Septembre 2015

Systèmes d'information, Méthodes avancées

- Support de cours en "Systèmes d'information, méthodes avancées"
- Cours dispensé aux classes de master 1, option "systèmes d'informations et aide à la décision", Départment d'informatique, Faculté des sciences exactes et informatique, Université de Jijel.

Elaboré par: Dr. Doulkifli BOUKRAA
Maitre de conférences B, département d'informatique, Faculté des sciences exactes et informatique, Université de Jijel

Avant-propos

Ce support de cours est destiné aux étudiants en première année du deuxième cycle LMD, spécialité *systèmes d'informations et aide à la décision* (S.I.A.D.) du département d'informatique, faculté des sciences exactes et informatique, université de Jijel. Il concerne le module « systèmes d'informations, méthodes avancées ». Ce document peut également être utilisé par les étudiants en deuxième année master dans le cadre de leurs projets de fin d'études. Il représente ainsi un guide retraçant les étapes à suivre tout au long du projet. Les étudiants de spécialités similaires peuvent également trouver ce support utile pour élargir leurs connaissances en matière de développement des systèmes d'information.

Ce support de cours contient en premier lieu l'ensemble des chapitres du cours, organisés selon les étapes d'un processus de développement des systèmes d'information. Trois parties annexes s'en suivent qui comportant les séries d'exercices, les interrogations et les examens depuis l'année universitaire 2011-2012 jusqu'à l'année 2014-2015. Pour certaines séries d'exercices, des corrigés-types ou indications sont donnés.

Ce document est, comme son nom l'indique, ne sert que de support et ne peut pas se substituer au cours présentiel. Aussi, les séries d'exercices, les interrogations et les examens peuvent être utilisées pour consolider la compréhension du cours et pour bien préparer les futurs examens ou concours.

Ce cours détaille le processus Two-Track Unified Process (2TUP) lequel est basé sur le processus unifié (UP) du développement des systèmes d'informations. Les références principales de ce cours sont les deux livres *UML en action* et *UML 2 en action* de Pascal Roques. Encore une fois, ces deux livres étant amplement détaillés, ne peuvent pas directement remplacer ce cours vu le degré avancé de détail des livres, devant lequel l'étudiant moyen risque de perdre les repères. Ainsi, ce support peut être vu comme un résumé fournissant une vue synthétique et se concentrant sur l'essentiel.

Ce document, par vocation, se veut didactique. Tout au long des chapitres, un seul exemple de la *gestion du personnel d'une entreprise X* est développé afin de permettre à l'étudiant de mieux comprendre les concepts de chaque étape du processus de développement à travers un cas pratique.

Enfin, ce support de cours représente le fruit de quatre années d'enseignement consécutives. D'année en année, le cours a été amélioré sur la base des interactions avec les étudiants et de leurs résultats obtenus, qui mettent en valeur les parties et aspects difficiles du cours et ayant nécessité de plus amples d'explications.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Comme tout projet d'entreprise, le développement des systèmes d'information (SI) d'une entreprise est une activité qui nécessite de suivre une méthode. Cette nécessité découle du besoin de satisfaire les attentes des futurs utilisateurs du système, tout en maitrisant les coûts et délais engendrés. Dans ce contexte, beaucoup de méthodes de développement de SI ont vu le jour. Les premières méthodes proposés préconisaient une séparation entre la structure du système et le fonctionnel. L'avènement de l'approche objet et son application dans le domaine du développement des SI a donné naissance aux approches orientées objets. Toutefois, la floraison des méthodes basées sur l'objet avec des notations variées de concepts similaires, parfois identiques, a poussé les chercheurs et industriels à unifier le langage de notation des concepts, ce qui a donné lieu au langage UML. De même, les processus de développement ont également étés unifié sous l'appellation de *processus unifié* (PU) ou *unified process*. Le PU est basé sur UML et énumère un ensemble d'activités à accomplir afin d'aboutir à un système tout en étant centré sur les besoins des utilisateurs.

Dans la pratique, l'application du PU a donné lieu à diverses implémentations, dont le Rationale Unified Process d'IBM. Dans ce cours, nous nous intéressons à l'implémentation two-track unified process (2TUP) de Pascal Roques. Le 2TUP met en action le langage UML tout au long d'un processus en Y qui sépare initialement les aspects fonctionnels d'un système des aspect opérationnels pour les unir par la suite.

Le processus 2TUP est enseigné à l'université de Jijel depuis une dizaine d'années, remplaçant ainsi la méthode Merise. Ce document, en plus de détailler ce processus et de servir de support du module « systèmes d'informations, méthodes avancées », représente un mini guide pour le développement des SI pour les étudiants en deuxième années de master dans le cadre de leur projets de fin d'études. Il peut également servir de memento du processus 2TUP pour les développeurs de petits systèmes d'information, étant donné que beaucoup de détails inutiles pour le besoin d'enseignement ont été omis.

Ce support de cours est structuré en neuf (09) chapitres. Le premier chapitre reprend les notions de bases nécessaires à la compréhension du processus 2TUP ainsi qu'à sa bonne application. Les chapitres restants sont ordonnés selon les étapes du processus. Les annexes à la fin du support regroupent l'ensemble des travaux dirigés, interrogations et examens ayant accompagné l'enseignement du processus tout au long des quatre dernières années. Les exercices ne sont malheureusement pas tous accompagnés de corrigés-types.

Enfin, et comme son nom l'indique, ce document étant un support de cours, il ne dispense pas de la présence physique de l'étudiant, vu que beaucoup de concepts du processus ont besoin d'être étayés avec davantage d'exemples.

Chapitre 1 : Notions de base

1. Introduction

Un système d'information (SI) est un ensemble de moyens matériels, logiciels, humains pour gérer l'information au sein de l'entreprise.

Le SI d'une entreprise assure quatre principales fonctions :

- Acquisition de l'information de l'environnement interne ou externe
- Transformation de l'information par traitement ou autre
- Stockage des données dans des bases de données ou autres
- Diffusion de l'information au sein de l'entreprise ou vers l'extérieur.

Le développement des systèmes d'information et de logiciel s'appuie sur une démarche, suivant un processus et en se basant sur un ensemble de modèles. UML offre à ce titre une solution en matière de notation des éléments composant un système. Etant simplement un langage de description, il doit être secondé d'un processus qui met en pratique l'ensemble des modèles qu'il propose. A ce titre, le Processus Unifié exécuté autour de UML permet d'encadrer l'élaboration des modèles d'UML, de maîtriser les risques et d'assurer la réutilisation des composants du système.

2. Concepts préliminaires

2.1. Génie logiciel

Le génie logiciel est une science de génie industriel qui étudie les méthodes de travail et les bonnes pratiques des ingénieurs qui développent des logiciels [1].

2.2. Modèles / Méthodes de développement

1.1.1. Modèle

Un modèle est une abstraction de la réalité. Selon [2], « *a data model is for a database designer what a box of colors is for a painter: it provides a means for drawing representations of reality* ». Il s'agit de la représentation des éléments d'une réalité pour n'en

retenir que les éléments essentiels pour un besoin donné. Il permet aussi de simplifier une réalité et en réduire la complexité. Un modèle est composé d'un ensemble de concepts de modélisation et de relations entre ces concepts ainsi qu'un ensemble de règles de modélisation. La modélisation est le passage de la réalité au modèle en utilisant un langage de modélisation.

Exemple : le modèle conceptuel de données, le modèle conceptuel de traitements (Merise [4]).

1.1.2. Méthode

Une méthode est un guide plus ou moins formalisé. Elle est constituée d'un ensemble de concepts de modélisations qui sont mis en œuvre en suivant une succession chronologique d'activité et en respectant un ensemble de règles.

Dans le domaine des SI, et selon Rambaugh [3], une méthode est *un ensemble de règles et de directives et se compose des éléments suivants*

- un ensemble de concepts fondamentaux de modélisation pour capturer la connaissance sémantique d'un problème et de sa solution. Des exemples de concepts sont l'entité, le processus, l'acteur...
- un ensemble de vues et de notations pour présenter la modélisation sous-jacente aux personnes qui les examineront et les modifieront. Des exemples de modèles sont le modèle entité/association, le réseau de Petri, ...
- un processus interactif pas-à-pas employé pour la construction des modèles et pour leur implantation ;
- une collection de suggestions et de règles qui conduisent à l'exécution du développement. Par exemple, séparer la modélisation des données de celles des traitements, mener certaines étapes en parallèle...

1.1.3. Modèle de développement de logiciels

Un modèle de développement de logiciel spécifie l'ensemble des étapes de développement indépendamment de tout modèle (donc de toute méthode). Il décrit la chronologie des étapes et leur organisation dans le temps.

Il existe un plusieurs modèles de développement de logiciels, dont les plus connus sont :

- le modèle en cascade
- le modèle en V
- le modèle en spirale.

2.3. Chronologie des méthodes de développement

Il existe une multitude de méthodes de développement de SI. On peut classer ces méthodes chronologiquement comme suit

- Années 70 : méthodes cartésiennes. Elles s'appuient sur un découpage fonctionnel et hiérarchique du système à analyser. Elles mettent l'accent sur l'aspect dynamique d'un

système tout en modélisant les données. Parmi les méthodes de cette catégorie on retrouve la méthode SADT (Structured Analysis and Design Technique).

- Années 80 : méthodes systémiques. Elles s'appuient sur la théorie des systèmes, en essayant d'aborder la réalité en tant que système complexe à modéliser. Les méthodes de cette catégorie sont orientée côté statique du système tout en modélisation les traitements mais de manière séparée. Exemple : Merise.

- Années 90. les années 90 sont caractérisées par l'apparition du paradigme objet en termes de méthodes d'analyse, puisqu'il existait depuis les années 60 en termes de programmation.

2.4. La démarche Objet

Les concepts de l'orienté objet ont débuté dans la programmation. Par la suite, durant les années 90, l'intérêt a été porté à ces concepts en matière de développement de systèmes d'information pour l'entreprise. Un des concepts-clé de l'introduction de l'objet dans le développement des systèmes étant la notion de réutilisation. En effet, la démarche objet permet de fabriquer des composants réutilisables dans d'autres contextes similaires, ce qui rentabilise l'effort du développement des SI.

Durant les années 90, on comptait déjà environ 50 méthodes différentes, mais les plus distinguées sont les méthodes OMT (Rambaugh), BOOCH, et OOSE (Jacobson). La fusion des trois méthodes ayant donné lieu en 1996 à UML, qui est soumise à l'OMG, et depuis, UML est devenu une notation reconnue dans le développement des systèmes.

Le développement des systèmes suit un cycle de vie qui définit les étapes, leur enchainement, et leur interdépendance. La diversité des méthodes a conduit à la diversité des cycles de vie des systèmes orientés objet. Cependant, il a été ressorti trois phases principales

- *L'analyse Orientée Objet* : il s'agit de l'abstraction des objets du monde réel en utilisant un modèle conceptuel ou spécification d'une application.

- *La conception Orientée Objet* : il s'agit d'organiser la solution informatique (la concevoir) avant de passer à l'implantation, en s'appuyant sur les spécifications de la phase d'analyse.

- *L'implantation Orientée Objet* : il s'agit ici de mettre en œuvre le logiciel en terme de programmes et base de données basés sur le paradigme objet.

Remarque : en général, les méthodes orientées-objet suivent un cycle de développement en spirale.

2.5. UML

1.1.4. Présentation

UML (Unified modeling language) est le résultat de la fusion, en 1996, de trois méthodes de développement orientées-objet (OMT, OOD et OOSE). UML devient une spécification de

l'object management group en 1997 [5]. Actuellement, UML est en version 2.1.2 et la version 2.2 est en cours de préparation. UML n'est pas une méthode [6] au sens propre de méthode mais plutôt un ensemble de notations graphiques (modèles) qui peuvent être utilisées par une méthode.

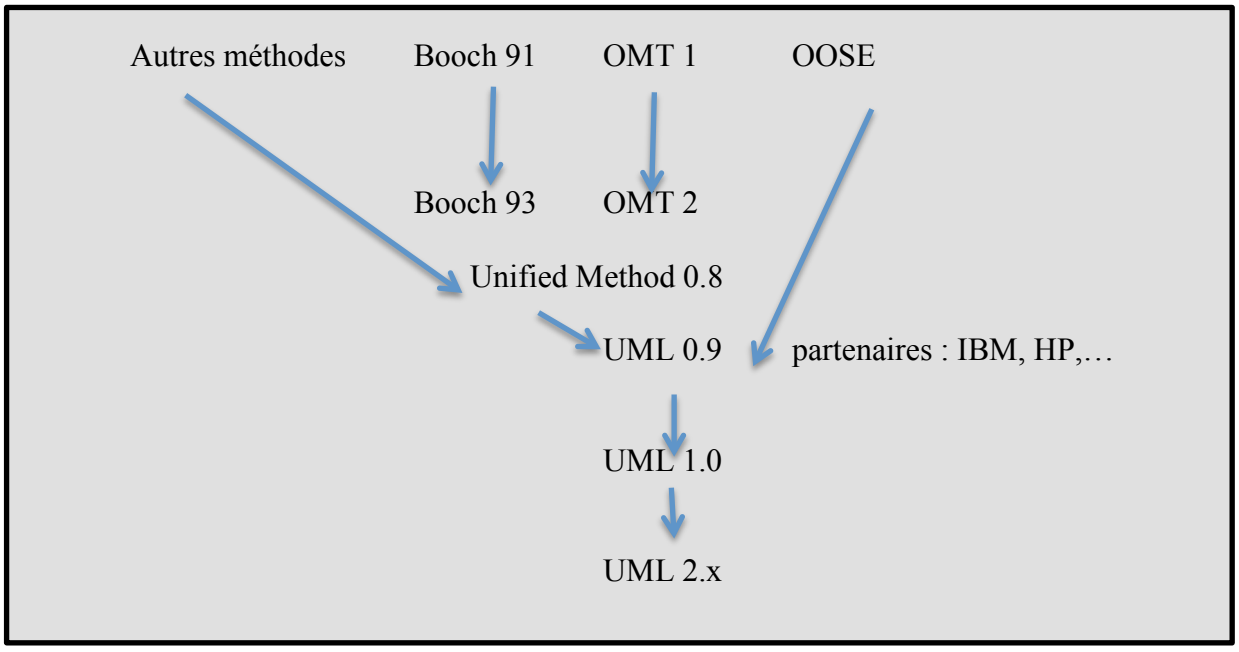


Figure 1. Evolution de UML

UML n'est pas une méthode, c'est un langage unifié qui offre un ensemble de diagrammes représentant différentes vues du système à développer. UML n'a pas été accompagné d'une méthode de développement pour respecter les spécificités des entreprises. Mais, une bonne pratique de développement nécessite d'intégrer l'élaboration des modèles dans un processus pour

- guider le développement : savoir d'où on vient et où on va, ce qu'on a fait, ce qui reste à faire
- maitriser les coûts du développement
- maitriser les délais associés à chaque étape
- gérer les risques liés au projet (ex : rejet par les utilisateurs du système)
- assurer l'évolutivité du système en tenant compte de nouveaux besoins ou des changements qui se produisent dans l'environnement de l'entreprise.

Les efforts déployés pour fournir un processus accompagnant UML se sont soldés par la proposition du processus unifié.

1.1.5. Processus Unifié (Unified Process) :

Il s'agit d'un processus de développement de logiciels construit autour d'UML [7]. Les activités de développement autour de ce processus sont énumérées comme suit :

(1) capture des besoins des utilisateurs,
(2) analyse des besoins
(3) conception des solutions

(4) implémentation des solutions par des outils informatiques
(5) mise en œuvre (codage, tests, etc.).

1.1.6. Caractéristique de UP

Le processus unifié se caractérise par ce qui suit

- Orienté utilisateur : il répond aux besoins métiers à travers les cas d'utilisation (use cases). Le cas contraire est de développer un système générique qui ne tient pas compte ni des spécificités de l'entreprise ni des besoins des utilisateurs.

- Centré sur l'architecture: c'est-à-dire il prend en compte l'architecture de l'entreprise (deux-tiers, trois-tiers, multi-tiers).
- Incrémental : il permet d'élaborer des résultats intermédiaires qui peuvent être évalués par rapports aux spécifications et donc de maitriser les risques, les coûts et les délais. Le cas contraire est de développer un système comme un seul bloc indivisible.
- Piloté par les risques : il permet de prendre en compte les besoins des utilisateurs et les spécificités et contraintes de mise en œuvre.
- Orienté modèle : à travers les étapes du processus, les diagrammes UML sont utilisés (classes, cas d'utilisation, ...) afin de capturer les besoins ou de concevoir le système.
- Permet la réutilisation car centré sur le développement de composants réutilisables.

