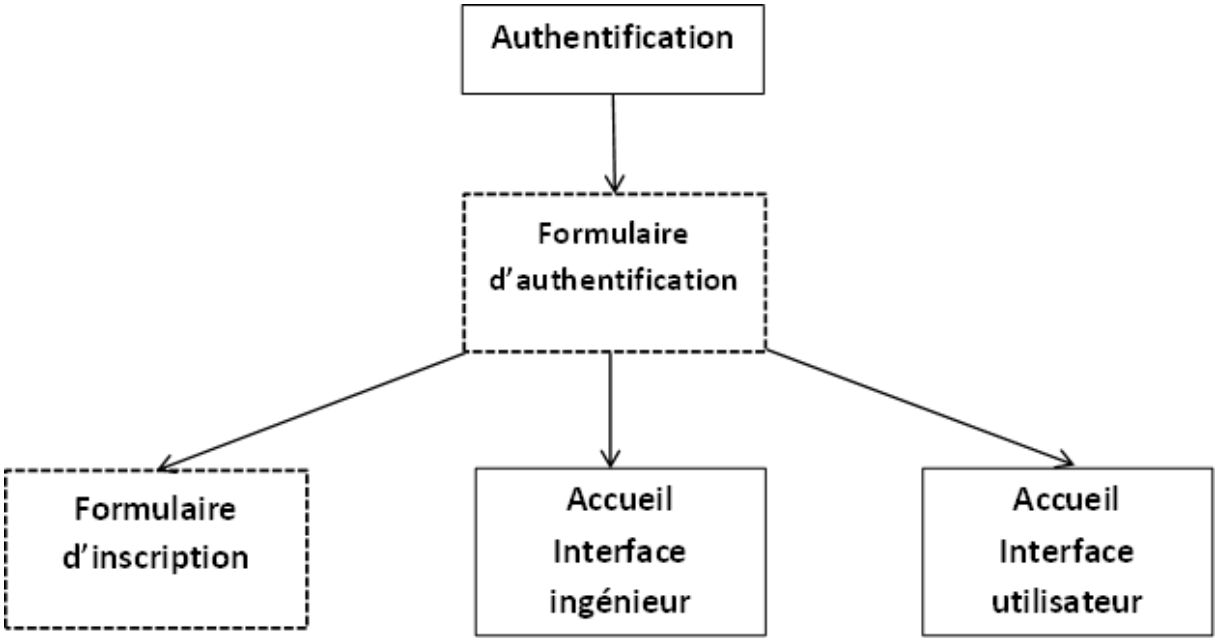

FIGURE 3.6 – Le schéma de la partie « Authentification »

Explication :

La première page qui s'affiche à l'ingénieur de besoin ou bien à l'utilisateur est la page d'authentification, où ils (ingénieur et utilisateur) doivent saisir leur login et mot de passe pour qu'ils se dirigent chacun à son propre page. Pour les ingénieurs qui n'ont pas encore inscrivent, ils doivent d'abord s'inscrire en cliquant sur le lien « s'inscrire »afin d'être dirigé vers la page d'inscription.

