Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Dédicace

Nous dédions ce présent mémoire

A

Nos parents

Pour les sacrifices déployés à nos égards ; pour leur patience leur amour et leur confiance en nous. Ils ont tout fait pour notre bonheur et notre réussite. Qu'ils trouvent dans ce modeste travail, le témoignage de notre Profonde affection et de notre attachement indéfectible. Nulle dédicace ne puisse exprimer ce que nous leur devons.

Que Dieu leur réserve la bonne santé et une longue vie.

A

Nos amis

En témoignage de nos sincères reconnaissances pour les efforts Qu'ils ont consentis pour nous soutenir au cours de nos études. Que Dieu nous garde toujours unis.

A

Toutes les personnes qui ont contribué de près ou de loin à la réalisation de présent projet.

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Remerciements

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes à qui nous voudrons témoigner toute notre reconnaissance.

En préambule à ce mémoire nous remercions ALLAH qui nous aide et nous donne la santé, la patience ainsi que le courage durant ces longues années d’étude.

Nous souhaitons adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l’élaboration de ce mémoire ainsi qu’à la réussite de cette formidable année universitaire.

Ces remerciements vont également à l’égard du corps professoral et administratif de notre école IAI-Niger, pour la richesse et la qualité de leur enseignement et qui déploient de grands efforts pour assurer à leurs étudiants une formation actualisée.

Nous tenons à remercier sincèrement Monsieur Chaibou Abdou, qui, en tant que Directeur de mémoire, s’est toujours montré à l’écoute et très disponible tout au long de la réalisation de ce mémoire, ainsi pour l’inspiration, l’aide et le temps qu’il a bien voulu nous consacrer et sans qui, ce mémoire n’aurait jamais vu le jour.

On n’oublie pas nos très chers parents qui ont toujours été là pour nous, pour leur contribution, leur soutien et leur patience. Vous nous avez donné un magnifique modèle de labeur et de persévérance. Nous sommes redevables d’une éducation dont on est fier.

Enfin, nous adressons nos plus sincères remerciements à tous nos proches et amis, qui nous ont toujours encouragé.

Merci à tous et à toutes.

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

# Sigles et Abréviations

TIC : Technologies d’Information et la Communication

NTIC : Nouvelles Technologies d’Information et de la Communication

UML : Unified Modeling Language

UP: Unified Process (Processus Unifié)

T2UP: Two Tracks Unified Process

PHP: Hypertext Preprocessor

HTML : Hyper Text Markup Language

CSS : Cascading Style Sheets (feuilles de style en cascade)

JS : JavaScript

POO : Programmation Orienté Objet

RSM : Responsable Sélection Médical

AS : Assistant Médical

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

# Table des figures

[Figure 1:Organigramme de SYNETCOM 3](#_Toc526901324)

[Figure 2 : Le système d’information soumis à deux types de contraintes 8](#_Toc526901325)

[Figure 3 : Le processus de développement en Y 9](#_Toc526901326)

[Figure 4 : Planning du projet 11](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901327)

[Figure 5 : Diagramme de contexte statique 17](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901328)

[Figure 6 : Diagramme de packages 28](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901329)

[Figure 7 : Diagramme de cas d’utilisation, package « gestion de la bibliothèque » 29](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901330)

[Figure 8 : Diagramme de cas d’utilisation, package « gestion administrative » 29](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901331)

[Figure 9: Diagramme de cas d’utilisation, package « gestion des enseignements » 30](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901332)

[Figure 10 : Diagramme de cas d’utilisation, package « gestion des promotions » 31](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901333)

[Figure 11 : Diagramme de cas d’utilisation, package « gestion des notes » 31](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901334)

[Figure 12 : Diagramme de cas d’utilisation, package «Gestion des absences/emplois du temps» 32](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901335)

[Figure 13 : Diagramme de cas d’utilisation, package « Gestion des droits d’accès » 33](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901336)

[Figure 14 : Cas d’utilisation gérer les filières, package « Gestion des enseignements » 35](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901337)