1.1.7. Le processus 2TUP

Le processus unifié est une spécification générale qui a été implémentée par plusieurs processus concrets (ex : 2TUP, RUP…). Le processus 2TUP (2-Track Unified Process) [8, 9] est un processus respectant le modèle en Y du développement et qui sépare les aspects techniques des aspects fonctionnels.

1.1.8. Schéma général du processus 2TUP

Le schéma général du processus 2TUP est le suivant [9]

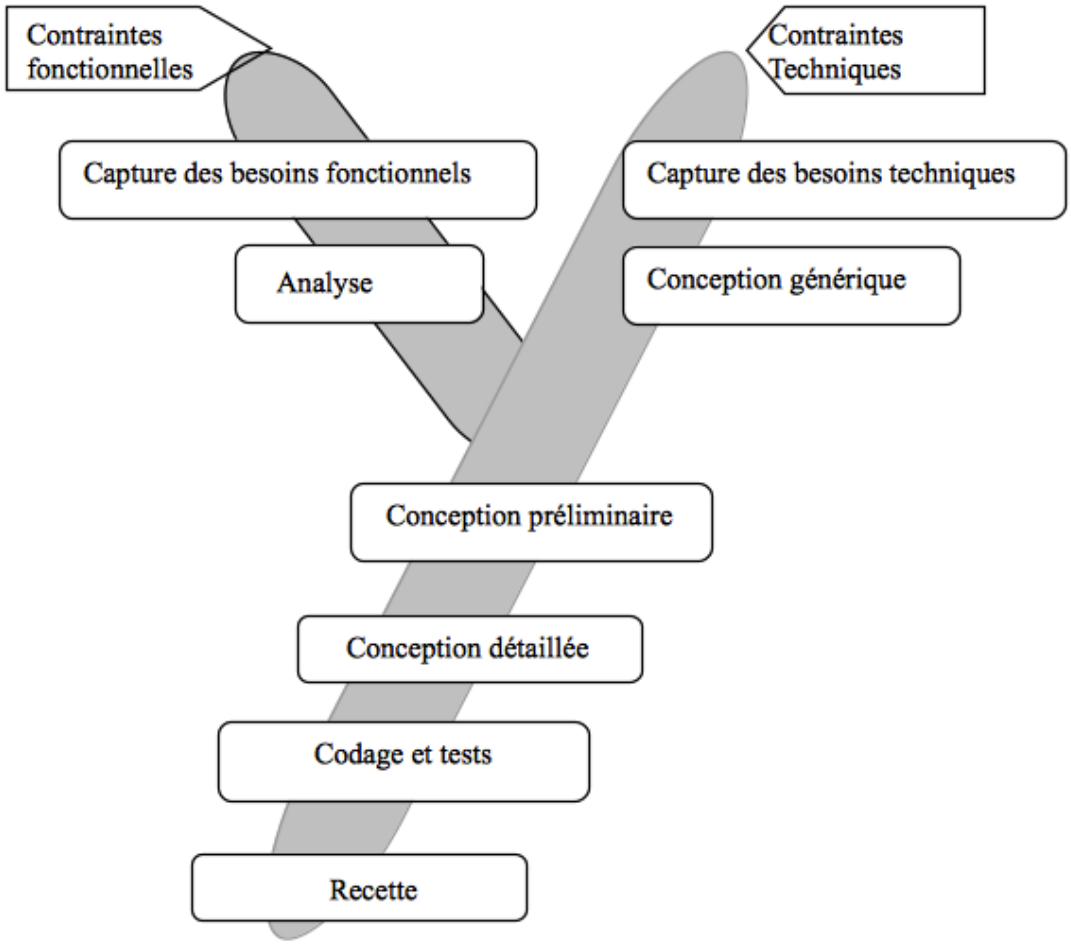


Figure 2. Processus 2TUP

Remarque : les étapes du processus 2TUP sont précédées par une étape de prise de contact avec le domaine et l'environnement de l'entreprise.

Ces étapes sont résumées comme suit :

- **Contraintes fonctionnelles** : limites et conditions à respecter qui sont liées au métier des utilisateurs.
- **Contraintes techniques** : limites et conditions à respecter qui ne sont liées au métier des utilisateurs mais sont d'ordre technique.
- **Capture des besoins fonctionnels :** recensement des besoins liés au métier que les utilisateurs veulent qu'ils soient pris en compte dans le système. Ces besoins sont capturés à l'aide des diagrammes de cas d'utilisation.
- **Capture des besoins techniques :** recensement des besoins à caractère technique non liés au métier des utilisateurs. Ces besoins sont capturés à l'aide des diagrammes de composants et de déploiement.
- **Analyse :** étude des besoins des utilisateurs et leur modélisation. On utilise essentiellement les diagrammes de classes et le diagramme de séquences.
- **Conception générique :** proposition de solution technique indépendante du métier
- **Conception préliminaire :** croisement de l'analyse et de l'architecture générique du système
- **Conception détaillée :** approfondissement de la conception (typage de données, détail

des algorithmes, etc.)

- **Codage et tests :** création des bases de données, implémentation des programmes, dans des environnements IDE ou autres.
- **Recette :** remise du système aux utilisateurs finaux.

Dans ce cours, nous allons traiter les premières étapes de la capture des besoins jusqu'à la conception détaillée.

3. Conclusion

Dans ce chapitre, nous avons rappelé la notion de système d'information et passé en revue les différentes approches de développement des systèmes d'informations. Nous avons également présenté l'approche orientée objet pour le développement ainsi que le processus unifié. Enfin, nous avons introduit le processus 2TUP qui sera la base de ce cours.

Chapitre 2 : Rappels sur l'objet et UML

1. Introduction

Les concepts de l'orienté objet ont débuté dans la programmation. Par la suite, durant les années 90, l'intérêt a été porté à ces concepts en matière de développement de systèmes d'information pour l'entreprise. Durant les années 90, on comptait déjà environ 50 méthodes différentes, mais les plus distinguées sont les méthodes OMT (Rumbaugh), BOOCH, et OOSE (Jacobson). Un effort d'unification des notations utilisées dans les méthodes objet ont donné lieu en 1996 à UML, qui est soumise à l'OMG, et depuis, UML est devenu une notation reconnue dans le développement des systèmes.

2. UML

- Résultat de la fusion, en 1996, de trois méthodes de développement orientées-objet (OMT, OOD et OOSE). Devenu spécification de l'Object management group en 1997.
- Les versions actuelles d'UML sont notées UML 2.x.
- UML n'est pas une méthode au sens propre de méthode mais plutôt un ensemble de notations graphiques (modèles) qui peuvent être utilisées par une méthode.

2.1. Diagrammes d'UML

Actuellement, UML utilise 14 diagrammes, comme schématisé par le diagramme de classes suivants :

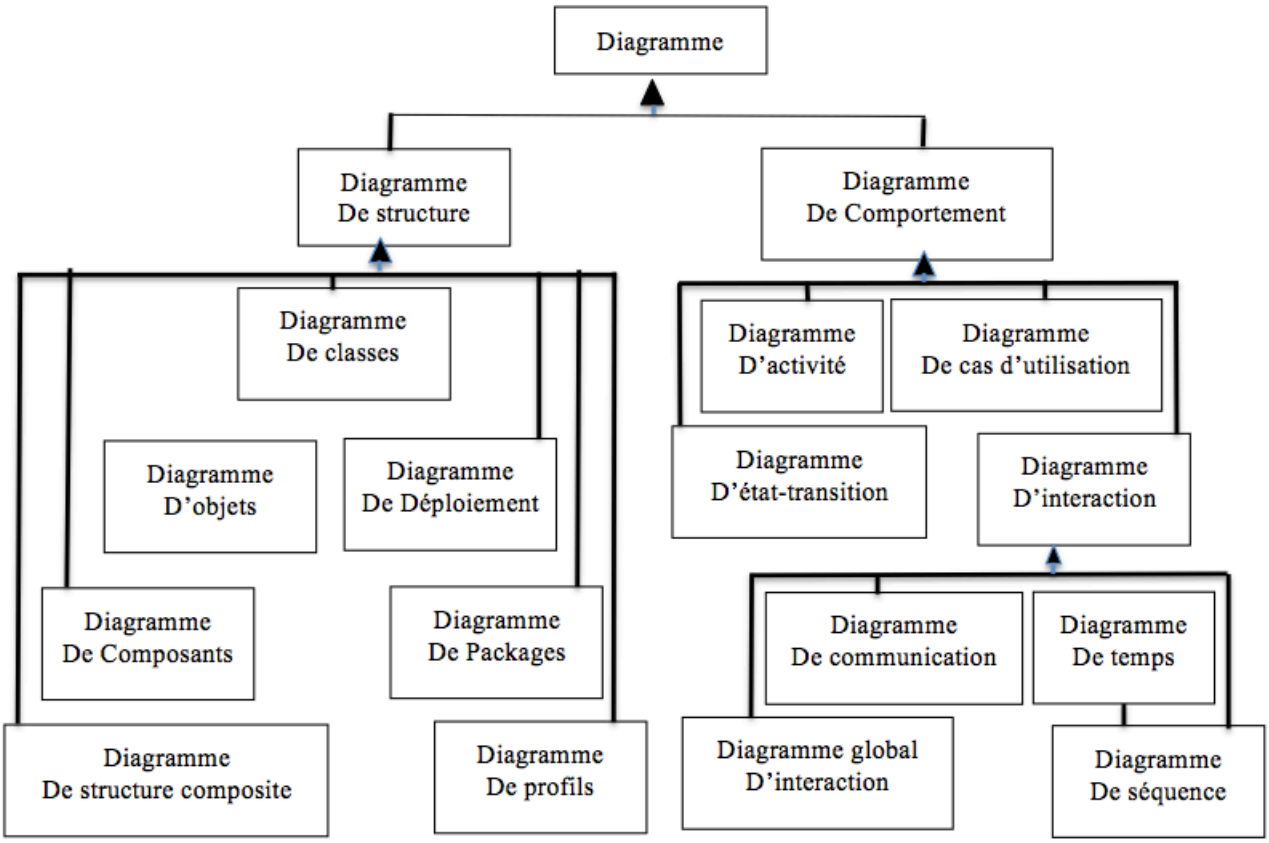


Figure 3. Diagrammes d'UML

2.2. Présentation de quelques diagrammes

Dans ce cours, nous nous contentons de présenter le diagramme de classes qui, selon nous, est l'un des diagrammes essentiels, notamment pour aboutir à un schéma de base de données correct. Aussi, nous présentons le diagramme d'objets qui sert à valider le diagramme de classes en amont et en aval. Le lecteur peut se référer à la documentation correspondant aux autres diagrammes.

2.2.1. Le diagramme de classes

- **Le concept d'objet**

Il existe plusieurs définitions du concept d'objet mais elles sont convergentes. Nous adoptons la définition suivante :

« Un objet est une entité abstraite ou concrète de la réalité. Un objet est caractérisé par un état interne constitué des attributs et par un comportement qui représente l'ensemble d'opérations qui permettent de changer l'état interne de l'objet ».

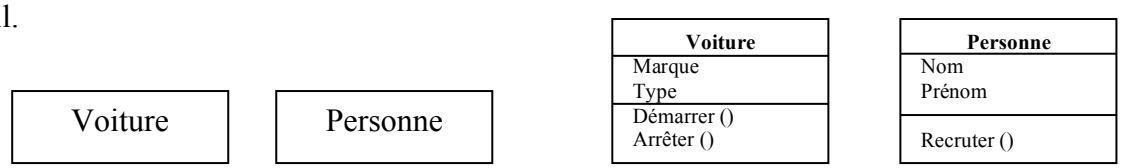
Exemple : objets concrets *voiture 1727, personne 877,…* objets abstrait : *trajet 123, processus A23B2, …*

- **Le concept de classe**

Une classe d'objet représente une catégorie de plusieurs objets ayant la même structure. Un objet représente l'instance d'une classe (dans un système à un seul niveau d'instanciation). Par contre, une classe représente l'abstraction de plusieurs objets ayant la même structure.

Exemple : les classes correspondant aux objets de l'exemple précédent sont *voiture, personne, trajets, processus*

Une classe est représentée graphiquement sous la forme d'un rectangle avec plus ou moins de détail.



Représentation non détaillée *représentation détaillée*

Figure 4. Représentation des classes

- **Notion d'encapsulation**

L'encapsulation d'un objet consiste à protéger l'état de l'objet (les attributs) en n'affichant que les méthodes qui permettent de les modifier. Aussi, le code des méthodes est caché à l'utilisateur externe à l'objet. Ceci permet de protéger les données d'un objet et d'en assurer l'intégrité. Les fonctions présentées à l'utilisateur de la classe sont appelées interfaces.

- **L'attribut**

Un attribut est une propriété caractérisant une classe. Il est représenté dans la partie du milieu dans la classe. Un attribut possède un type de données prédéfini ou défini par l'utilisateur.

Exemples de types de données prédéfinis : String, Int, Double, Float, Date, Booelan …
Exemples de types de données définis par l'utilisateur : T_couleurs pour l'attribut Couleurs, tel que les valeurs permises soient vert, blanc, noir.

Exemple : la figure suivante représente les attributs de la classe personne.

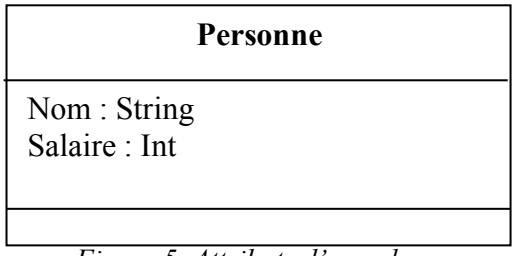


Figure 5. Attributs d'une classe

- **Valeurs initiales et valeur par défaut**

Un attribut peut avoir une valeur initiale et une valeur par défaut. Cela est spécifié par les symboles = et # avant la valeur

Exemple : la figure suivante représente les valeurs initiales des attributs Salaire et Prime.

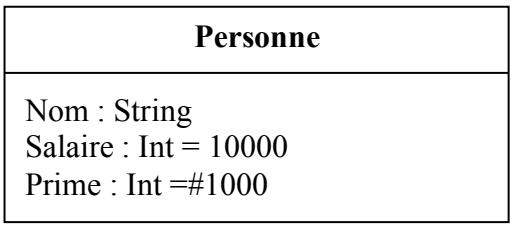


Figure 6. Valeurs initiales des attributs d'une classe

- **Multiplicité des valeurs d'un attribut**

Les classes UML ne respectent pas forcément la première forme normale, dans le sens où un attribut peut être multivalué. Cela est représenté par les cardinalités (multiplicités d'attributs), entre crochets. Les valeurs des multiplicités possibles sont 0 (pas de valeur) ; 1 (une valeur) , * (plusieurs valeurs).

Exemple : l'attribut ages_enfants est multi-valué pour permettre de représenter différents âges.

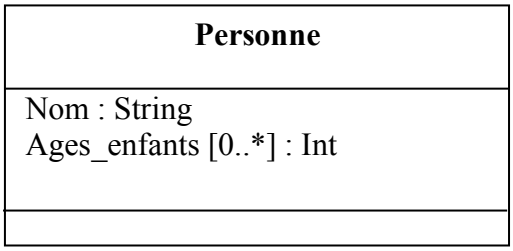


Figure 7. Attributs multi-valués

- **Visibilité des attributs**

La visibilité d'un attribut est définie par le spectre dans lequel l'attribut est visible dans l'objet. Il existe trois visibilités

- public (+) : l'élément est visible pour tous les clients de la classe
- protégé (#) : l'élément est visible pour les sous-classes de la classe
- privé (-) : l'élément n'est visible que par les objets de la classe dans laquelle il est déclaré.

Exemple : l'attribut Nom est public, l'attribut Salaire est protégé, par contre l'attribut Prime est privé.

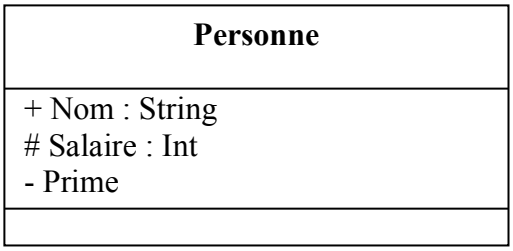


Figure 8. Visibilité des attributs

- **Stéréotypes d'attributs**

Les attributs peuvent être stéréotypés pour ajouter de nouvelles informations non prises en charge par les éléments de modélisation d'UML.

Exemple : les attributs identifiant et salaire sont stéréotypés respectivement « unique » et le stéréotype « const ».