3.3.2 Partie « Ingénieur du besoin »

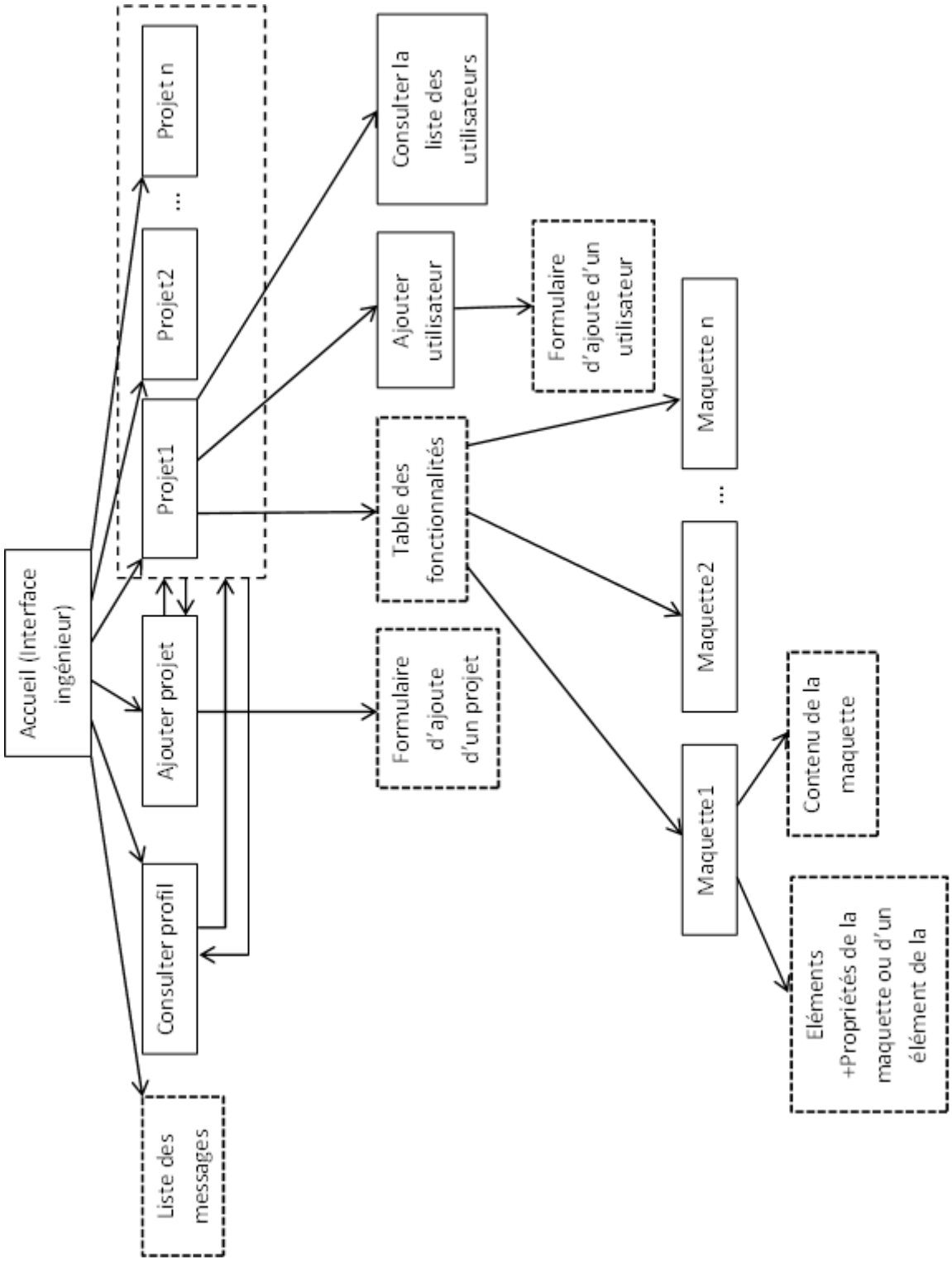

FIGURE 3.7 – Le schéma de la partie « Ingénieur du besoin »

Explication :

Après que l'ingénieur de besoin s'authentifie, il sera guidé vers la page d'accueil de l'ingénieur où il peut :

- Consulter la liste des messages envoyés par les utilisateurs, chaque message est accompagné par son projet.
- Déplacer vers la page « consulter profil »pour qu'il consulte ces informations ou bien les modifie.
- Déplacer vers la page « Ajouter projet »pour créer un nouveau projet.
- Déplacer vers la page « Projet »afin de travailler sur le projet concerné.

La figure suivante représente la page d'accueil de l'ingénieur :

FIGURE 3.8 – La page d'accueil de l'ingénieur de besoins

Pour ce qu'il concerne la page d'un projet, on retrouve les possibilités suivantes :

- Table des fonctionnalités (voir la figure 3.9) où on peut ajouter, modifier, supprimer un but ou ajouter des sous but .Les buts d'un projet sont structurés sous forme d'une arborescence à l'aide d'un tableau. Donc chaque élément du tableau représente un but ou un sous but où on peut modifier ces propriétés qui sont :