[Figure 15:Diagramme d’activité gérer les filières, package « Gestion des enseignements » 38](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901338)

[Figure 16: Diagramme de séquence gérer les filières, package « Gestion des enseignements » 39](#_Toc526901339)

[Figure 17 : cas d’utilisation : Gérer les classes, package « Gestion des enseignements » 40](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901340)

[Figure 18 : Diagramme d’activité gérer les classes, package « Gestion des enseignements » 43](#_Toc526901341)

[Figure 19 : Diagramme de séquence gérer les classes, package « Gestion des enseignements » 44](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901342)

[Figure 20 : Cas d’utilisation : gérer les années scolaires, package « Gestion des promotions » 45](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901343)

[Figure 21 : Diagramme de séquence gérer les années scolaires, package « Gestion des promotions» 47](#_Toc526901344)

[Figure 22 : Cas d’utilisation maintenir les notes, package « Gestion des notes » 48](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901345)

[Figure 23 : Diagramme de séquence maintenir les notes, package « Gestion des notes» 49](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901346)

[Figure 24 : Diagramme d’activité gérer les étudiants, package « Gestion Administrative » 52](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901347)

[Figure 25 : Diagramme de classe participante des cas d’utilisation, package « gestion des enseignements » 53](file:///C:\Users\hp\Downloads\Memoire%203aap%20Final.docx#_Toc526901348)

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Table des tableaux

[Table 1 : Exemple de calcul de la moyenne d’une unité d’enseignement 13](#_Toc526903045)

[Table 2 : Exemple de calcul de la moyenne d'un élément. 14](#_Toc526903046)

[Table 3 : Description des besoins fonctionnels 20](#_Toc526903047)

[Table 4 :Liste des cas d’utilisation 25](#_Toc526903048)

[Table 5 : Liste des cas d’utilisation par package 27](#_Toc526903049)

[Table 6 : Formalisme de description d’un cas d’utilisation 34](#_Toc526903050)

[Table 7 : Description détaillé, cas d’utilisation gérer les filières, package « Gestion des enseignements » 37](#_Toc526903051)

[Table 8 : Description détaillé, cas d’utilisation gérer les classes, package « Gestion des enseignements » 42](#_Toc526903052)

[Table 9 : Description détaillé, cas d’utilisation gérer les années scolaires, package « Gestion des promotions » 46](#_Toc526903053)

[Table 10 : Description détaillée, cas d’utilisation maintenir les notes, package « Gestion des notes » 49](#_Toc526903054)

[Table 11 : Description détaillée, cas d’utilisation :gérer les étudiants , package « Gestion administrative » 51](#_Toc526903055)

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Table des matières

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Introduction

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

La révolution des technologies de l’information et de la communication (TIC), a engendré au cours de ces dernières années une progression notable de la création et de l’usage des applications de gestion dans la vie courante et professionnelle. Dans ce contexte de transformation numérique, et afin d’aider les centres de transfusions sanguines à mettre en place un système de gestion automatique*,* SYNETCOM a initié un projet de développement d’une plateforme web standard qui répondra aux attentes des gestionnaires des centres de transfusions sanguines objet de la présente étude.

La présente étude qui nous a été soumise dans le cadre de notre projet de fin d’étude pour l’obtention du diplôme d’ingénieurs des travaux informatiques, vient pour répondre aux besoins d’un secteur de base intéressant et très sensible, à savoir la santé.

En effet, la santé fait partie des domaines qui sont à la traine dans l’adoption des outils TIC alors qu’il en a plus besoin compte tenu du volume d’information géré tant au niveau des enseignements qu’au niveau des enseignés mais aussi au niveau des différents acteurs impliqués.