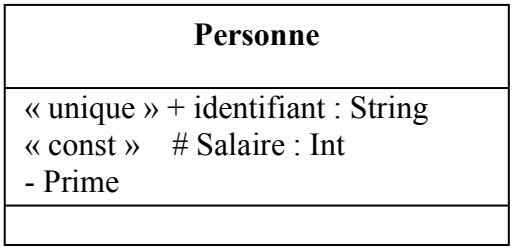


Figure 9. Stéréotype d'attributs

- **L'opération (méthode)**

Une opération (appelée aussi méthode) est un comportement de l'objet. Il s'agit d'une fonctionnalité assurée par la classe. Une opération est désignée par un verbe et est suivie de parenthèses. Une opération est caractérisée par son nom, la liste des paramètres (chaque paramètre est décrit par son nom, son type et éventuellement sa valeur par défaut), le type de résultat et ainsi que des propriétés indiquées entre accolades.

Opération (liste de paramètres) : type résultat {propriétés}

Exemple : on peut définir les opérations démarrer, arrêter et réparer pour une voiture.

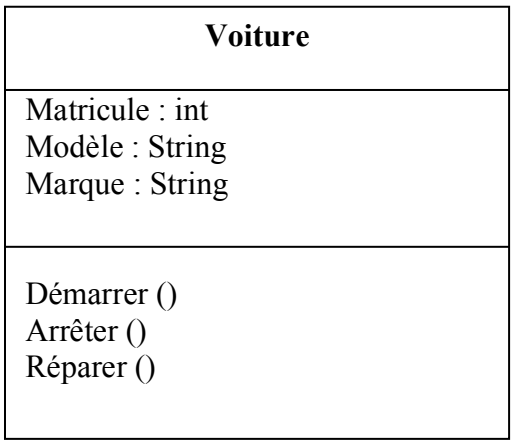


Figure 10. Opérations de classes

- **Visibilité d'une opération**

Comme l'attribut, une opération peut avoir trois visibilités : protégée, publique et privée.

- public (+) : l'opération est visible pour tous les clients de la classe
- protégé (#) : l'opération est visible pour les sous-classes de la classe
- privé (-) : l'opération n'est visible que par les objets de la classe dans laquelle elle est déclarée.

Exemple : l'exemple ci-dessous montre différentes visibilités des opérations sur une voiture.

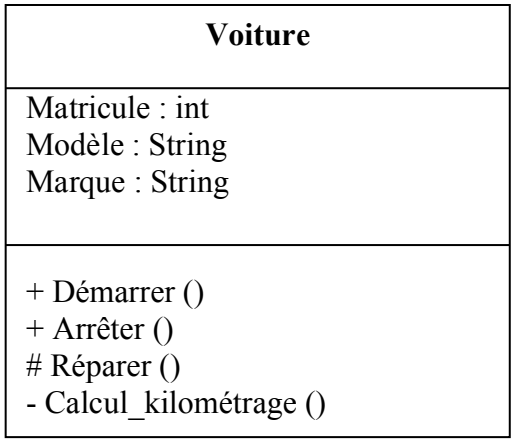


Figure 11. Visibilité des opérations

- **Les relations entre les classes**

L'association

Il s'agit d'un lien entre des classes ou des objets. Dans certaines méthodes, on distingue entre association entre objets et association entre classes. Sinon, l'existence d'une association entre les objets suppose son existence entre leurs surclasses. Une association peut être caractérisée par une multiplicité, des rôles, et peuvent être élaborées pour être des classes à part entière. L'association la plus simple est binaire et lie deux classes. La terminaison d'une association peut être désignée par un *rôle.*

Exemple : dans l'exemple suivant, nous associons une personne à une société dans le sens que la personne travaille au sein de la société.

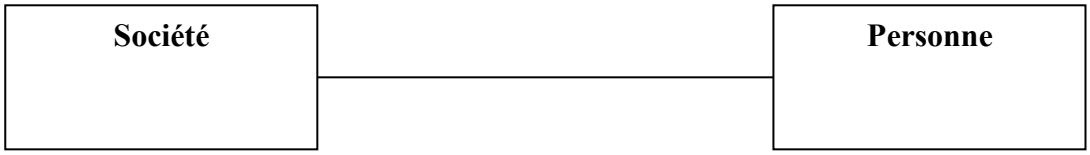


Figure 12. Association entre classes

Une multiplicité désigne le nombre d'instances d'une classe qui participent avec une instance de la classe avec laquelle elle est liée par une association. Les multiplicités possibles sont 0..1 ; 1 (1..1) ; 0..* (*) ; 1..* ; 1..n ; 2,4,..*.

Exemple : la figure suivante illustre une association employé employeur entre des personnes et des sociétés.

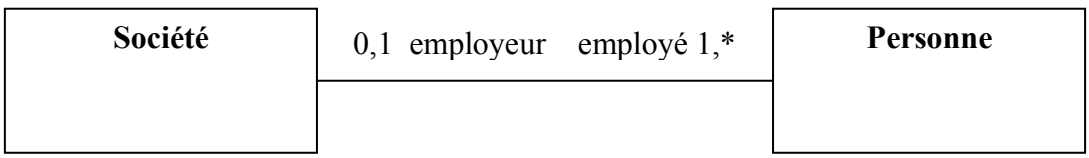


Figure 13. Multiplicité d'association entre classes

La *navigabilité* désigne la possibilité d'atteindre la ou les instances d'une classe à partir de l'instance de l'autre classe à travers l'association. Cela est représenté par une flèche lorsque la

navigabilité est possible et par une croix lorsqu'elle est interdite.

Exemple : dans l'exemple suivant, on ne peut connaître les personnes travaillant dans une société mais on ne peut pas connaître la société dans laquelle travaille chaque personne.

Figure 14. Navigabilité d'associations

La notion de *classe d'association* est similaire à la notion de relation porteuse de propriétés dans modèle conceptuel des données. Dans ce cas, les attributs de la classe d'association dépendent fonctionnellement des identifiants des classes liées par l'association. Par contre, la classe d'association peut être liée à d'autres classes indépendamment des classes qui sont à son origine. Une classe d'association est représentée comme suit.
Exemple : l'exemple suivant montre une classe d'association ayant le sens de ligne de commandes.

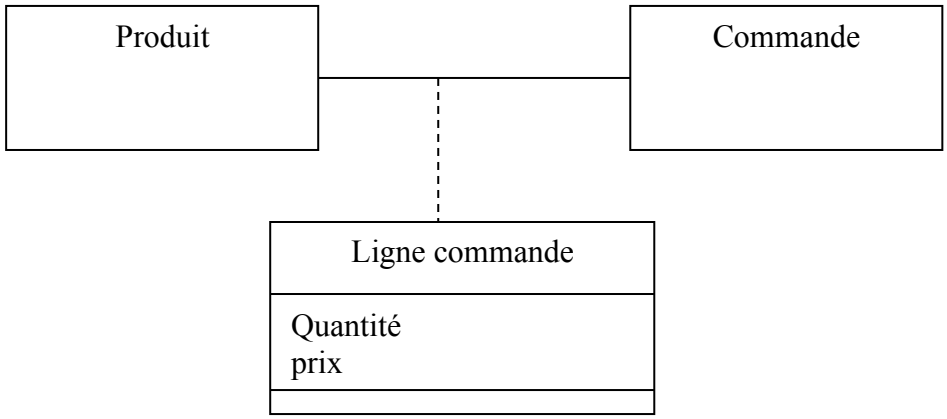


Figure 15. Classes d'association

Une association n-aires est une association qui lie plus de deux classes. Elle est représentée par un losange.

Exemple : l'exemple ci-dessous représente une association ternaire entre trois classes.

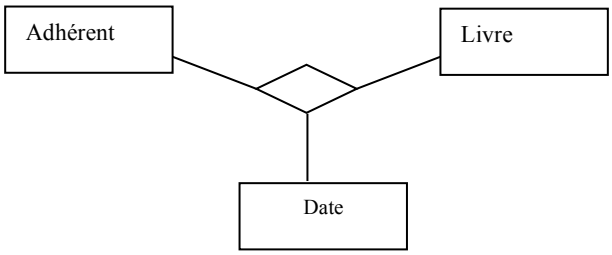


Figure 16. Association ternaire

Dans certains cas, il est possible de découper une association n-aires en n-1 associations binaires. L'association sera dans ce cas une classe liée à chaque classe qui participait dans la relation. La décomposition d'une association n-aire peut simplifier la lecture d'un diagramme de classes, mais peut se traduire par la perte de la sémantique voire la perte de la l'intégrité de

certaines données.

Exemple : la classe Prêt suivante est le résultat de la décomposition d'une association ternaire entre adhérent, livre et a

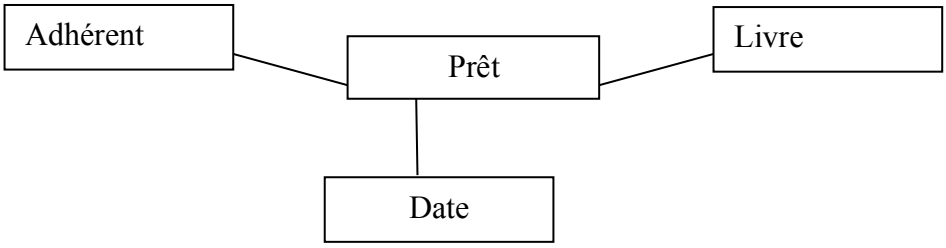


Figure 17. Décomposition d'une association

L'agrégation et la composition

L'*agrégation* traduit un relations de type composant /composé entre deux classes. Il s'agit d'une relation transitive et antisymétrique. L'agrégation est représentée par un losange du côté de l'agrégat. L'agrégation peut utiliser les multiplicités.

Exemple : la figure suivante illustre deux agrégations entre direction et sous-direction d'une part et sous-direction et service d'autre part.

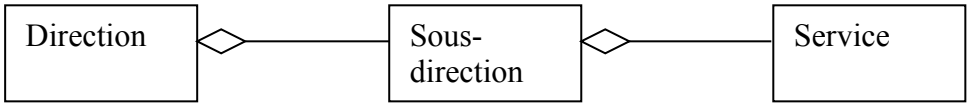


Figure 18. Aggrégation de classes

La *composition* est un type particulier d'agrégation dans lequel la classe composant n'a pas d'existence indépendante de l'existence de ses composants. Il s'agit souvent de composition physique. La composition est représentée par un losange noir.

Exemple : la figure suivante illustre une composition entre Livre et Introduction, Chapitre et Conclusion.

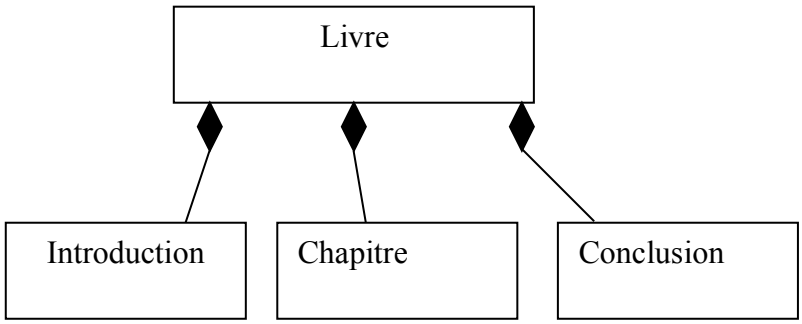


Figure 19. Composition de classes

L'héritage (spécialisation / généralisation)

La notion d'héritage décrit une relation hiérarchique entre des classes d'objets sous la forme d'héritage d'attributs et de comportements. Cette relation est transitive et antisymétrique.
La classe héritée est appelée surclasse ou classe –mère, alors que la classe héritant est appelée sous-classe ou classe-fille. La création de surclasse à partir de classes-filles est appelée *généralisation* alors que la création de sous-classes à partir d'une surclasse est appelée *spécialisation*. La généralisation consiste à prendre des classes existantes déjà mises en évidences) et de créer de nouvelles classes qui regroupent leurs parties communes ; il faut aller du plus spécifique au plus général. La spécialisation consiste à sélectionner des classes existantes (déjà identifiées) et d'en dériver des nouvelles classes plus spécialisées, en spécifiant simplement les différences. L'héritage peut être simple lorsqu'une classe hérite d'une seule classe ou multiple lorsqu'une classe hérite de plusieurs classes en même temps. L'héritage est représentée par une flèche (qui peut avoir plusieurs extrémités de début) et pontant directement sur la classe mère.

Exemple : dans l'exemple ci-dessous, un employé vacataire et un employé permanent sont des cas particuliers d'employés et ils en héritent les caractéristiques communes.

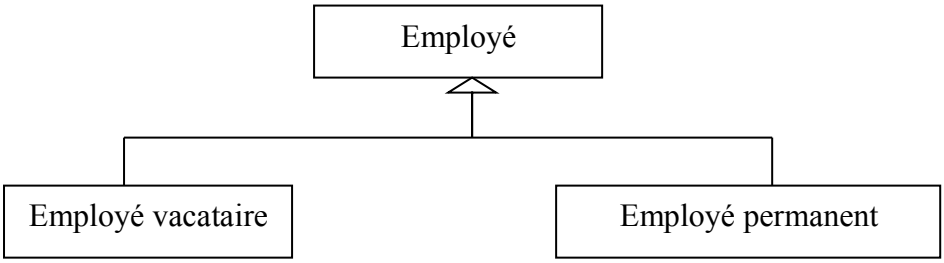


Figure 20. Spécialisation / Généralisation

Remarque : le digramme de classe est un ensemble de classes (au moins une) liées entre elles des différentes manières vues dans le chapitre. Une manière de construire un diagramme de classes sera décrite dans la présentation du processus unifié.

2.2.2. Le diagramme d'objets

Le diagramme d'objets, permet de représenter les instances des classes, c'est-à-dire des objets. Il exprime les relations qui existent entre les objets, mais aussi l'état des objets. Il est moins général que le diagramme de classes.

Exemple : la figure suivante illustre un digramme de classes et un diagramme d'objets correspondant.

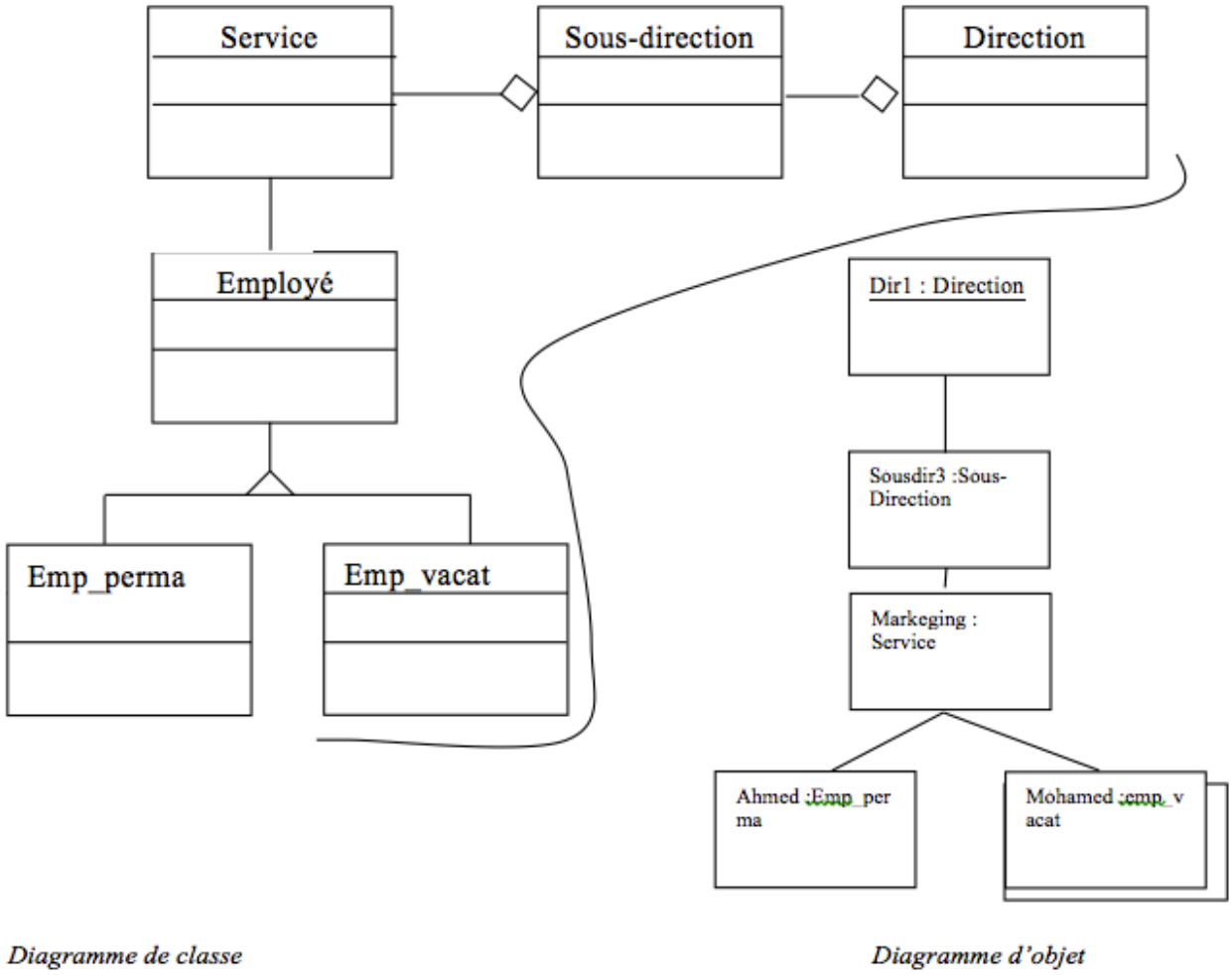


Figure 21. Diagramme de classes et diagramme d'objets

2.3. Concepts transversaux entre diagrammes

Tout au long de la présentation des concepts d'UML, un ensemble de concepts est utilisé de manière transversale aux diagrammes. Ces concepts sont décrits dans ce qui suit.