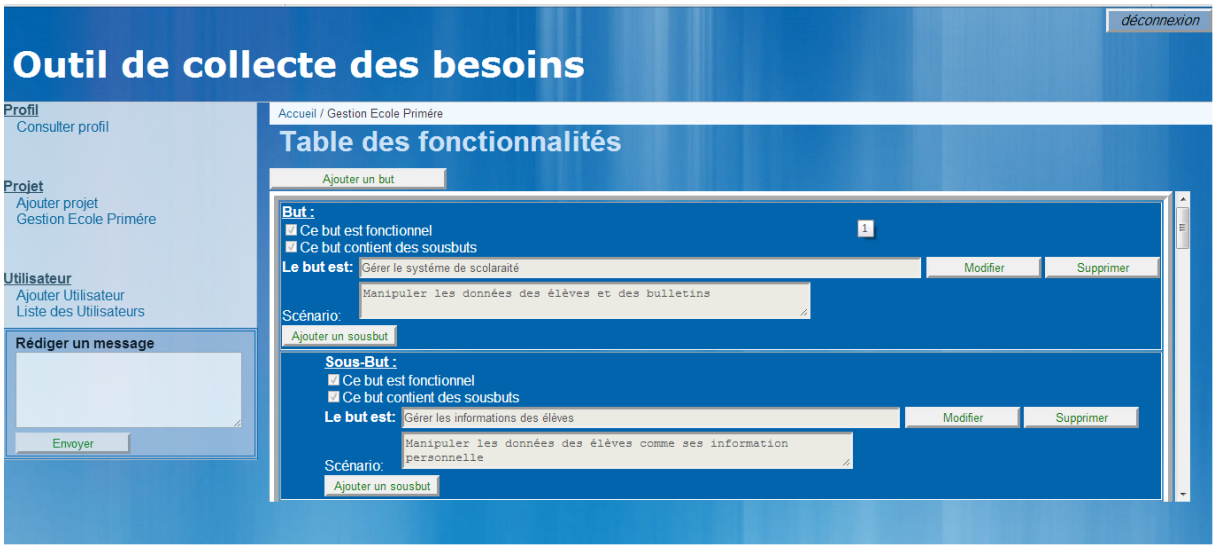

FIGURE 3.9 – Table des fonctionnalités d'un projet

- Titre du but.
- Scenario du but.
- Ce but est-il fonctionnel ou non.
- Ce but contient des sous buts ou non.
- Un lien vers la maquette associée à ce but à condition que ce but soit fonctionnel et il ne contient pas des sous buts. Cette maquette permet à l'ingénieur de consulter et colleter les informations mis par l'utilisateur (attributs, opérations, événements), elle est composée de deux parties :
  - ▷ La première partie contient tous les éléments que l'utilisateur aura besoin pour exprimer ces intentions, plus les propriétés associées à ces éléments ou bien à la maquette.
  - ▷ La deuxième partie concerne le contenu de la maquette c'est-à-dire le besoin de l'utilisateur.

La figure suivante représente la page maquette :