Pour concevoir et mettre en place un système numérique répondant à notre thème « Conception d’une plateforme web de gestion d’une école supérieure », et qui réunit les différentes fonctionnalités nécessaires à la gestion des établissements supérieurs, prenant en compte la reforme LMD (Licence-Master-Doctorat) d’une école supérieure, le présent mémoire s’articulera autour de six chapitres :

* Premier chapitre intitulé : Contexte général du projet
* Deuxième chapitre : Démarche du projet
* Troisième chapitre : Etude préliminaire
* Quatrième chapitre : Capture des besoins fonctionnels
* Cinquième chapitre : L’Analyse
* Sixième chapitre : Capture des besoins techniques
* Septième chapitre : Réalisation

Mémoire de fin de cycle : Analyste Programmeur 2018-2019



# Chapitre I : Présentation du centre d’accueil

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

## Présentation de SYNETCOM

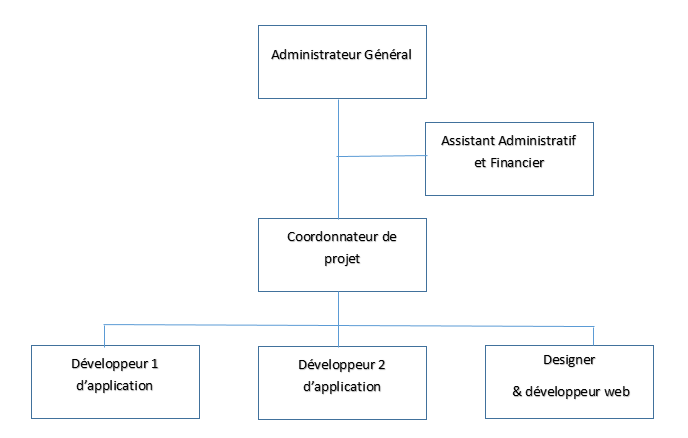
SYNETCOM est une jeune entreprise spécialisée dans les domaines des nouvelles technologies de l’information et de la communication.

Vecteur d'innovation, levier d'efficacité, accélérateur de développement, grâce à son expérience et à son anticipation permanente sur les besoins des clients, SYNETCOM agit en véritable partenaire technologique pour lever les défis et les difficultés auxquels heurtent ses partenaires et optimiser les performances de leur système de gestion.

Sa vocation est d’apporter les ressources nécessaires pour faire bénéficier au mieux à ses partenaires les atouts des nouvelles technologies

SYNETCOM réfléchit aux problèmes du marché et conçoit les solutions novatrices et les services d'accompagnement adaptés aux besoins des clients.

## Organigramme



# Chapitre II : Présentation du projet

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

## Contexte du projet

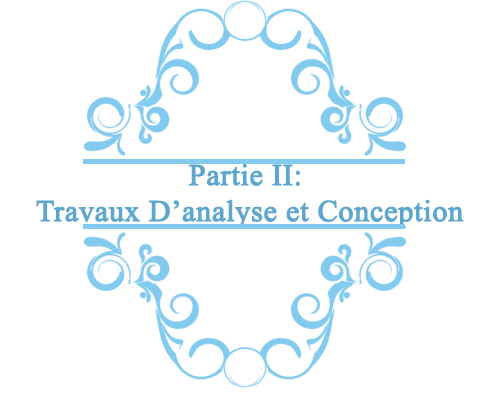
## Description de l’existant

## Problématique soulevée

## Travail attendu

## Planning prévisionnel

Mémoire de fin de cycle : Analyste Programmeur 2018-2019



# Chapitre III : Analyse

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

## Méthode d’analyse

Une méthodologie de développement est un cadre utilisé pour structurer, planifier et contrôler le développement d’une application. En effets, nous disposons de nos jours d’un ensemble de méthodologie chacune respectant un certain nombre de principe pour aboutir à la réalisation d’un projet. Ainsi nous allons parler principalement de quatre types de méthodes de conception.

## Pourquoi utiliser une méthode d’analyse

On appelle « gestion de projet » éventuellement « conduite de projet » l’organisation méthodologique mise en œuvre pour faire en sorte que l’ouvrage réalisé par le maitre d’œuvre réponds aux attentes du maitre d’ouvrage et qu’il soit livré dans les conditions de coût et de délais prévus initialement.