2.3.1. Stéréotype

Un stéréotype UML permet d'étendre les éléments de modélisation d'UML pour répondre à des besoins spécifiques de modélisation. Un stéréotype exprime le fait d'une classe par exemple soit l'instance d'une méta-classe définie par l'utilisateur.

Exemple : On peut stéréotyper les classes Fournisseur et Client comme des Partenaires.

Figure 22. Concept de stéréotype

2.3.2. Contrainte

Une contrainte exprime une relation ad-hoc entre deux éléments de modélisation ou plus. Elle est spécifiée entre {} et peut être écrit en langage naturel, en pseudo-code ou dernièrement, en utilisant un langage de formalisation des contraintes OCL.

Exemple : on peut exprimer la contrainte de valeur sur un attribut *prix* sous la forme de {prix>0}.

2.3.3. Note

Appelée aussi commentaire, il s'agit d'un texte qui décrit, en langage naturel un élément de modélisation.

Exemple : l'exemple suivant illustre une note décrivant la classe client.

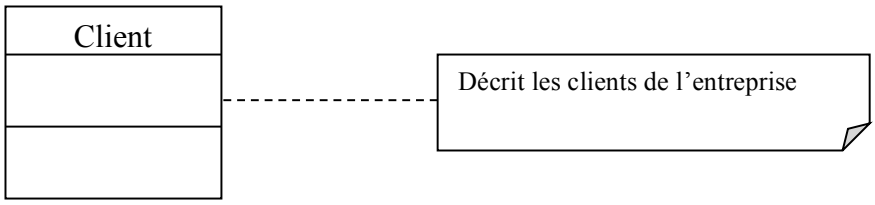


Figure 23. Concept de note

2.3.4. Paquetage

Un paquetage définit un ensemble d'éléments de modélisation homogènes. Dans le cas de classes, il s'agit d'empaqueter des classes à fort couplage entre elles.

Exemple : l'exemple suivant illustre trois paquetages avec leur contenu en termes de classes.

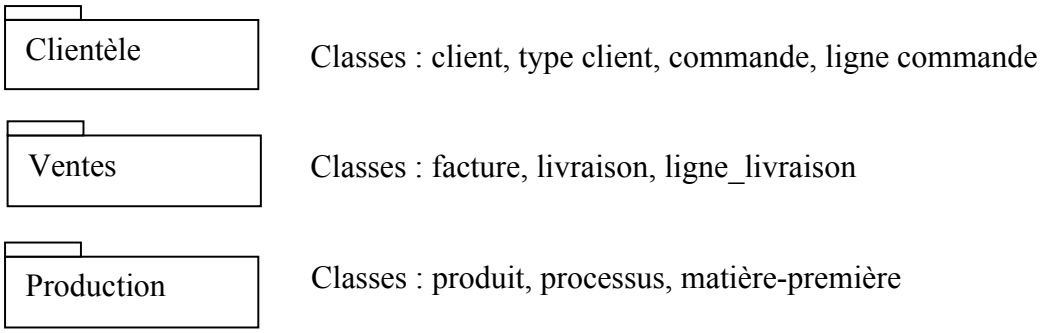


Figure 24. Concept de paquetage

2.3.5. Dépendance

Une dépendance entre des paquetages afin d'exprimer le fait qu'un élément d'un paquetage utilise un élément ou plus d'un autre paquetage.

Exemple : le schéma suivant montre une dépendance entre la gestion de la production, la gestion de stock et la gestion des ventes.

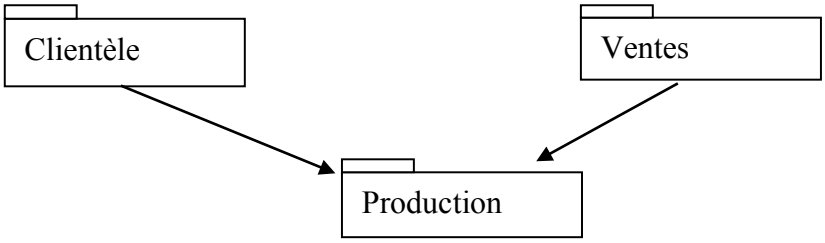


Figure 25. Concept de dépendance

3. Les diagrammes UML et le processus 2TUP

Les différents diagrammes d'UML trouvent naturellement leur terrain d'utilisation dans le cadre du processus UP. Cette utilisation se manifeste comme suit :

- Diagramme des cas d'utilisation : il permet de capturer et représenter les besoins fonctionnels et techniques du système.

- Diagramme de classes et d'objets : il permet de représenter l'aspect statique du système, tant du côté fonctionnel que technique en définissant entre autres les classes d'analyse fonctionnelles et les classes d'analyse techniques.

- Diagrammes de composants : il intervient dans les étapes de conception générique et conception détaillée. En conception générique, il permet de définir les composants du système indépendamment des fonctionnalités. En conception détaillée, il intervient dans la confrontation des besoins fonctionnels avec les composants définis.

- Diagrammes de déploiement : de même que le diagramme de composants, le diagramme de déploiement intervient dans la phase de conception générique pour définit l'architecture du système indépendamment des besoins fonctionnels. Il intervient aussi dans la conception préliminaire.

- Diagrammes de séquences et de collaboration : les diagrammes de séquences et de collaboration interviennent dans la phase de capture des besoins fonctionnels pour permettre de détailler le diagramme des cas d'utilisation.

- Diagramme d'états-transition : ce diagramme intervient dans les étapes d'analyse et de conceptions pour compléter les connaissances sur les classes.

- Diagramme d'activités : ce diagramme consolide les spécifications des besoins illustrées par les cas d'utilisation.

4. Conclusion

Dans ce chapitre, nous avons présenté la panoplie des diagrammes d'UML tout en mettant l'accent sur le diagramme de classes. Cependant, ce cours n'étant pas dédié à UML, il appartient à l'étudiant de compléter ses connaissances sur les diagrammes UML ou de réviser les connaissances acquises dans le cursus de licence. Les chapitres suivants sont consacrés aux étapes du processus 2TUP et le cheminement du cours suit l'enchainement des étapes du processus.

Chapitre 3 : Etude préliminaire

1. Introduction

L'étude préliminaire permet de recueillir des informations initiales sur le système. Il s'agit dans cette phase de définir le contour du système, les différents acteurs, ainsi que les messages d'interaction entre les acteurs et le système.

2. Elaboration du cahier des charges

Le cahier des charges (CdC) document qui rassemble les spécifications du projet. A cette étape s'effectue l'identification de l'entreprise, ses activités, et les activités concernées par le projet. Le CdC doit contenir les choix techniques de l'entreprise. Par exemple

- La méthode ou approche de développement. **Exemple** : UP/UML,
- L'architecture de développement / déploiement. **Exemples** : Client/serveur, trois tiers,...
- L'infrastructure de (télé) communication. **Exemples** : réseau privé, Internet...
- Le langage ou environnement de développement. **Exemples** : Java, ...
- Le type du SGBD. **Exemples** : SGBDR, base XML…

Besoins fonctionnels et besoins opérationnels

Le nouveau système à placer doit satisfaire un ensemble de besoins. On distingue deux types de besoins :

- fonctionnels : ils concernent le métier, i.e. les tâches relevant du domaine étudié. **Exemple** : inscrire les étudiants, établir les documents de scolarité, rechercher un étudiant par son nom ou numéro d'inscription…
- opérationnels : cela concerne les aspects liés à l'exploitation du système mais indépendants des fonctions.
  Exemple : sécuriser l'application, prendre en charge un grand volume de données à échanger, assurer la disponibilité H24, 7/7 du système…

Remarque : les besoins métier et opérationnels sont exprimés par les utilisateurs du système. Cette expression ne doit pas sous-entendre nécessairement les solutions (informatiques) pour répondre aux besoins.

Exemple pratique : la fiche ci-dessous représente un extrait de cahier de charges d'un système de gestion de personnel pour une entreprise X.

Cahier des charges

Le projet est à réaliser dans l'entreprise X, en particulier le service des ressources humaines. Le système à développer doit répondre aux besoins fonctionnels suivants :

- Centraliser les informations des employés en une seule base de données et ainsi éliminer le risque d'incohérence.
- Réduire le nombre d'exemplaires de certains documents.
- Automatiser les tâches de gestion du personnel qui sont les suivantes : inscription des informations de base, gestion des absences, ajouter un employé, consulter et modifier leurs informations, supprimer un employé, noter un employé, saisir la fiche de pointage, ajouter et consulter les absences (gérer l'assiduité), qualifier un employé, gérer les congés, gérer les sanctions, gérer la dégradation, gérer la promotion, réintégrer un employé, gérer les arrêts du travail et la fin de carrière d'un employé.

Tableau 1. Exemple de cahier de charges

Les besoins opérationnels sont exprimés comme suit :

- authentification des utilisateurs par login et mot de passe
- l'administrateur est la seule personne autorisée à définir et modifier les mots de passe

Les grands choix techniques

- Processus de développement : processus de développement suivi (processus 2TUP)
- Langage de modélisation : UML
- Langage de programmation : PHP
- SGBD : MYSQL
- Architecture : Client/serveur

3. Identification des acteurs

Un acteur est un utilisateur type du système qui représente une responsabilité par rapport au système ou un rôle plutôt qu'une personne physique. Un acteur peut être concrétisé par une personne, un système ou une machine.

Un acteur est représenté par le symbole  auquel on peut associer son rôle par rapport au système sous la forme d'une note UML. On peut aussi définir une hiérarchie des acteurs.

Erreurs à éviter

L'identification du contour du système est parfois difficile et les deux erreurs à éviter sont les suivantes :

- Impliquer des acteurs indirects qui n'ont pas d'interaction directe avec le système.
- Identifier des composants du système en tant qu'acteurs.

Exemple pratique : le schéma ci-dessous illustre les acteurs du système SGP.

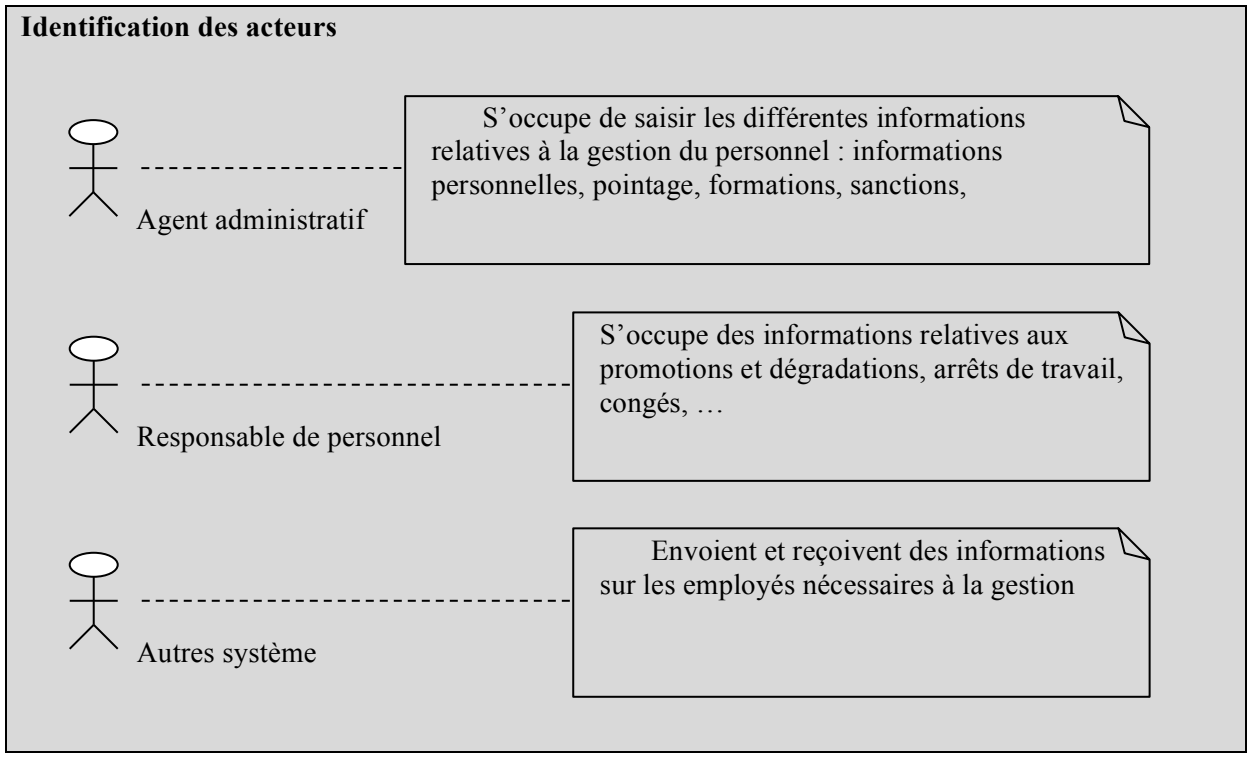


Figure 26. Exemple d'identification d'acteurs

4. Identifier les messages

Un message est la spécification d'une communication d'informations entre un émetteur et un récepteur. La notion de message n'est pas limitée à l'échange de messages entre les objets. A ce niveau, elle représente l'échange d'informations entre le système et les différents acteurs. L'acteur envoie des messages d'alimentation du système et en reçoit éventuellement les mêmes messages (en guise de consultation) ou carrément de nouveau messages (issus de traitements). Les messages doivent être identifiés entre chaque acteur et le système. → *Erreurs à éviter :* identifier les messages entre les acteurs eux-mêmes.

Exemple : la fiche ci-dessous illustre les messages entre le système et l'agent administratif.

Identification des messages : on va s'intéresser à l'échange effectué entre le système et l'acteur « agent administratif » - Le système qu'on va appeler SGP envoie les informations suivantes (liste non exhaustive): les informations des employés, la liste des employés, la liste des formations, les liste des employés sanctionnés, les employés candidats aux promotions, aux avancements en échelons - Par contre, il reçoit les informations suivantes (liste non exhaustive): les informations détaillées de chaque employé, les affectations, formations, sanctions…

Tableau 2. Exemple de messages entre les acteurs

5. Modéliser le contexte

Tous les messages (système <--> acteurs) identifiés dans l'étape précédente peuvent être présentés de façon synthétique sur un diagramme, que l'on peut qualifier de diagramme de contexte dynamique. On utilise un diagramme de collaboration de la façon suivante :

- Le système étudié est représenté par un objet central ;
- Cet objet central est entouré par un autre objet symbolisant les différents acteurs.
- Des liens relient le système à chacun des acteurs ;
- Sur chaque lien sont montrés les messages en entrée et en sortie du système.