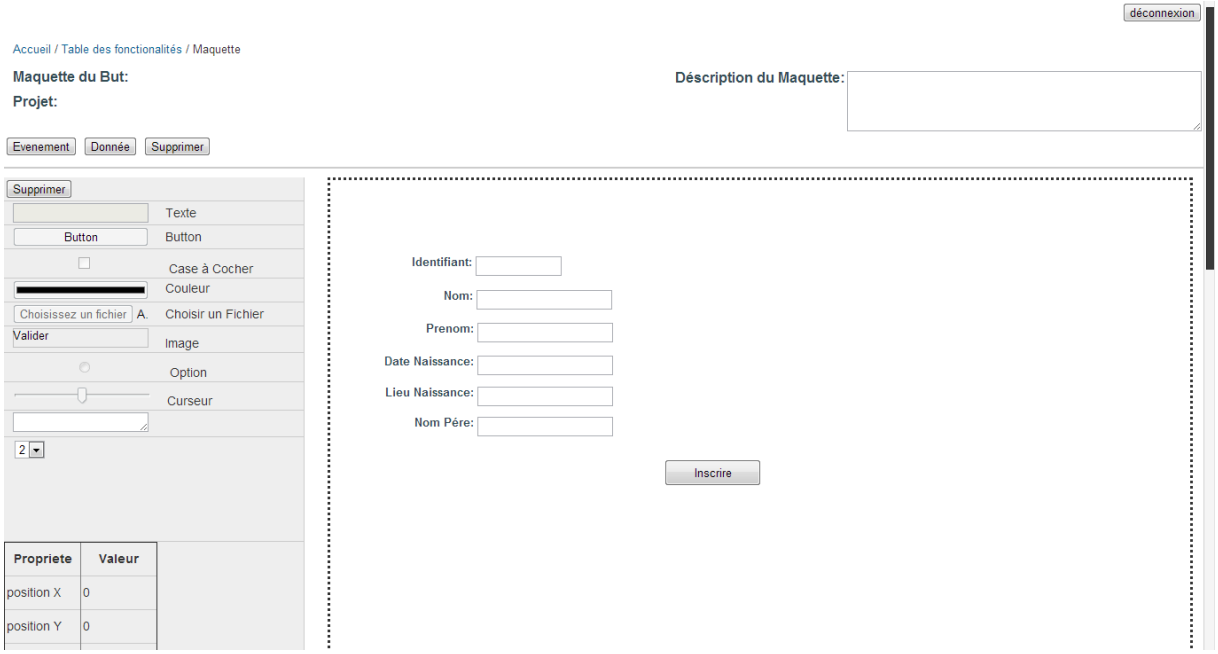

FIGURE 3.10 – La page maquette

- En dessus de la page, on retrouve avec un champ où l'ingénieur peut envoyer un message vers l'utilisateur.
- Un lien vers la page ajouter utilisateur afin d'ajouter un utilisateur.
- Un lien vers la page consulter utilisateur afin de consulter, modifier ou supprimer un utilisateur.