En effet, les projets sont gérés avec la méthode dite « classique » qui se caractérise pour recueillir les besoins, définir le produit, le développer et le tester avant de le livrer. On parle alors ici d’une approche prédictive. Comme son nom l’indique, il s’agit ici de prévoir des phases séquentielles où il faut valider l’étape précédente pour passer à la suivante.

Le chef de projet doit alors  
s’engager sur un planning précis de réalisation du projet en prévoyant des  
jalons de débuts et fins de phases ainsi que les tâches à effectuer. Il faut tout  
faire bien du premier coup car elle ne peut pas permettre de retours en arrière.  
Une décision ou un problème rencontré dans une phase peuvent remettre en  
cause partiellement ou totalement les phases précédentes validées. Dans un  
cycle « en cascade » les risques sont détectés tardivement puisqu’il faut  
attendre la fin du développement pour effectuer la phase de test. Plus le projet  
avance, plus l’impact des risques augmente : il sera toujours plus difficile et  
coûteux de revenir en arrière lorsqu’on découvre une anomalie tardivement.  
Par conséquent, comment peut-on augmenter la satisfaction du client en  
facilitant la gestion de projet et améliorant la qualité de développement ? Pour  
se faire, nous allons voir d’autres méthodes modernes plus adéquates.

## Classification des méthodologies

## Méthodes fonctionnelles

Les méthodes fonctionnelles ou cartésiennes sont des démarches consistant à rechercher et à caractériser les fonctions offertes par un produit pour faire les besoins de son utilisateur. Il est aussi défini comme étant une méthode permettant à décomposer hiérarchiquement une application en un ensemble de sous applications. Parmi les fonctions de chacune de celles-ci sont affinées alternativement en sous fonctions simples à coder dans un langage de programmation donné. L’une de ces méthodes, on peut noter FAST (Fonction Analyse System Technique), SADT (Structurel-Analyse-Design-Technique) Warnier …

Forces de ces méthodes :

* Simplicité du processus de conception préconisée ;
* Adéquations à capturer les besoins de l’utilisateur ;
* Capacités de produire des solutions à plusieurs niveaux d’abstraction.

Faiblesses de ces méthodes

* Effort d’analyse concentré sur les fonctions (négligeant la cohérence des données) ;

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

* Règles de décomposition non explicites ;
* Difficulté à tenir compte des interactions non hiérarchiques dans les systèmes complexes.

## Méthodes classiques

Souvent spécialisées pour la conception d’un certain type de systèmes, les méthodes systémiques définissent différents niveaux de préoccupation ou d’abstraction et proposent de nombreux modèles complémentaires. Autrement dit ces méthodes proposent une double démarche de modélisation, celles des données et des traitements. Elles sont influencées par les systèmes de gestion de bases de données SGBD. On a comme exemple MERISE, AXIAL RACINE...

Forces de ces méthodes :

* Les concepts sont peu nombreux et simples
* Elle est assez indépendante vis-à-vis de la technologie,
* Bonne adaptation à la modélisation des données et à la conception des bases de données,

Faiblesses de ces méthodes

* Double démarche de conception celles des données et des traitements,
* Impossible de fusionner les deux aspects (données et traitements)

## Méthodes unifiées :

Les méthodes unifiées sont des méthodes de développement pour les logiciels orientés objets. Plus exactement, ce sont les meilleures pratiques du  
développement objet suivies pour la réalisation d’un système. Donc le Processus Unifié (UP) est un processus de développement logiciel « Itératif et incrémental, centré sur l’architecture, conduit par les cas d’utilisation et piloté par les risques » :

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

## 2.3.1. Méthodes UP :

Le Processus Unifié (PU ou UP en anglais pour Unified Process) est une méthode de développement logiciel construite sur UML ; elle est itérative et incrémentale, centrée sur l’architecture, conduite par les cas d’utilisation et pilotée par les risques.