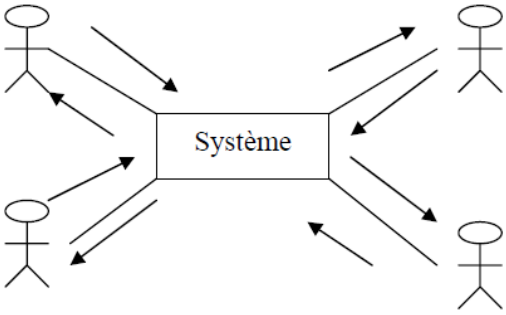


Figure 27. Diagramme de contexte dynamique

Exemple : la figure ci-dessous illustre le diagramme de contexte dynamique du système de gestion d'une bibliothèque

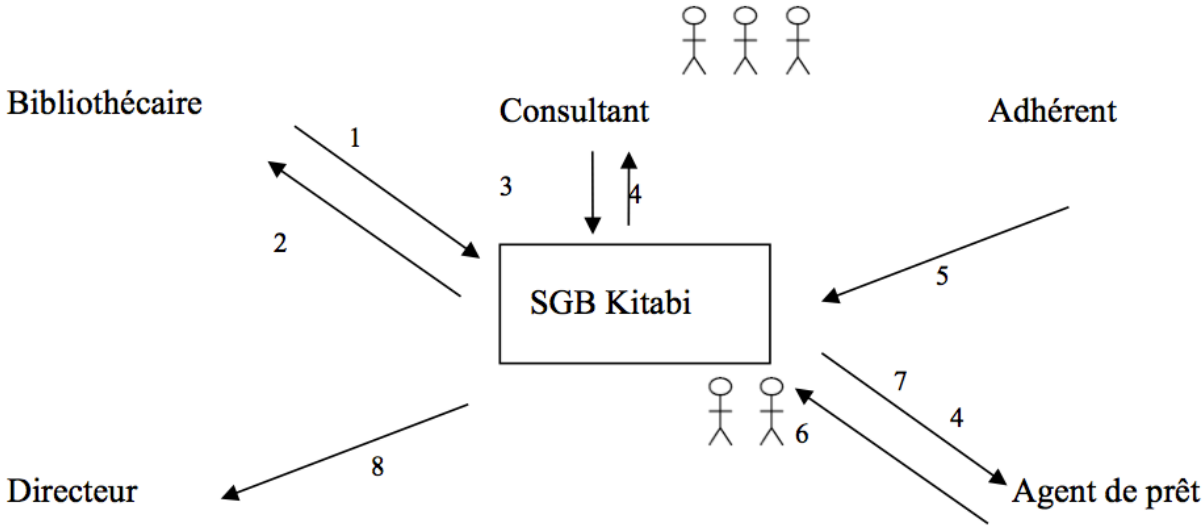


Figure 28. Exemple de diagramme de contexte dynamique

Numéro du message	Nature des messages
1	Information sur les livres Informations sur les agents de prêt
2	Livres trop demandés Livres aux exemplaires insuffisants
3	Livres recherchés
4	Informations sur les livres et leurs disponibilités

5	Informations de réservation (livre, date de réservation) – annulation de réservation
6	Informations sur les adhérents – informations sur les prêts et restitutions – informations sur les comptes d'utilisateurs
7	Adhérents sanctionnés, informations détaillées sur les prêts
8	Taux de renouvellement, taux de prêt de livres

Tableau 3. Légende du diagramme de contexte dynamique

6. Conclusion

Dans ce premier chapitre, nous avons abordé le processus 2TUP par la présentation de la première étape. Pour illustrer cette étape, nous avons présenté un cas fictif d'une entreprise X qui veut mettre en œuvre un système de gestion du personnel. Cette première étape permet de définir le contour du système, d'identifier ses utilisateurs ainsi que les messages qu'ils envoient ou reçoivent à partir du système. Le chapitre suivant est consacré à l'étape de capture des besoins fonctionnels.

Chapitre 4 : Capture des besoins fonctionnels

1. Introduction

Par besoins fonctionnels on désigne l'ensemble des besoins liés au métier et domaine traités par le système, par opposition aux besoins technique, que nous verrons dans le chapitre suivant. Aussi, il faut se détacher à cette étape des besoins en interaction homme-machine (IHM) qu'il est toutefois possible d'intégrer dans d'autres étapes. Les étapes de capture des besoins fonctionnels sont décrites dans ce qui suit.

2. Identification des cas d'utilisation

Il s'agit dans cette étape d'identifier l'ensemble des cas d'utilisation à partir du diagramme de contexte dynamique de l'étape d'analyse préliminaire. Un cas d'utilisation regroupe un ensemble cohérent de messages (peut être réduit à un seul message) émis vers et/ou reçu du système.

L'erreur à éviter est de descendre trop bas dans l'identification des cas. Le cas d'utilisation ne doit pas être réduit à une seule transaction (dans le sens de manipulation informatique : ajout, modification, suppression). Il doit plutôt décrire une intention de l'acteur vis-à-vis du système en termes de changement d'état global et de bénéfice métier.

3. Identification des acteurs principaux et les acteurs secondaires

Un acteur principal est obligatoire pour un cas d'utilisation (au moins un). Il représente l'acteur concerné par l'intention fonctionnelle. Les acteurs secondaires sont des acteurs qui ne sont pas concernés directement par le cas, mais qui peuvent être sollicités pour la réalisation du cas. Exemple : l'agent administratif peut consulter le chef de service en cas de problème de traitement des informations personnelles d'un employé.

Exemple pratique : nous reprenons l'exemple du projet de gestion du personnel. Le tableau suivant liste les cas d'utilisations ainsi que les acteurs et messages émis et reçus.

Cas d'utilisation	**Acteur principal** ***Acteur secondaire***	**Messages émis et reçus**
Suivi de la fiche de l'employé	Agent administratif *Responsable de personnel*	Emet : informations détaillées des employés Reçoit : liste des employés
Suivi de l'assiduité	Agent administratif	Emet : information de pointage Reçoit : états mensuels d'assiduité (absences, présences)
Suivi des formations	Agent administratif	Emet : formations, inscriptions, évaluations, Reçoit : état de formations des employés
Suivi de carrière	Responsable de personnel *Agent administratif*	Emet : informations de recrutement, promotions, avancements… Reçoit : classement des employés, carrières…
Suivi des congés	Responsable de personnel *Agent administratif*	Emet : congés, arrêts de travail… Reçoit : états mensuels d'activités …

Tableau 4. Exemple de liste de cas d'utilisation

4. Description des cas d'utilisation

Durant cette étape, chaque cas d'utilisation sera décrit par l'intention (but) suivi de l'acteur dans l'exécution du cas et les actions élémentaires qu'il peut effectuer. La liste des actions peut ne pas être exhaustive.

Exemple de description textuelle d'un cas d'utilisation : la fiche ci-dessous décrit brièvement le cas d'utilisation suivi de la fiche de l'employé.

Suivi de la fiche de l'employé

But : maintenir les informations sur un employé à jour
Actions : ajouter, supprimer, mettre à jour les informations personnelles d'un employé.

Tableau 5. Exemple de fiche descriptive d'un cas d'utilisation

5. Elaboration du diagramme des cas d'utilisations

Dans cette étape, on va élaborer le diagramme des cas d'utilisation en utilisant les notations graphiques vues dans la partie 1 de ce cours.

Exemple: le diagramme suivant illustre les cas d'utilisation de la gestion de personnel.

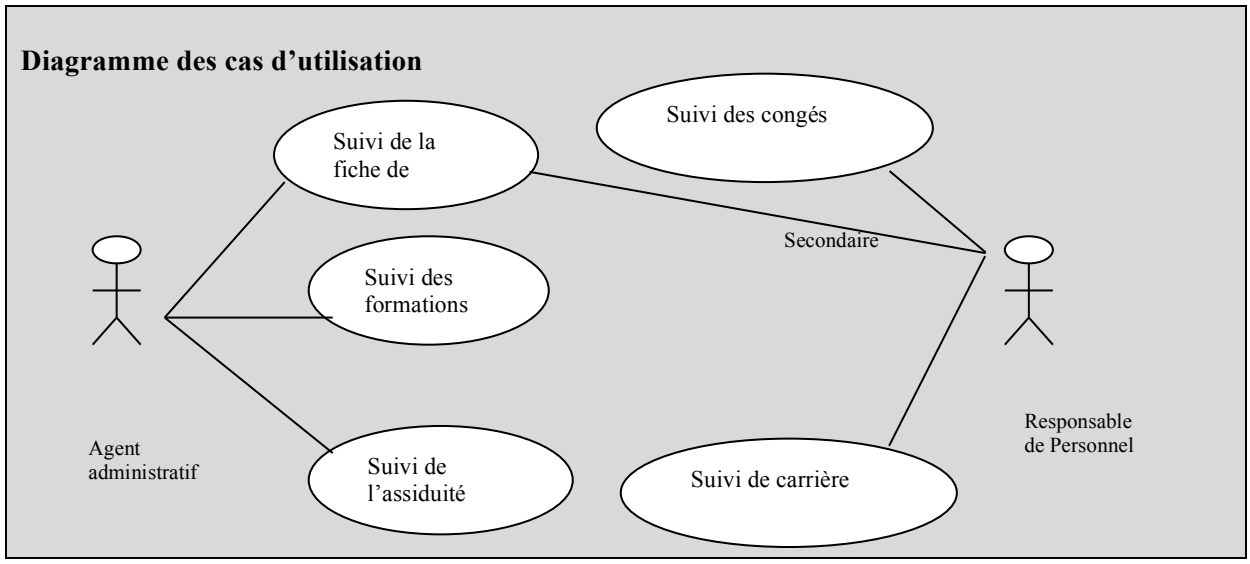


Figure 29. Exemple de diagramme de cas d'utilisation

6. Structuration des cas d'utilisation

Chaque cas d'utilisation peut faire l'objet d'une structuration avec une fiche descriptive. Le contenu de la fiche est laissé à l'analyste. Un exemple de fiche descriptive est le suivant.

Sommaire d'identification
Titre du cas d'utilisation : *le nom du cas d'utilisation* But : *l'intention ou valeur ajoutée attendue de l'exécution du cas* Résumé : *un bref résumé du cas* Acteurs : *l'acteur principal et les éventuels acteurs secondaires* Date de création : *date de création du cas* Date de mise à jour : *date de la dernière MAJ* Version: *version actuelle* Responsable : *créateur du cas ou responsable*

Description des enchaînements
Préconditions : *ce que doit être réalisé avant l'exécution du cas* **Enchaînements :** • Début du cas : *quand est-ce que commence le cas ?* • enchainements 1 : nom de l'enchainement o actions à effectuer o exceptions possibles pour les actions • enchainement 2 • … • Fin du cas : *quand est-ce que se termine le cas ?* **Exceptions :** • Exception 1 : *cas exceptionnel dans la réalisation d'une action* • Exception 2 • … **Postconditions :** *ce que doit être réalisé à la fin de l'exécution du cas*

Tableau 6. Fiche détaillée d'un cas d'utilisation

Exemple pratique : la fiche suivante décrit en détail le cas d'utilisation *suivi de la fiche employé*.

Sommaire d'identification

Titre du cas d'utilisation : suivi de la fiche de l'employé
But : maintenir la fiche des employés à jour
Résumé : ajouter un employé, supprimer un employé, modifier les informations sur un employé

Acteurs : agent administratif (principal), responsable du personnel (secondaire)

Date de création : 30/12/2008 **Date de mise à jour :** 30/12/2008
Version: 1.0 **Responsable :** B. Ahmed

Description des enchainements

Précondition :

- l'agent authentifié
- les informations à ajouter au système visées par le responsable du personnel

Enchaînements
Ce cas commence lorsque l'agent administratif reçoit les informations d'un employé nouveau ou existant.

Enchainement 1 : ajouter un employé
L'agent saisit les informations personnelles du nouvel employé
[Exception : 1 : l'employé avec le même nom et prénom existe déjà]
…
Ce cas se termine lorsque l'agent termine la mise à jour des informations sur les employés.

[Exception 1 : l'employé avec le même nom et prénom existe déjà] : dans ce cas, l'agent vérifie s'il s'agit bel et bien du même employé ou d'un homonyme. Dans le premier cas, l'ajout est annulé. Dans le second cas, l'employé est ajouté.

Postcondition

- les fiches des employés sont à jour

Tableau 7. Exemple de fiche détaillée d'un cas d'utilisation

7. Compléter la description des cas d'utilisation

Bien que la description textuelle des cas d'utilisation soit indispensable, il est toujours utile de la compléter par une description graphique sous la forme de diagrammes UML. Les différents diagrammes qu'on peut utiliser à ce niveau sont les suivants :

- ***Diagramme d'activités :*** c'est le digramme le plus recommandé. Il permet de montrer les enchainements d'un cas et aussi les enchainements parallèles.

- ***Diagramme d'états-transition :*** il est plus complexe à comprendre.

- ***Diagramme de séquences :*** il s'adapte à des scénarios particuliers. Comme le diagramme d'activités, ce diagramme permet de mieux illustrer les enchainements.

- ***Diagramme de collaboration :*** son pouvoir d'expression par rapport aux cas n'est pas aussi élevé que celui des diagrammes de séquence et d'activité.

Organisation des cas d'utilisation

A cette étape, on identifie les éventuelles relations entre les cas d'utilisation (inclusion, extension, généralisation/spécialisation).

7.1. Identification des relations d'inclusion

Les relations d'inclusion sont identifiées par factorisation des traitements communs à plusieurs cas. Un exemple de cela est l'authentification requise pour chaque acteur avant le début de toute utilisation du système.

7.2. Identification des relations d'extension

Les cas d'utilisation définis comme extensions à d'autre cas regroupent des traitements optionnels ou répondant à des conditions spécifiques. Un exemple de cela est l'extension de l'ajout d'une commande par l'ajout de produits.

7.3. Identification des relations de généralisation/spécialisation

Ce type de relation est identifié lors de l'existence de traitements spécifiques ou modifiés d'un cas ou de plusieurs cas par rapport à un traitement normal.

8. Structuration des cas d'utilisation

Dans un système ordinaire, on peut comptabiliser jusqu'à 20 cas d'utilisation. En général, certains cas d'utilisation sont fortement liés entre eux par rapport à d'autres. Cette liaison peut être caractérisée par un rapprochement temporel (traitement des commandes, gestions des clients), organisationnel (service clientèle) ou autre. De ce fait, il est possible de structurer les cas d'utilisation dans des packages.

9. Identification des classes candidates

Cette étape consiste à identifier la liste préliminaire des classes qui permettent de répondre aux exigences statiques (attributs) et dynamiques (opérations) de chaque ca d'utilisation. L'identification des classes est intuitive mais guidée par le contenu et description des cas d'utilisation. A ce niveau là, il n'est pas obligatoire de définir les attributs et opérations de chaque classe. Cependant, on peut décrire chacune des classes par une note contenant la responsabilité qu'elle joue.

Une responsabilité d'une classe est sa raison d'être. On l'exprime par le rôle joué par la classe avec un niveau de détail élevé par rapport aux attributs et opérations.

Exemple : la classe véhicule permet de (1) connaître ses caractéristiques, connaître son propriétaire et connaître sa disponibilité. Graphiquement, on représente cela comme suit.

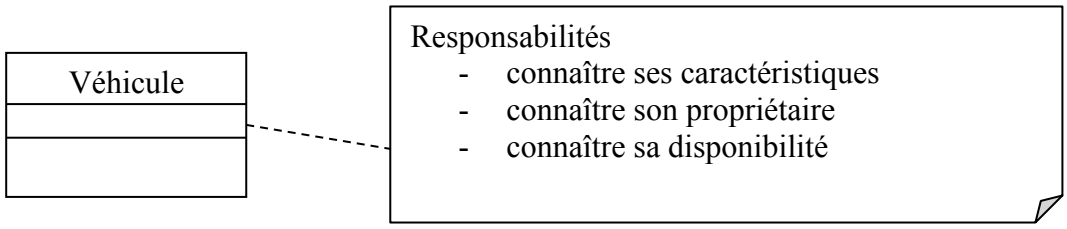


Figure 30. Exemple de responsabilités d'une classe

Une responsabilité peut contenir dans sa description les attributs de la classe, les opérations, mais aussi les associations. Cependant, lorsque deux responsabilités de deux classes donnent lieu à deux associations identiques, il faut représenter une seule association entre les deux classes.

Le diagramme de classes issu de chaque cas d'utilisation est appelé « diagramme de classes participantes ». A ce niveau là, on ne doit pas chercher l'ensemble exhaustif des classes, car d'autres classes peuvent réapparaître plus tard.

10. Validation et consolidation

Au terme de la capture des besoins fonctionnels, il convient, avant de passer à l'étape parallèle (capture des besoins techniques) de valider les résultats de cette étape avec les utilisateurs. La validation peut être guidée par un tableau comme suit

	Besoin 1	**Besoin 2**	**…**	**Besoin *m***
Cas d'ut. 1	X			X
...		X		X
Cas d'ut. *n*	X			

Tableau 8. Validation des besoins d'utilisateurs par les cas d'utilisation

Naturellement, l'analyse de ce tableau doit permettre de s'assurer que toutes les exigences du système sont prises en charge avec le cas d'utilisation. Faute de quoi, il faudra revenir sur les étapes précédentes autant de fois que nécessaire.

11. Conclusion

Ce chapitre a été consacré à la première étape réelle du processus 2TUP qui est l'étude préliminaire. Cette étape permet de traduire les besoins fonctionnels recensés dans l'étude préliminaire sous la forme de cas d'utilisation. C'est durant cette étape qu'on fait le passage être une expression en langue naturelle de ce que les utilisateurs veulent fonctionnellement et la manière de répondre à ces besoins. L'étape suivante est similaire à la capture des besoins fonctionnelle quoiqu'il soit possible de ne pas l'entamer qu'près avoir terminé les étapes de la branche gauche. Néanmoins, il est préférable de la présenter juste après la capture des besoins fonctionnels.

Chapitre 5 : Capture des besoins techniques

1. Introduction

La capture des besoins techniques traite la spécification logicielle et matérielle du système. Elle s'oppose ainsi à la spécification métier et fonctionnelle. Cette étape est du ressort des architectes techniques du système. Au préalable de cette étape, les architectes doivent disposer de minimum d'informations techniques, telles que l'environnement de développement, le matériel…Les étapes à suivre sont décrites dans ce qui suit.