3.3.3 Partie « Utilisateur »

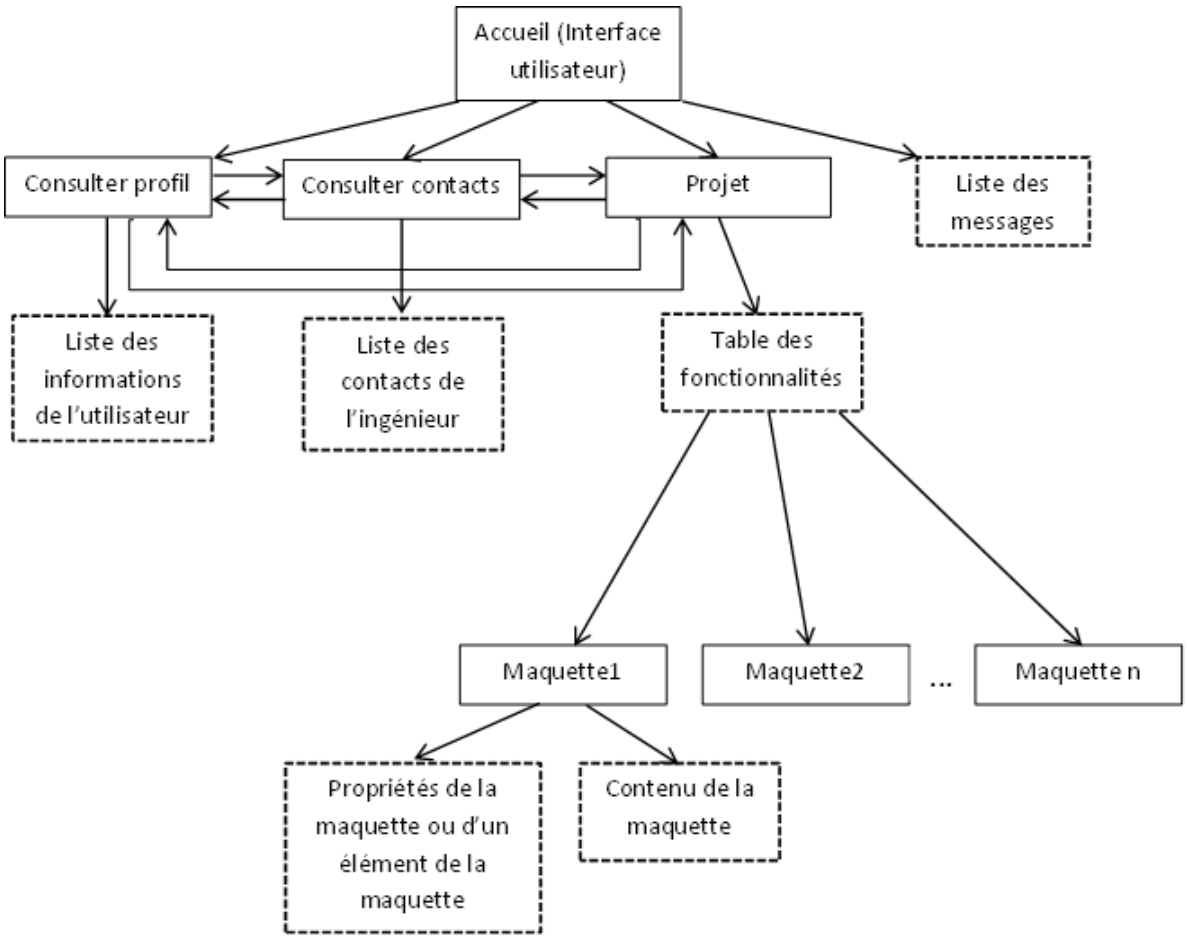

FIGURE 3.11 – Le schéma de la partie « Utilisateur »

Explication :

Après l'authentification de l'utilisateur, il se retrouve dans la page d'accueil d'utilisateur(voir la figure 3.12) où il peut :

- Consulter les messages envoyés par l'ingénieur.
- Consulter et modifier son profil.
- Se déplacer vers la page projet.
- Consulter les contacts de l'ingénieur et l'envoyer un e-mail.

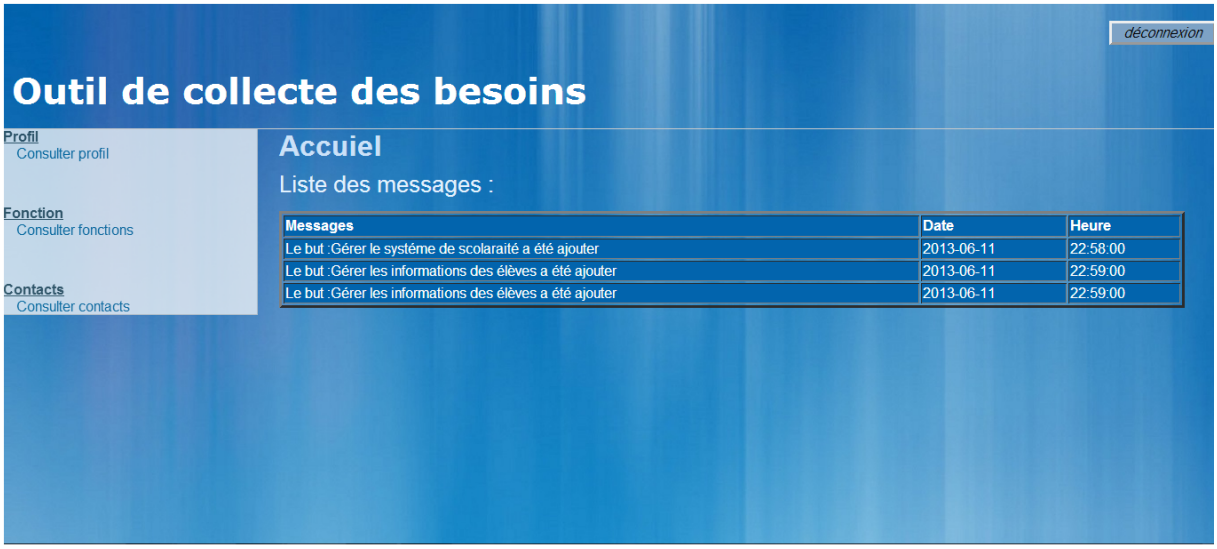

FIGURE 3.12 – La page d'accueil de l'utilisateur

Dans la page « projet »l'utilisateur peut exprimer tous ses besoins à partir du table de fonctionnalité et la page « maquette » ;

Remarque : La table de fonctionnalités et la page « maquette »sont les mêmes pour l'ingénieur et l'utilisateur sauf que :

- L'utilisateur les utilise pour exprimer ses intentions.
- L'ingénieur les utilise pour collecter ses intentions.

3.4 Conclusion

Jusque-là, on a terminé avec la partie implémentation qui est aussi une partie non négligeable dans notre travail. On a implanté la base de données, programmer les pages web pour qu'elle support notre démarche.