Itérative et incrémentale : la méthode est itérative dans le sens où elle propose de faire des itérations lors de ses différentes phases, ceci garantit que le modèle construit à chaque phase ou étape soit

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

* Affiné et amélioré. Chaque itération peut servir aussi à ajouter de nouveaux incréments.
* Conduite par les cas d’utilisation : elle est orientée « utilisateur » pour répondre aux besoins de celui-ci.
* Centrée sur l’architecture : les modèles définit tout au long du processus de développement vont contribuer à établir une architecture cohérente et solide.

### Pilotée par les risques : en définissant des priorités pour chaque fonctionnalité, on peut minimiser les risques d’échec du projet.

## Méthode 2TUP :

**2TUP** est un processus de développement logiciel qui implémente le Processus Unifié. Le **2TUP** propose un cycle de développement en Y, qui dissocie les aspects techniques des aspects fonctionnels. Il commence par une étude préliminaire qui consiste essentiellement à :

* Identifier les acteurs qui vont interagir avec le système à construire ;
* Identifier les messages qu'échangent les acteurs et le système ;
* Produire le cahier des charges ;
* Modéliser le contexte (le système est une boîte noire, les acteurs l'entourent et sont reliés à lui, sur l'axe qui lie un acteur au système on met les messages que les deux s'échangent avec le sens).

Le processus s'articule ensuite autour de 3 phases essentielles :

**

Figure 1 : Le système d’information soumis à deux types de contraintes

**La branche gauche** (fonctionnelle) comporte :

* La capture des besoins fonctionnels, qui produit un modèle des besoins focalisé sur le métier des utilisateurs. Elle qualifie au plus tôt le risque de produire un système inadapté aux utilisateurs. De son côté, la maîtrise d’œuvre consolide les spécifications et en vérifie la cohérence et l'exhaustivité.
* L'analyse, qui consiste à étudier précisément la spécification fonctionnelle de manière à obtenir une idée de ce que va réaliser le système en termes de métier. Les résultats de l'analyse ne dépendent d'aucune technologie particulière.

### **La branche droite** (architecture technique} comporte :

* La capture des besoins techniques, qui recense toutes les contraintes et les choix dimensionnant la conception du système. Les outils et les matériels sélectionnés ainsi que la prise en compte de contraintes d'intégration avec l'existant conditionnent généralement des prés requis d'architecture technique.

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

* La conception générique, qui définit ensuite les composants nécessaires à la construction de l'architecture technique. Cette conception est complètement indépendante des aspects fonctionnels. Elle a pour objectif d'uniformiser et de réutiliser les mêmes mécanismes pour tout un système. L'architecture technique

Elle construit le squelette du système informatique et écarte la plupart des risques de niveau technique. L'importance de sa réussite est telle qu'il est conseillé de réaliser un prototype pour assurer sa validité.

**La branche du milieu** comporte :

* **La conception préliminaire**, qui représente une étape délicate, car elle intègre le modèle d'analyse dans l'architecture technique de manière à tracer la cartographie des composants du système à développer, la conception détaillée, qui étudie ensuite comment réaliser chaque composant ; l'étape de codage, qui produit ces composants et teste au fur et à mesure les unités de code réalisées
* **La conception détaillée**, qui étudie ensuite comment réaliser chaque composant ;
* **Le codage et tests**, qui produisent ces composants et teste au fur et à mesure les unités de code réalisées ;
* **L’étape de recette**, qui consiste enfin à valider les fonctions du système développé.