2. Spécification technique du point de vue matériel

Le matériel à mettre en œuvre dans un système dépend de deux choses :

- L'architecture organisationnelle de l'entreprise. Celle-ci peut être à deux niveaux (central, local), à trois niveaux (central, départemental, local) ou multi niveaux dans une organisation plus complexe.
- Les besoins opérationnels : ces besoins peuvent se manifester de différentes manières dont (1) la sécurité (firewall, …), (2) la disponibilité (serveur haut de gamme), (3) la performance (cluster de machine), (4) l'interopérabilité (systèmes d'exploitation différents)…

Exemple : reprenons l'exemple du projet de gestion du personnel. La figure ci-dessous illustre la configuration matérielle à l'étape de capture des besoins techniques.

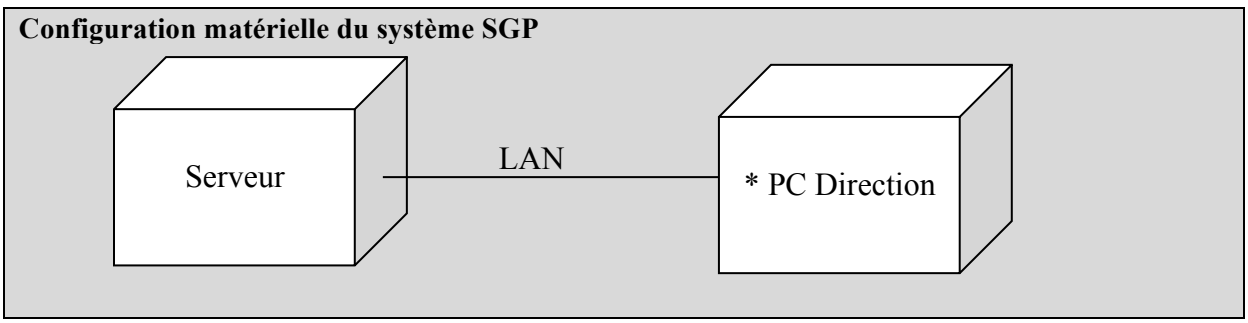


Figure 31. Exemple de spécifications techniques du point de vue matériel

3. Spécification d'architecture

En plus de la spécification matérielle, on procède à la spécification logicielle permettant de satisfaire l'architecture de fonctionnement choisie (par exemple Client / Serveur). Les éléments de spécification logicielle sont les suivants :

- **Composant d'exploitation**

Un composant d'exploitation est un élément logiciel (programme, fichier,…) indépendant de tout métier. Le choix des composants logiciel dépend de l'architecture choisie. Par exemple, lorsque le système fonctionnera en mode Client/Serveur, les principaux composants d'exploitation sont l'application et le serveur de base de données. Un composant doit avoir une autonomie de sorte à pouvoir le faire évoluer sans altérer le système, le dépanner ou le changer le cas échéant.

- **Composant métier**

Les composants métiers correspondent à des composants logiciels qui ont une vocation métier, et fournissent des services métier. Les composants métiers conviennent aux architectures 3-tiers, dans lesquelles un middleware applicatif d'interpose entre les serveurs de bases de données et les clients.

Un exemple de composants métier sont les classes qui traversent plusieurs métiers de l'entreprise. Par exemple la classe Produit peut être utilisée pour les métiers Production, Clientèle, Vente…

Exemple pratique : Gestion du personnel de l'entreprise X.

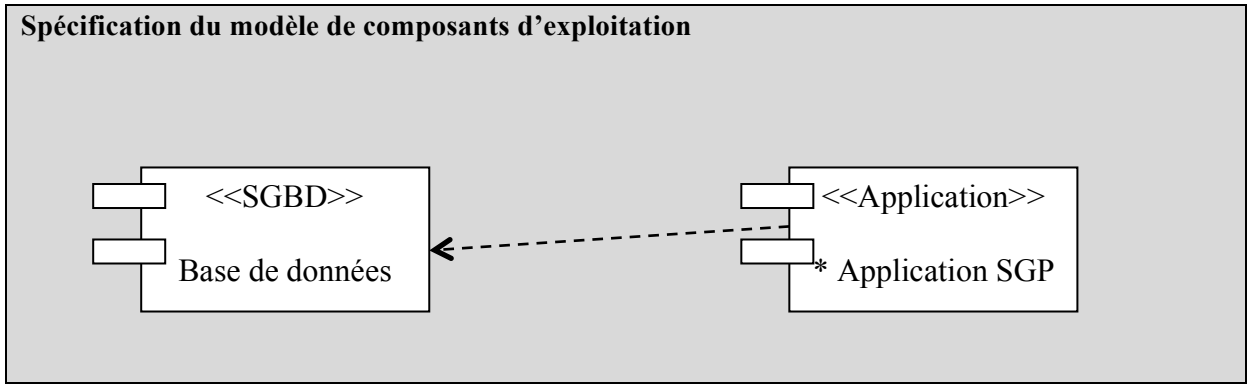


Figure 32. Exemple de spécification d'architecture

Dans ce schéma, l'étoile du coté du composant Application SGP signifie que plusieurs instances de l'application utilisent la même instance de la base de données. Ici, les deux instances de l'application sont utilisées respectivement par l'agent administratif et le responsable du personnel.

4. Elaboration du modèle de spécification logicielle

C'est à cette étape que l'on va se référer aux cas d'utilisation techniques. Un cas d'utilisation technique est un cas qui ne produit aucune valeur fonctionnelle (métier) mais fournit des services techniques à l'exploitant, c'est-à-dire, la personne qui utilise le cas. Des exemples de services techniques sont la connexion au système et l'utilisation de l'aide.

- L'exploitant : il s'agit d'un acteur du système au sens UML, qui bénéficie des services techniques du système. L'utilisateur classique d'une application est dans ce sens un exploitant car il bénéficie au moins du service de connexion à l'application.

- Le cas d'utilisation technique : tout comme les cas d'utilisation métier, les cas d'utilisation techniques sont spécifiques à chaque système. Toutefois, il est possible d'identifier, dans un premier temps, les cas les plus récurrents, comme la connexion au système, l'utilisation de l'aide, le débogage, le changement de mots de passe.

Exemple pratique : Gestion du personnel de l'entreprise X

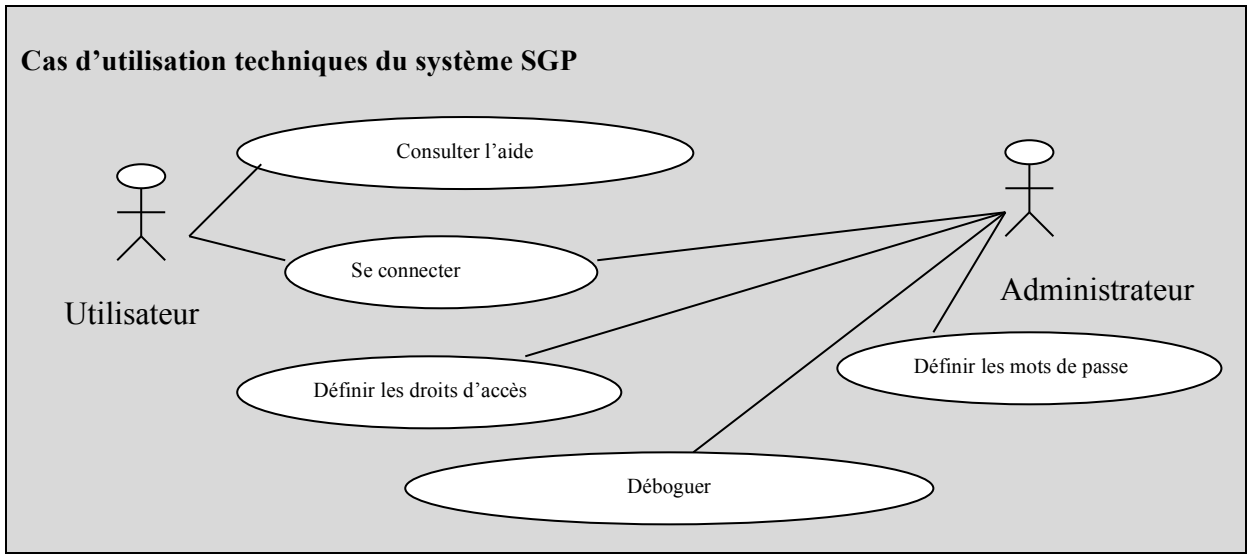


Figure 33. Exemple de modèle de spécification logicielle

5. Organisation du modèle de spécification logicielle

A cette étape, il s'agit de répartir l'ensemble des actions d'un cas d'utilisation technique sur des couches logicielles. Cela permet de mieux organiser les éléments logiciels en paquetages, et aussi de mieux les réutiliser.

Une couche logicielle définit un niveau contenant des spécifications logicielles homogènes. Les couches sont empilées les unes sur les autres. Cela est similaire aux couches réseaux du modèle OSI, dans le sens ou chaque couche offre des services à la couche adjacente.

Dans le cas d'un mode de fonctionnement client/serveur, les couches utilisées sont les suivantes : Présentation, Application, Métier, Accès aux données et Stockages de données. Un cas d'utilisation de l'exploitant entrainera une cascade d'actions (qu'on peut considérer comme des cas d'utilisation technique inter-couches) qui sont reparties sur les différentes couches. D'autres modèles de spécification logicielle peuvent être spécifiés comme le modèle MVC.

6. Conclusion

Dans ce chapitre, nous avons présenté la capture des besoins techniques qui permet de traduire les besoins opérationnels de l'étude préliminaire en cas d'utilisation techniques. Aussi, cette étape permet de traduire les besoins matériels et logiciels des utilisateurs en modèles de spécifications matérielle et logicielle, représentant la première ossature de la solution informatique qui sera détaillée dans les étapes suivantes.

Chapitre 6 : Analyse

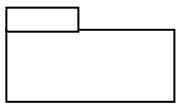
1. Introduction

L'étape d'analyse est entamée à la suite de l'étape de capture de besoins fonctionnels. A la différence avec les approches classique (ex : Merise), l'analyse cible le système futur plutôt que le système existant. Elle est située sur la branche gauche du processus 2TUP.

2. Découpage en catégories

La notion de catégorie est l'application du concept de package UML pour regrouper des classes identifiées dans les diagrammes de classes candidates. Une catégorie contient un ensemble de classes fortement couplées. Le couplage entre classes est issu de plusieurs situations (associations, agrégation, composition, héritage ou échange de messages).

Le découpage en catégorie permet une meilleure représentation de la structure statique du système. Il permet en effet de répartir les responsabilités des catégories sur plusieurs équipes de développement, de réutiliser les catégories dans d'autres systèmes et de faciliter l'évolutivité, la réutilisation et la maintenance des composants du système.

Une catégorie est représentée graphiquement comme un paquetage 

Critère du découpage : le point de départ est l'ensemble des diagrammes de classes. Il convient de regrouper les classes en catégories. Les critères de découpage peuvent être

- la cohérence des classes : dans le sens ou les classes sont liées sémantiquement
- l'évolution : en regroupant les classes a fort degré d'évolution ensemble et vice-versa
- la durée de vie des objets : on regroupe des classes dont les objets ont des durées de vies proches
- la dépendance entre les classes : la notion de dépendance est décrite plus loin

Exemple : dans le système de gestion du personnel, on peut identifier les catégories suivantes (un autre découpage est possible selon d'autres critères).

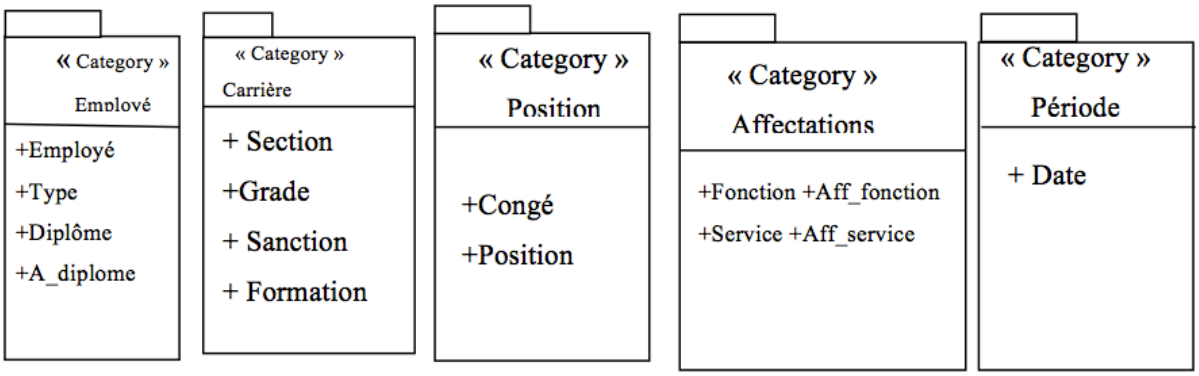


Figure 34. Exemple de découpage en catégories

- *Notion de dépendance entre les catégories* : lors du découpage en catégorie des classes, il est naturel de se retrouver avec des liens entre des classes de catégories différentes. Ces liens traduits par des associations ou autres doivent être conservés. On utilise pour ce faire la notion de dépendance entre les catégories.

La notion de dépendance est liée à la notion d'utilisation de classes en dehors de leurs catégories d'origine. Lorsqu'une association existe entre deux classes A et B de catégories différentes, si à partir de la classe A on veut accéder à la classe B, cela se traduit par une dépendance entre les catégories. La catégorie de A dépend alors de la catégorie de B. cependant, l'utilisation des classes d'une catégorie en dehors de sa catégorie est conditionnée par sa visibilité à l'extérieur de la catégorie qui doit être publique. Dans ce cas, le nom de la classe est précédé par le symbole +. Par contre, lorsque la classe est précédée de – elle n'est pas visible en dehors de sa catégorie. L'utilisation de classe à partir dans les catégories autres que celle d'origine est symbolisée par le stéréotype <<import>>.

Exemple :

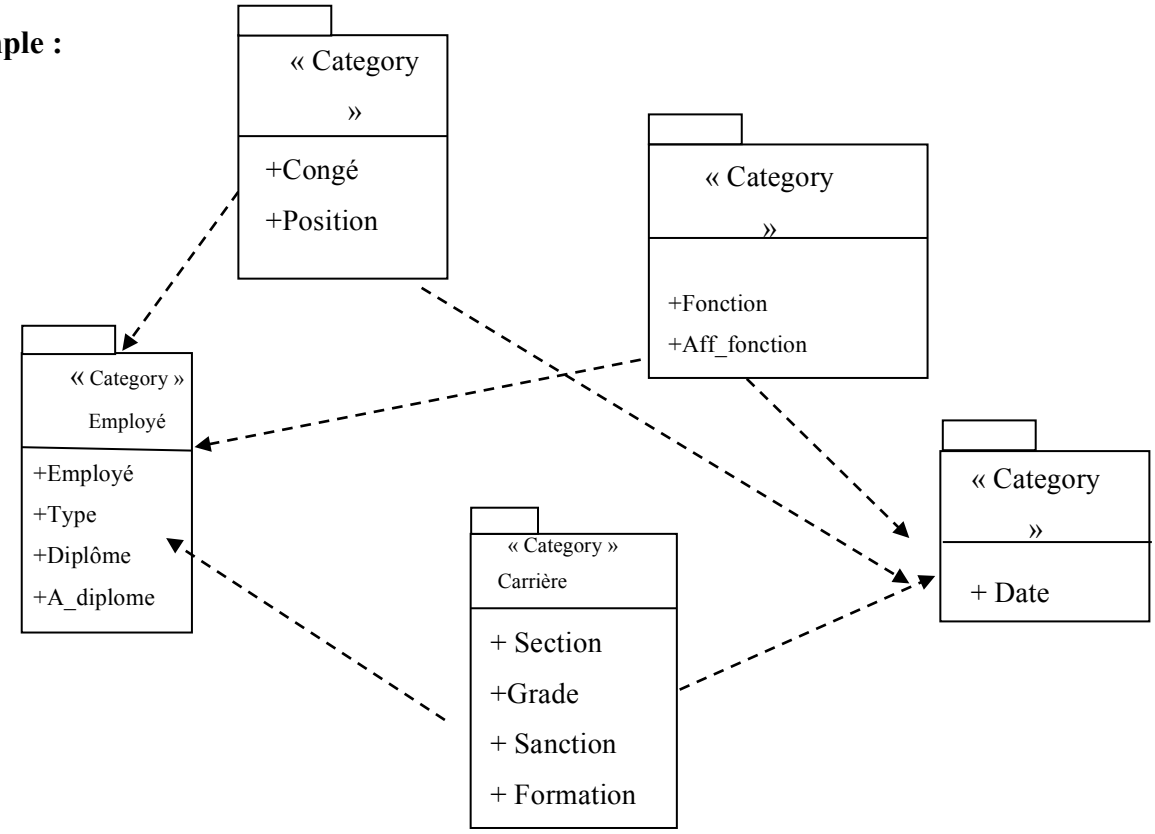


Figure 35. Dépendances entre les catégories

Remarque : les dépendances entre les catégories ne sont pas transitives.