Conclusion

Enfin, et après tout ce qu'on a vu dans les quatre chapitres précédents, on peut conclure que le processus de l'IB est une étape importante dans le développement de n'importe quel système, car une simple erreur dans cette étape va conduire vers une conception fausse qui à son tour va donner une mauvaise implémentation (logiciel, application, site web...) et qu'il ne va pas répondre aux besoins des utilisateurs ce qu'il va influencer négativement sur la qualité du système étudié et sur son rendement. C'est pourquoi le processus de l'IB est incontournable dans le développement d'un système.

Sous l'effet de trouver des solutions au problème de l'IB, il existe des méthodes mais elles ne sont pas encore standardisées.

Dans une perspective d'enrichir ce créneau de recherche nous avons proposé une méthode d'IB basés sur les buts et maquettes.

Cependant et comme tout travail de recherche plusieurs points peuvent être encore améliorés :

- Améliorer l'interface de notre outil afin qu'il soit plus compréhensible et plus flexible.
- Réfléchir sur un format de sortie pour notre outil.
- Arranger l'outil pour qu'il supporte plus de format de données (texte, date,...).
- Ajouter le taux d'accomplissement à chaque maquette.
- Protéger et améliorer la sécurité de notre outil.
- Rendre notre outil multi-utilisateurs et multi-ingénieurs pour un projet spécifié.
- Rendre notre outil accessible pour tous les navigateurs.
- Adapter notre outil pour qu'il soit supporté par les tablettes.
- Définir une méthode de discussion avec l'utilisateur pour guider la discussion.
- Définir les déférents types de question à poser dans le première rencontre.

Ces différents points représentent des perspectives qui peuvent faire l'objet d'une autre étude.

Bibliographie

[1] Hakim BENDJENNA. Ingénierie des exigences pour les processus inter-organisationnels. 2010.

[2] Colette Rolland. L'ingenierie des besoins : L'approche l'ecritoire. 2012.

[3] Sai Ganesh. Requirements engineering : Elicitation techniques. 2008.

[4] PICARD Fabienne. Elaboration d'une stratégie régionale d'innovation : les apports d'une approche par la méthode d'enquête kano. 2010.

[5] Mustapha Tawbi. Crews-l'ecritoire : Une approche guidant l'ingénierie des besoins.

[6] Martin Glinz. Improving the quality of requirements with scenarios. September 2000.

[7] Ismaïl KHRISS. Vers un paradigme transformationnel dans le développement orienté objet. 2000.

[8] Alistair Sutcliffe. Scenario-based requirement analysis. 1988.

[9] Alistair Sutcliffe. Scenario-based requirements engineering. 2003.

[10] Colette Rolland. Une methode de conception orientee objet & evenement.

[11] Mohamed Tarek El Allam Mohamed Yassir Bouhaddaoui. Etude comparative sur les méthodes de modélisation. 2011.

[12] Christian SOUTOU. *UML2 Pour les Base de Donnée*. 2002.

[13] Xavier Blanc. Mda en action. 2005.

[14] Jean Charlet Pierre-Yves Vandenbussche. Méta-modèle général de description de ressources terminologiques et ontologiques. 2009.

[15] Charlotte HUG. Méthode, modèles et outil pour la méta-modélisation des processus d'ingénierie de systèmes d'information. 2009.

[16] Luc VAN LANCKER. Html 4 maîtrisez le code source. 2008.

[17] Luc VAN LANCKER. Des css au dhtml java script appliqué aux feuilles de style. 2006.

[18] Fabien Duchateau. Lif4 programmationweb cours css. 2012.

[19] Emmanuel GUTIERREZ. Javascript des fondamentaux aux concepts avancés. 2008.

[20] Site de zéro. www.siteduzero.com/informatique/tutoriels/dynamisez-vos-sites-web-avec-javascript/qu-est-ce-que-le-javascript.

[21] Richard Grin. Langage sql. 2008.

[22] Egon Schmid Jim Winstead Lars Torben Wilson Rasmus Lerdorf Zeev Suraski Andrei Zmievski Jouni Ahto Damien Seguy Stig Saether Bakken, Alexander Aulbach. Manuel php. 2001.

[23] Jean-Marie Defrance. Formation elephorm « apprendre php et mysql ».

[24] Jean-Marie Defrance. Premières applications web2.0 avec ajax et php. 2008.

[25] Birnou Sébarte. Formation videobrain les fondamentaux d'ajax par la pratique.