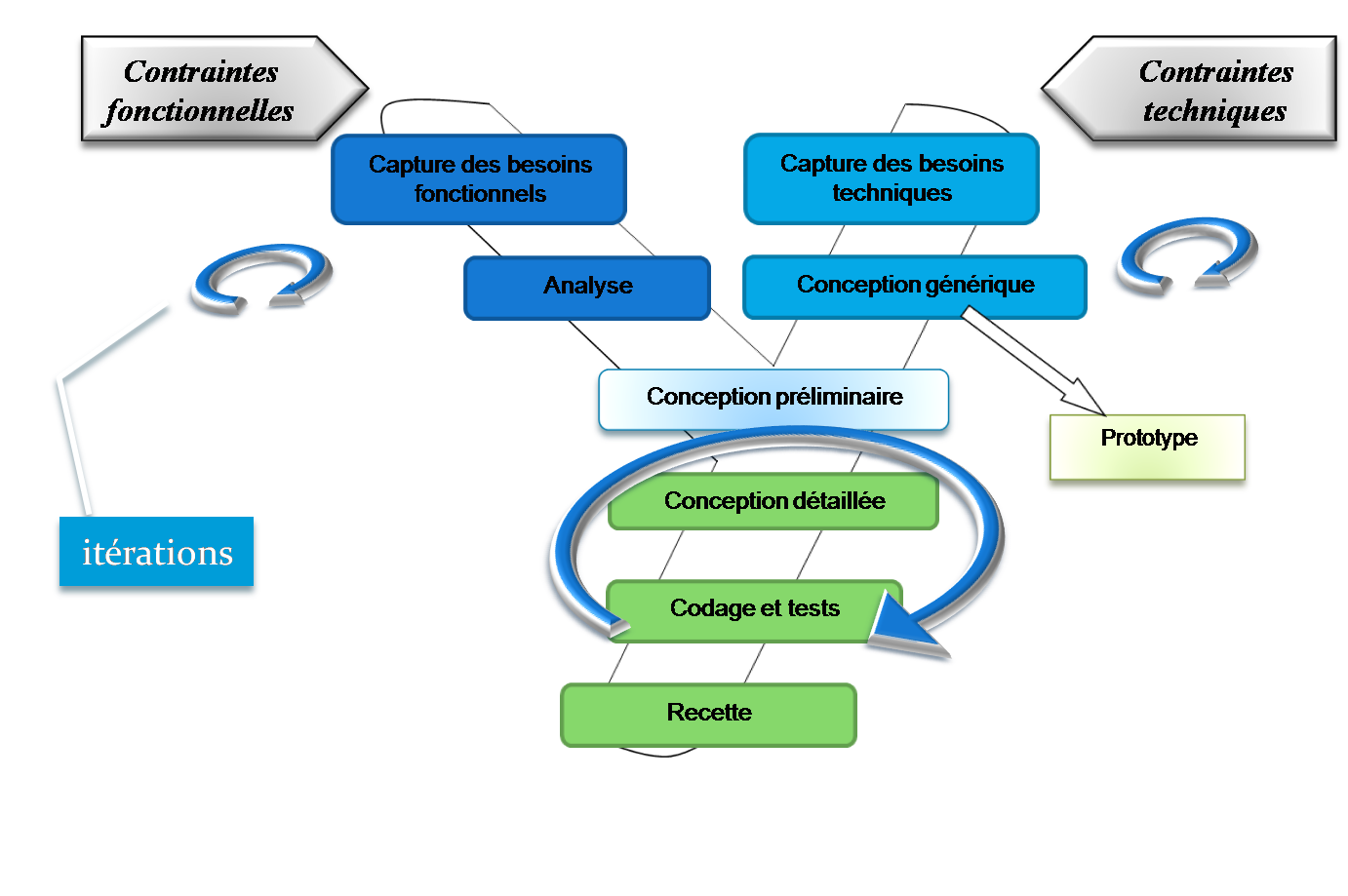


Figure 2 : Le processus de développement en Y

## Méthodes agiles :

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Ces méthodes constituent un gain en productivité ainsi qu'un avantage  
compétitif tant du côté client que du côté du fournisseur. En effet, ces méthodes de développement dites « méthodes agiles » visent à réduire le cycle de vie du logiciel (donc accélérer son développement) en développant une version minimale, puis en intégrant les fonctionnalités par un processus itératif basé sur une écoute client et des tests tout au long du cycle de développement. Parmi ces méthodes nous pouvons citer : Scrum, XP (eXtreme Programming),   
RAD (Rapid Application Development), etc.…