3. Développement du modèle statique

Le découpage en catégorie génère un regroupement et répartition des classes candidates. Ces classes ne sont pas raffinées et ils convient de les examiner de près. C'est l'objectif de cette étape.

3.1. Traitement des anomalies

L'analyse de près des classes a pour objectif d'éliminer les anomalies. Ces anomalies se manifestent sous plusieurs formes :

Redondance, Classes à la place de rôle, Classe représentant des acteurs, Classe techniques (de conception), classe de groupes ou listes d'objets et classe grosses. D'autres concepts de modélisation sont à vérifier : multiplicités, navigabilité, associations ayant le sens d'agrégation ou de composition.
Aussi, il est possible d'ajouter des contraintes aux extrémités des associations, comme les contraintes d'ordre, d'ajout / suppression d'instance…ainsi que des stéréotypes aux classes et aux liens.

3.2. Traitement des attributs

Les attributs doivent être analysés pour en éliminer ceux qui traduisent des rôles. Aussi, on peut, pour expliciter des contraintes entre des propriétés, ajouter des attributs dérivés, en les faisant précéder du symbole /. Un attribut dérivé peut être calculé à partir des attributs de la même classe ou de classes différentes associées. Aussi, on prévoit à cette étape les attributs jouant le rôle de qualificatifs.

3.3. Traitement des opérations

L'identification de certaines opérations se fait par analyse textuelle du cahier de charges mais aussi à partir des cas d'utilisation.

Opérations particulières : certaines opérations décrivant des aspects logiciels peuvent apparaître au niveau du cahier des charges ou cas d'utilisation. Il faut les écarter à ce niveau là. Aussi, il faut écarter à ce niveau les opérations implicites sur les classes et les objets :

- création et destruction d'instances
- lecture et mise à jour des valeurs d'attributs
- création et destruction des liens entre les instances d'associations
- recherche des instances d'associations

Enfin, les noms des opérations doivent être abrégés.

Ajout de contraintes : outre les règles de calculs, le diagramme d'association peut être enrichi, à ce niveau par des contraintes. Exemple, on peut exprimer la contrainte {date de recrutement

> date de naissance} pour les employés dans le système de gestion du personnel. Les contraintes peuvent être écrites en langue naturelle ou en OCL.

4. Développement du modèle dynamique

Le développement du modèle dynamique se fait en parallèle que le développement du modèle statique. Il concerne le détail de la description des cas d'utilisation par d'autres diagrammes, à savoir ; les diagrammes de séquences et les diagrammes de collaboration.

4.1. Notion de scénario

La notion de scénario découle du fait que les enchaînements d'un cas d'utilisation ne sont pas exécutés tous à la fois, quoique l'ensemble des enchaînements constitue le cas complet.

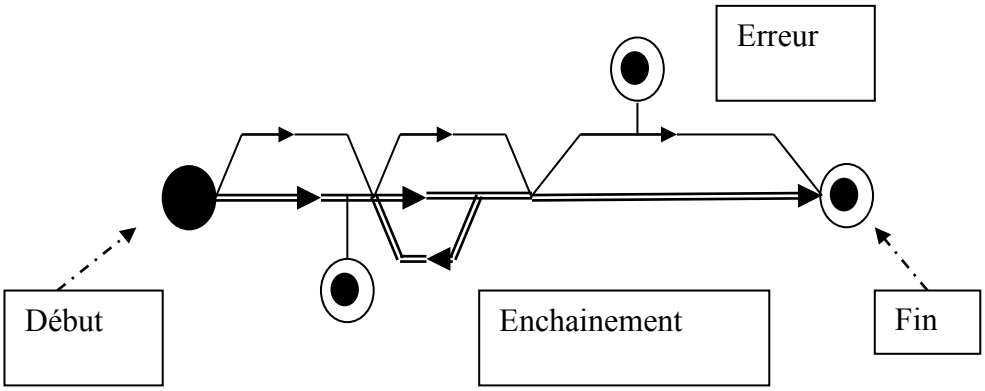


Figure 36. Notion de scénario d'un cas d'utilisation

Un ensemble d'enchainements ordonnées et exécutées d'emblée est appelé scénario. Il s'agir d'une configuration ou d'une exécution particulière du cas d'utilisation. Toutefois, il est possible de différencier les scénarios selon leur affectation du système, c'est-à-dire la manière dont ils aboutissent aux post-conditions du cas d'utilisation. Les types de cas d'utilisation sont les suivants :

- *Les scénarios nominaux :* il s'agit de scénarios qui permettent d'atteindre la fin du cas (et en conséquent réaliser la post-condition) de manière naturelle.
- *Les scénarios alternatifs :* il s'agit d'enchainement qui aboutissent à la fin du cas mais dans des situations rares car il s'agit d'alternatives en face à des exceptions du traitement normal.
- *Les scénarios aux limites :* ces scénarios définissent les limites d'exécution du cas de telle sorte qu'une autre exécution du cas provoquerait une erreur.
- *Les scénarios d'erreur :* il s'agit de sortie anormales du cas d'utilisation qui n'aboutit pas aux post-conditions.

Exemple: prenons le cas d'utilisation de traitement des informations personnelles d'un employé. Ce cas inclut l'affectation des différents diplômes de l'employé. Il s'agit dans ce cas d'ajout d'employé et de la modification d'informations le concernant.

- scénarios nominaux

    - création de l'employé
    - affectation d'un diplôme
    - consultation des informations sur un employé

- scénarios alternatifs : modification des informations sur un employé

- scénarios aux limites : affectation du dernier chauffeur à une livraison dans la gestion des commandes
- scénarios d'erreur : non validation de l'ajout de l'employé à cause de doublons dans les noms et prénoms de l'employé

4.2. Formalisation des cas d'utilisation

4.2.1. Notion de message

La notion de message est centrale dans la formalisation d'un scénario car un message est l'élément échangé entre les objets (y compris les acteurs). Un message est une spécification de communication entre les objets. Cette communication englobe l'ensemble des paramètres valorisés passés de l'émetteur vers le récepteur. Il existe deux catégories de messages :

- les signaux : il s'agit de messages asynchrones ne nécessitant pas de retour d'informations vers l'émetteur à partir du récepteur
- les appels d'opérations : il s'agit de messages synchrones qui nécessitent un retour d'informations vers l'émetteur à partir du récepteur

4.2.2. Situations exceptionnelles

- Messages destinés à un groupe d'objets : dans certains cas, il est possible concerne plusieurs instances d'objets. Par exemple, lors de l'affectation des diplômes à un employé, plusieurs instances de diplômes sont concernées.

- Messages de création / destruction d'objets : les instances auxquelles sont destinés les messages ne sont pas initialement pré-existant. Dans ce cas, un message particulier permet de créer l'instance. Cette opération peut être spécifiée par le stéréotype « create ». De même, un envoi de message peut être la destruction d'une instance d'objet.

- Les auto-messages : dans certaines situations, un message peut s'auto envoyer un message. Par exemple, on peut l'ajout de la date de recrutement d'un employé peut provoquer un message de demande de comparaison de la date de recrutement avec la date de naissance et peut produire une exception.

4.2.3. Les différentes formalisations possibles

Les diagrammes qui interviennent dans la formalisation des cas d'utilisation sont les quatre diagrammes vus dans la partie une du cours, à savoir le diagramme de séquences, le diagramme de collaboration, le diagramme d'états-transition et le diagramme d'activité. Ci-

après quelques recommandations relatives aux diagrammes qui accompagnent les cas d'utilisation :

- **Diagrammes d'états-transitions**

    - Diagrammes facultatifs. Pas toutes les classes sont concernées par ce diagramme mais seulement celles qui ont un comportement dynamique complexe.
    - Au moins trois états par diagrammes
    - Les événements induisant des changements d'état peuvent être (1) la réception de signal (2) la réception d'opération (3) le temps (on utilise le mot-clé *after*) et (4) la satisfaction d'une condition (on utilise le mot-clé *when*).

- **Diagrammes de séquences**

    - Recommandé.
    - Utiliser des notes pour spécifier les exceptions. Les notes peuvent être dessinées en marge et au même niveau que la flèche décrivant l'exception
    - Utiliser les auto-messages sous la forme de flèche réflexive partant et revenant au même couloir de l'objet en question.

5. Conclusion

L'analyse est une étape importante qui permet de développer les modèles statiques (diagrammes de classes) et les modèles dynamiques (diagramme des cas d'utilisation). Pour le modèle statique, il s'agit de corriger les anomalies avant de compléter les diagrammes dans les étapes suivantes. Pour le modèle dynamique, le système est remplacé par l'ensemble des classes manipulées par le cas d'utilisation. Dans le chapitre suivant, nous basculerons encore une fois vers le côté droit du processus afin d'aborder la conception générique du système.

Chapitre 7 : Conception générique

1. Introduction

La conception générique consiste à développer la spécification technique proposée durant la capture des besoins techniques tout en restant indépendant des besoins fonctionnels. Dans ce qui suit, nous décrivons les éléments de base nécessaires à la construction des modèles de la conception générique.

2. Classes, frameworks techniques et interfaces

La notion de classe technique décrit une classe UML dont la responsabilité est technique, par opposition aux classes fonctionnelles. Ces classes peuvent être réutilisées quel que soit le domaine ou l'étape du processus métier modélisé.

Exemples de classes techniques

- la classe Tuplet qui représente un enregistrement dans une table relationnelle
- la classe fenêtre qui est liée à la présentation

La notion de framework découle du fait que généralement une classe technique n'est pas utilisée seule, mais dans un ensemble de classes. Ce regroupement de classes techniques est appelé framework. Associé à la notion de framework est la notion d'interface. Une interface définit un ensemble d'opérations abstraites qui définissent les services offerts par une classe ou par un composant.

Une interface est réalisée (ou implémentée) pour donner lieu à des classes concrètes. Un ensemble d'interfaces interdépendantes définit un framework abstrait. Par contre, un ensemble d'interfaces concrètes définit un framework concret.

3. Elaboration du modèle logique de conception

Le modèle logique de conception est construit à partir des classes techniques. Les éléments d'un diagramme de classe vus dans la première partie de ce cours reviennent ici (classes, associations, agrégations, etc.) avec la différence qu'ici, l'intérêt porte sur les classes techniques.

Les classes techniques et frameworks sont définis selon le modèle de couches logicielles adopté durant l'étape de capture des besoins techniques. Si on reprend le modèle à 5 couches, on retrouve les couches suivantes avec les différents objets qui peuvent y figurer :

3.1. Couche présentation

Cette couche contient les objets (et donc les classes) nécessaires à la présentation. On y trouve les concepts fenêtre, panneau, bouton, …

3.2. Couche application

Cette couche contient les objets nécessaires au contrôle et pilotage de l'application. On y trouve les concepts de modèle, listes d'objets…

3.3. Couche métier

Dans cette couche, on retrouve les objets métier avec les règles de gestion associés. On y retrouve les concepts d'objet métier, objet composite …

3.4. Couche d'accès aux données

A ce niveau, on prévoit des objets qui gèrent l'accès aux données. On peut à ce niveau prévoir une classe technique qui récupère les données pour chaque classe métier comme l'employé, les diplômes, …et une classe Tuplet qui porte les valeurs d'une ligne d'une table physique.

3.5. Couche de stockage de données

La couche de stockage concerne le stockage physique de données. On y retrouve les classes Table et Clé étrangère …

Exemple : prenons l'exemple de gestion des droits de lecture/écriture des utilisateurs dans des tables du système. Chaque utilisateur peut avoir trois droits : lecture, écriture et lecture/écriture. On peut définir ce besoin à différents niveaux de spécification

Au niveau de la couche présentation : un panneau renseigne des caractéristiques de l'utilisateur et un autre recense les tables sous la forme d'une liste, avec pour chacune les droit associés. On aura donc le schéma ci-dessous.

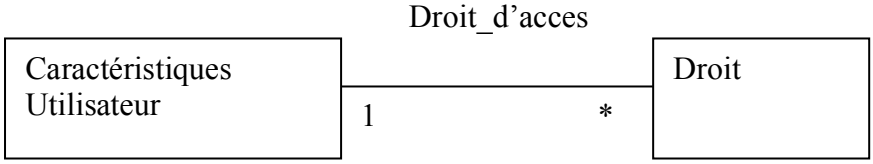


Figure 37. Exemple de classes techniques de gestion de droits d'utilisateurs

Par contre, au niveau de la couche de stockage, on trouvera trois tables relationnelles (si le

choix physique porte sur le modèle relationnel): utilisateur, table_app et utilisateur_droit qui recense les droits de chaque utilisateur sur chaque table.

4. Conception dynamique d'un Framework

La notion de framework implique de décrire l'aspect statique composé des classes techniques. Cependant, il peut être nécessaire de représenter aussi la dynamique d'un framework. Cette dynamique peut être décrite en UML à l'aide des diagrammes de séquences ou de collaboration si le framework nécessite une synchronisation entre les classes, ou à l'aide de diagramme d'état s'il s'agit d'une seule classe concernée par les changements d'états.

5. Organisation du modèle logique de conception technique

Tout comme le modèle qui structure les classes métier en catégories, le modèle logique de conception générique structure les classes techniques en packages qui correspondent aux frameworks. Par ailleurs, les besoins de séparer les différentes responsabilités du logiciel en couches nécessitent d'avoir des frameworks dont les responsabilités de chacun sont liées à une seule couche logicielle. Toutefois, certains framework ont une transversalité par rapport aux couches logicielles, dans ce cas, ils seront liés aux autres frameworks des couches logicielles.

Exemple : si l'on adopte toujours l'architecture à base de 5 couches logicielles, il en découle 5 frameworks techniques (présentation, application, métier, accès aux données, stockage de données) auxquels ont peut ajouter d'autres frameworks comme l'authentification, et la sauvegarde périodique de données. Le modèle d'organisation des frameworks est illustré comme suit.

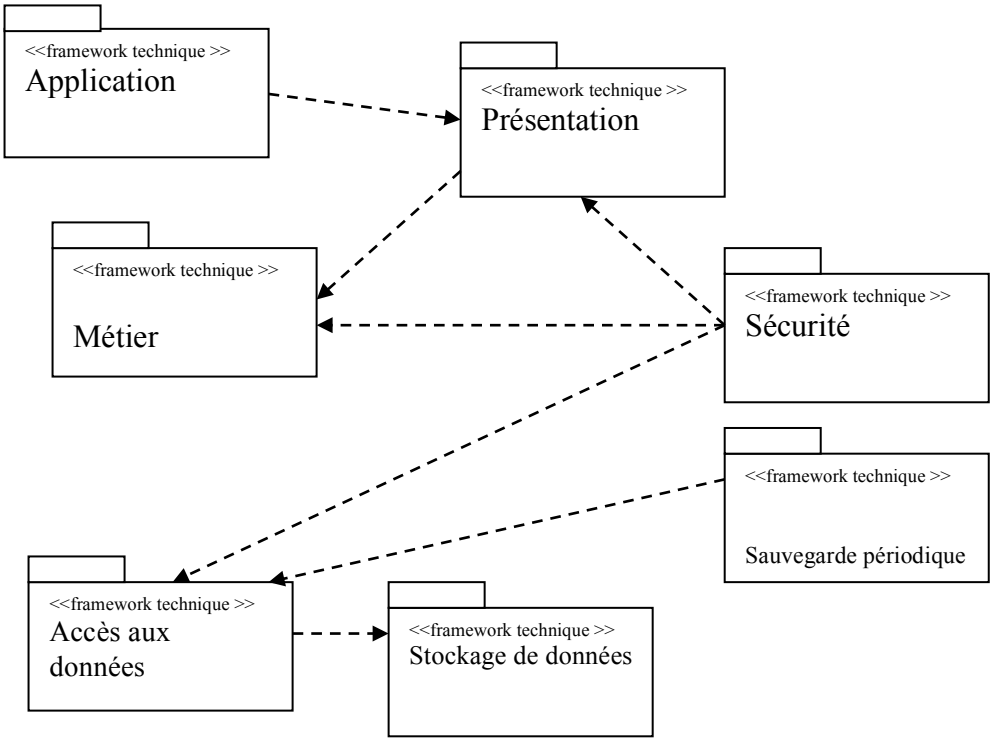


Figure 38. Exemple de modèle d'organisation de frameworks techniques

Les flèches en pointillé expriment des liens de dépendances. On remarque que l'utilisation du stéréotype <<framework technique>> et aussi les framework qui dépendent de plusieurs couches logicielles (frameworks « sécurité » et « sauvegarde »).