## 2.4.1. XP :

Extrême Programming, ou XP, est une méthode agile de gestion de projet particulièrement bien adaptée aux projets de développement informatique. Elle a été conçue par Kent Beck pour accélérer les développements alors qu’il travaillait pour la société Chrysler. L’idée lui est venue alors qu’il devrait intervenir sur un logiciel de paie écrit en langage « Small talk » ayant accumulé une dette technique considérable, le rendant particulièrement complexe à maintenir et à faire évoluer.

Le principe fondamental de la méthode XP est de faire collaborer étroitement tous les acteurs du projet et d’opter pour des itérations de développement très courtes. La planification des tâches reste très souple et l’estimation des charges simplifiée par des projections à très court terme. Ainsi la correspondance entre ce qu’attendent le client et les réalisations est garantie. Les fonctionnalités sont livrées régulièrement, afin d’être testées et validée au travers de prototypes opérationnels. L’Extrême Programming préconise également le travail en binôme des développeurs, facilitant ainsi la production d’un code simple, facilement lisible et maintenable.

## 2.4.2. Scrum :

Scrum est considéré comme un cadre ou « Framework » de gestion de projet.  
Ce cadre est constitué d'une définition des rôles, de réunions et d'artefacts. Le mot Scrum est un terme emprunté au rugby qui signifie « mêlée » encore appelée « Morning **»**. Dans cette méthode, on focalise l'équipe de façon itérative sur un ensemble de fonctionnalités. En effet, Scrum définit 3 rôles :

* Le propriétaire du produit « Product Owner » qui porte la vision du produit à réaliser (représentant généralement le client) ;
* Le directeur du produit « Scrum Master » garant de l'application de la  
  méthodologie Scrum ;
* L'équipe de développement qui réalise le produit.

La vie d'un projet Scrum est rythmée par un ensemble de réunions clairement  
définies et strictement limitées dans le temps (timeboxing):Planification du Sprint (Sprint = itération) : au cours de cette réunion, l'équipe de développement sélectionne les éléments prioritaires du « Product Backlog » (liste ordonnancée des exigences fonctionnelles et non fonctionnelles du projet) qu'elle pense pouvoir réaliser au cours du sprint (en accord avec le « Product Owner »).Revue de Sprint : au cours de cette réunion qui a lieu à la fin du sprint, l'équipe de développement présente les fonctionnalités terminées au cours du sprint et recueille les feedbacks du Product Owner et des utilisateurs finaux. C'est également le moment d'anticiper le périmètre des prochains sprints et d'ajuster au besoin la planification de release (nombre de sprints restants).  
Rétrospective de Sprint : la rétrospective qui a généralement lieu après la revue de sprint est l'occasion de s'améliorer (productivité, qualité, efficacité, conditions de travail, etc.) à la lueur du "vécu" sur le sprint écoulé (principe d'amélioration continue).Mêlée quotidienne : il s'agit d'une réunion de synchronisation de l'équipe de développement qui se fait debout (elle est aussi appelée "stand up meeting") en 15 minutes maximum au cours de laquelle chacun répond principalement à 3 questions :« Qu'est-ce que j'ai terminé depuis la dernière mêlée ? Qu'est-ce que j'aurai terminé d'ici la prochaine mêlée ? Quels obstacles me retardent ? »

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

C:\Users\MINA\Desktop\momo IGS\Untitled-Diagram.png

Figure 3 : La méthode scrum

## 3. Choix de la méthodologie :

## 3.1. Méthodologie de conception :

Après l’étude des différentes méthodologies, nous avons opté pour Scrum et UP pour les raisons qui suivent :

* Scrum convient aux équipes ayant un nombre de développeurs réduits. Ceci est le cas de notre projet ;
* La progression des tâches s’effectue pendant une durée de développement courte ;
* UP s’applique à tous les types de projet informatique pour les suggestions et remarques ;
* UP est une méthode itérative et incrémental ;
* UP est essentiellement basée sur la satisfaction des besoin set des exigences du clients et des utilisateurs ;
* UP se base sur le diagramme des cas d’utilisation qui illustre au mieux les besoins fonctionnels et opérationnels auxquels doivent répondre le produit final.

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

## 3.2. Présentation du langage UML :

Les auteurs des méthodes (OOSE de Jacobson, OMT de Rumbaugh, BOOCH de Grady Booch) se sont fixé des objectifs dont entre autres la représentation des systèmes entiers par des concepts objet ; l’établissement d’un couplage explicite entre les concepts et les interfaces exécutables et la création d’un langage de modélisation utilisable à la fois par les humains et les machines et adaptés aux systèmes simples et complexes. D’où la naissance d’UML qui est donc une norme du langage de modélisation objet qui a été publiée, dans sa première version en septembre 1997 par l’OMG (Object Management Group), instance de normalisation internationale du domaine de l’objet. UML est un langage de modélisation graphique et textuel destiné à comprendre et à décrire des besoins, spécifier et documenter des systèmes, esquisser architectures logicielles, concevoir des solutions et communiquer des points des vues.  De plus UML comble une lacune importante des technologies objet, il permet d’exprimer, d’élaborer et de modéliser au sens de la théorie des langages. De ce fait il contient les éléments constitutifs de ces derniers : concepts, une syntaxe et une sémantique.

## 3.2.1. Les diagramme UML :

UML définit 13 types de diagrammes divisés en deux catégories : les diagrammes structurels ou statiques rassemblent :

* Diagramme de classe (Class Diagram) ;
* Diagramme d’Objet (Object Diagram) ;
* Diagramme de composant (Composent Diagram) ;
* Diagramme de déploiement (Deployment Diagram) ;
* Diagramme de paquetage (Package Diagram)
* Diagramme de structure composite (Composite Structure Diagram) .

Les diagrammes comportementaux qui rassemblent :

* Diagramme des cas d’utilisation (Use Case Diagram).
* Diagramme d’état-transitions (State Machine Diagram) ;
* Diagramme d’activité (Activity Diagram) ;
* Diagramme de séquence (Sequence Diagram) ;
* Diagramme de communication (Communication Diagram) ;
* Diagramme global d’interaction (Interaction Overview Diagram) ;
* Diagramme de temps (Timing Diagram) ;

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

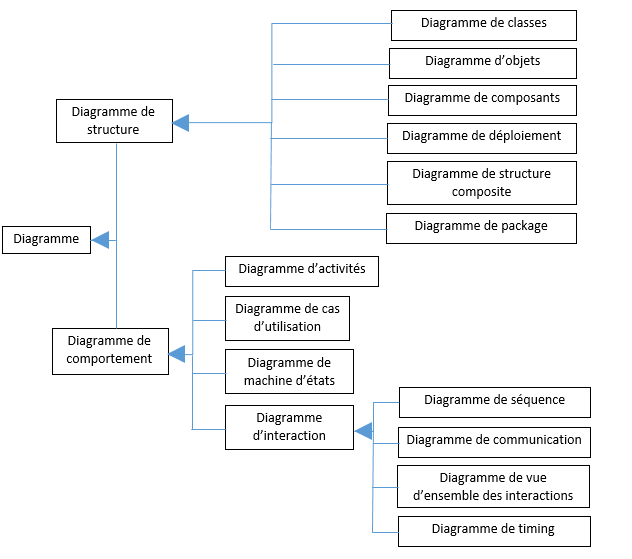


Figure 4 : Les diagrammes UML

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

## Identification des besoins

## Les besoins fonctionnels

Au cours de cette étape, nous allons définir les différentes fonctionnalités qu’offrira notre application :

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

1. Les besoins non fonctionnels

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019

Mémoire de fin de cycle : Analyste Programmeur 2018-2019