6. Elaboration du modèle d'exploitation de la conception technique

Le modèle d'exploitation reprend la notion de composant d'exploitation vue dans la spécification des besoins techniques, mais les détaille avec les composants nécessaires à configuration logicielle.

Pour reprendre l'exemple du système SGP, il s'agit d'inclure les composants nécessaires à la sécurisation de l'application et aux opérations de sauvegarde. Le cas de la sécurité nécessite de reposer sur un composant de base de données du même type que celui qui gère les données métier, tout en partageant le même composant qui représente le moteur de base de données.

Pour la sauvegarde, s'il l'on opte pour une sauvegarde sur un fichier externe, le composant d'exploitation « fichier » apparaitra. Nous aurons ce schéma illustratif des composants d'exploitation.

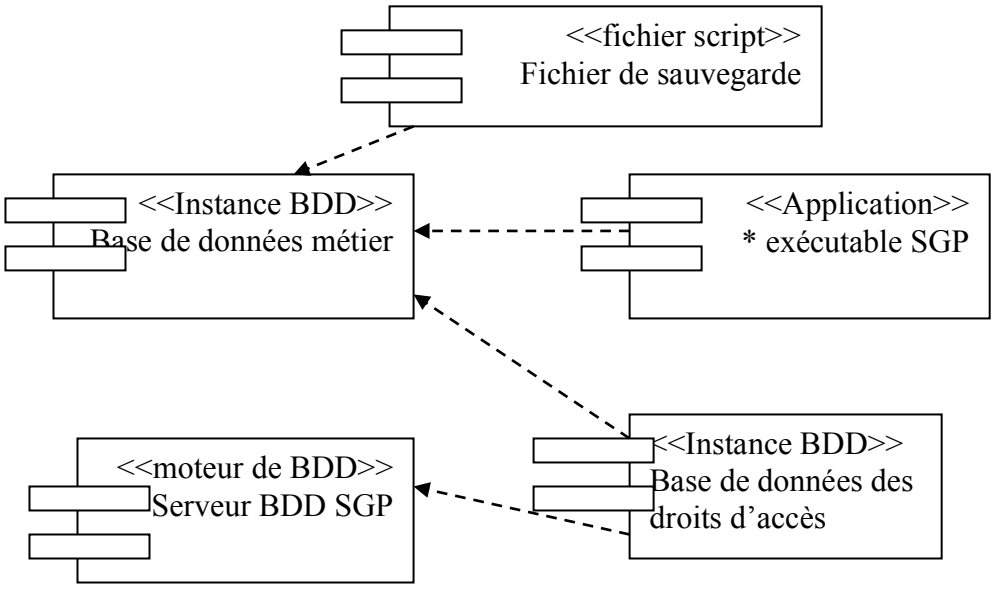


Figure 39. Exemple de composants d'exploitations

7. Conclusion

La conception générique est une étape importante dans laquelle les classes techniques du système sont développées, et ce, malgré que, de nos jours, les environnements de développement *clé en main* implémentent directement ces classes. La conception générique permet de détailler le modèle de spécification logicielle en détaillant les classes techniques qui serviront pour la conception des classes d'analyse. Ces classes sont alors organisées en paquetages appelés frameworks techniques correspondant à chaque couche du modèle logiciel choisi. En outre, des frameworks supplémentaires peuvent s'ajouter au modèle logiciel pour prendre en charge des aspects spécifiques du système tels que la sécurité, l'archivage, etc

Chapitre 8 : Conception préliminaire

1. Introduction

La conception préliminaire consiste à tenir compte des besoins exprimés dans la phase de capture des besoins fonctionnels dans la conception du logiciel. Il s'agit donc d'adapter la conception générique en terme de modèle de composants et de configuration logicielles aux métier et domaine du système à développer. La conception préliminaire est une première étape de la fusion qui sera suivie par le détail de conception plus tard (chapitre suivant). La conception préliminaire passe par les étapes décrites ci-après.

2. Concevoir le déploiement

Le modèle de déploiement décrit durant la spécification des besoins techniques ne définissait pas des nœuds spécifiques au domaine. En effet, à cette étape là, on introduit la notion de poste de travail. Un poste de travail représente un ou plusieurs acteurs localisés sur une machine de même type. Un poste de travail n'est pas réduit à une seule machine physique mais peut être matérialisé par plusieurs machines physiques à condition de répondre à la même configuration de déploiement.
D'autre part, le modèle de déploiement peut contenir des nœuds qui ne correspondent pas nécessairement à des postes de travail au sens métier mais peuvent être considérés comme des postes de travail au sens technique. Des exemples de cela sont les nœuds représentant les serveurs et les navigateurs.

Exemple : dans le système SGP, dans un déploiement client serveur, on peut prévoir deux postes de travail qui correspondent au responsable de personnel et à l'agent administratif. Aussi, le nœud « serveur » permet de stocker et gérer les données.

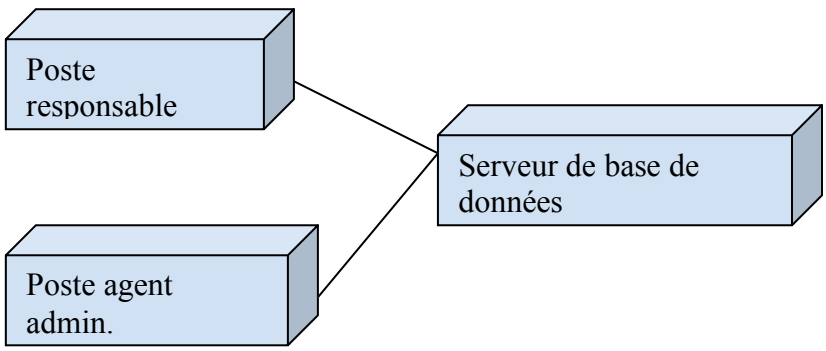


Figure 40. Exemple de diagramme de déploiement

3. Concevoir le modèle d'exploitation

Les modèles d'exploitation et de composants décris jusque là permettent de spécifier, en général, les composants et leurs répartition sur les postes de travail. Parmi ces composants, figurent les applications et les instances de bases de données. Hormis les composants techniques, les autres composants métier (application, instance de base de données métiers) n'ont pas été spécifiés. C'est à cette étape d'analyse préliminaire qu'on va le faire.

3.1. Découpage du système en applications

Le point de départ pour l'identification des applications est le regroupement des cas d'utilisation effectué durant l'étape de capture des besoins. Dans certains cas, un groupe de cas d'utilisation peut être affecté à un poste de travail. Ce poste de travail se verra assigner alors l'application qui réalise l'ensemble de ces cas d'utilisation. Par contre, on peut aussi trouver la situation où un cas d'utilisation est partagé entre plusieurs applications car exécuté par plusieurs acteurs sur des postes de travail différents.

Exemple : reprenons le diagramme des cas d'utilisation de SGP :

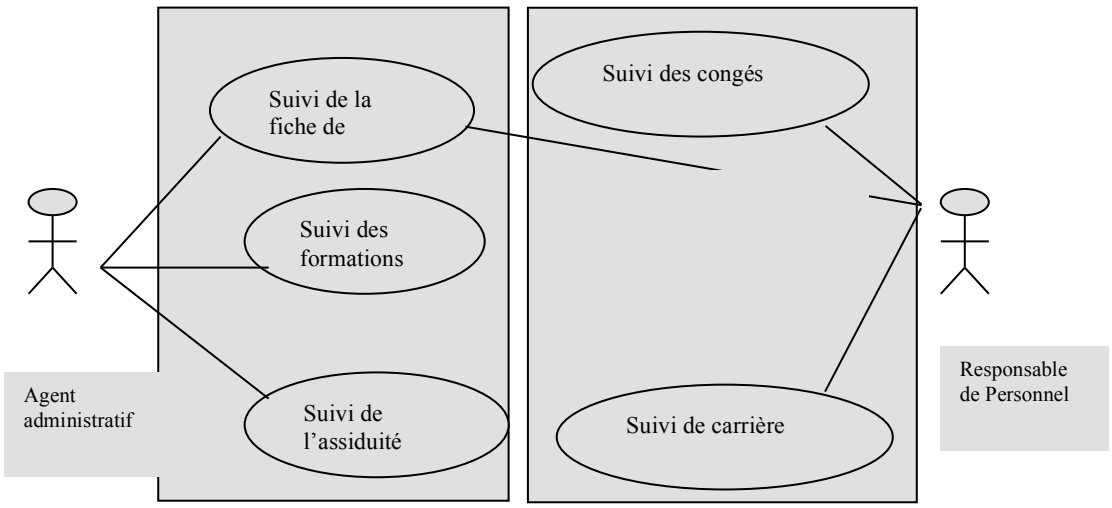


Figure 41. Exemple de découpage du système en applications

On remarque qu'il est possible de regrouper les cas d'utilisation en deux groupes, répartis sur les postes de travail *responsable de personnel* et *agent administratif*. On peut dès à présent tracer le schéma de répartition des applications sur les postes de travail. On appelle la première application Gest_CC (carrière et congés) la deuxième Gest_AP (Affectations et positions). Le schéma ci-dessous illustre la répartition des applications sur les postes de travail.

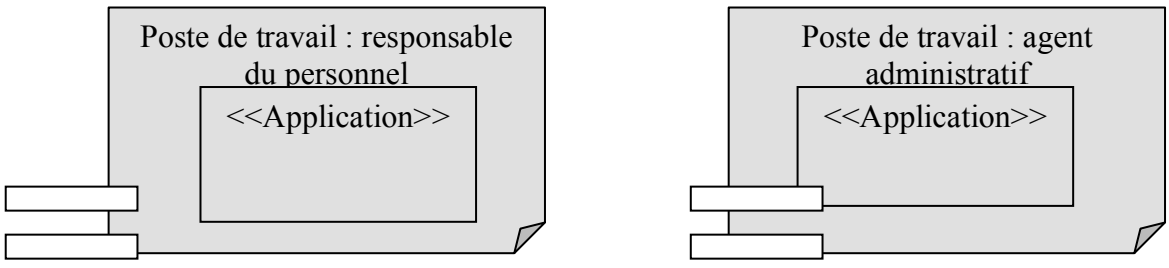


Figure 42. Exemple de postes de travail

3.2. Identification des composants distribués

Définition des composants : à ce niveau de conception, on s'intéresse aux composants métiers, issus du découpage en catégories. Puisque qu'une application reflète un regroupement de cas d'utilisation interdépendants, il en découle des classes regroupées en catégories. Par ailleurs, il est possible que certaines catégories soient partagées entre plusieurs applications. Par exemple, si on considère la catégorie Employé, on remarque qu'elle est partagée entre les deux applications. Dans ce cas, les catégories qui sont partagées entre plusieurs applications donneront lieux à des composants partagés et distribués. En outre, à cette étape là on affine la définition du composant d'instance de base de données. Le détail de ce composant dépend de l'architecture.

Exemple : dans le cas du système SGP, on peut distinguer la catégorie Employé qui peut être utilisée qui est partagée entre les deux applications voire avec plusieurs applications qui seront centrées sur l'employé plus tard. De ce fait, on peut ressortir le composant Employé comme composant métier à part. Les autres catégories donneront lieu chacune à un composant métier. En ce qui concerne la base de données, l'architecture du système étant en 2 tiers, une seule instance de base de données suffira. Le schéma suivant illustre le modèle d'exploitation comportant des composants métiers. Ici, le composant Employé est partagé et est constitué d'une seule catégorie d'analyse. Par contre, les composants Gest_CC et Gest_FFA ne sont pas partagés.

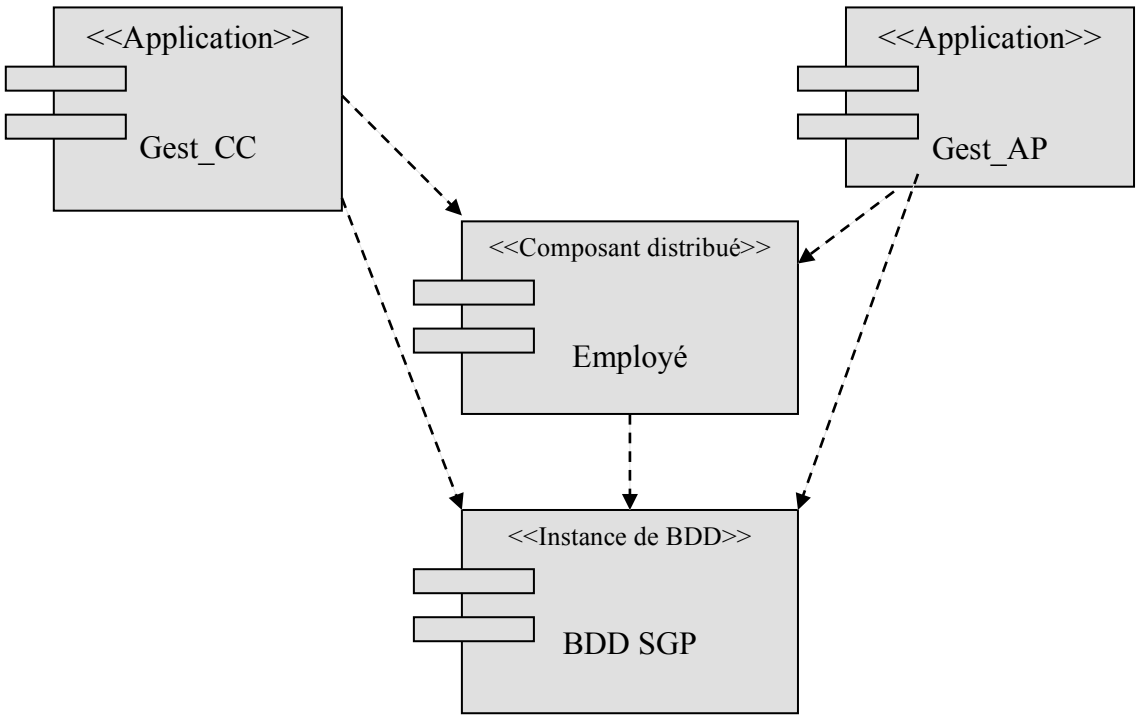


Figure 43. Exemple de composants distribués

L'étape suivante consiste à décrire les interfaces d'accès aux composants réutilisables. Ces interfaces sont identifiées à l'intérieur des catégories contenues dans chaque composant métier. Ainsi, chaque classe de catégorie d'analyse sera représentée par une interface. Chaque interface est décrite par ses responsabilités.

Exemple : dans le cas de SGP, le seul composant distribué étant Employé. Ce composant contient une catégorie qui contient pour sa part sept classes. Chaque classe donne lieu à une

interface. Le tableau suivant décrit ces interfaces avec des noms précédés d'un « I » ainsi que leurs responsabilités.

Composant distribué	**Interface**	**Responsabilité**
Employé	IEmployé	Distribuer les informations générales sur les employés
	IType	Association des employés à leurs types
	IDiplôme	Recensement des diplômes
	IA_diplome	Connaître et distribuer les fonctions

Tableau 9. Exemple d'interfaces

3.3. Enumération des interfaces utilisateurs (IHM) des applications

Par interface, nous signifions les interfaces de présentation des applications par opposition aux interfaces objet. La définition des interfaces objets peut être prévue dès l'étape de capture de besoins fonctionnels et étape d'analyse, mais c'est à ce niveau là que les maquettes peuvent être dessinées et distribuées sur les applications. La définition des interfaces liées aux applications permet, plus tard, d'identifier les objets de la couche présentation. Les IHM peuvent être décrits textuellement dans un tableau comme illustré dans l'exemple suivant du système SGP pour l'application SGP_FP. La liste n'est pas exhaustive.

Vue IHM	**Description**
Edition Employé	Editer les informations d'un employé. Création, modification, suppressions…
Sélection diplôme	Sélection d'un diplôme à assigner à un employé
Edition diplome	Ajout d'un nouveau diplôme avec les différentes informations
Sélection Employé	Sélection d'un employé à partir d'une liste pour effectuer des traitements le concernant
Période	Sélection d'une date à partir d'un calendrier
Sélection fonction	Sélection d'une fonction
Edition fonction	Edition d'une formations : création, modification…
Sélection service	Sélection d'un service
Edition service	Edition d'un service : création, modification …

Tableau 10. Enumération d'interfaces homme-machine

4. Concevoir le modèle logique

A cette étape, nous développons le modèle logique de conception en tenant compte d'une part des catégories d'analyse, et d'autre part, des framework techniques et des composant d'exploitation.

L'approche de conception du modèle logique consiste donc à procéder à un découpage en catégories de conception. Pour aboutir à ce découpage, on prend tour à tour chaque composant d'exploitation (applications, composants métier distribués, instance de base de données) d'une part, et chaque catégorie d'analyse d'autre part et de la projeter sur les framework abstraits qui représentent les couches logicielles.

Naturellement, certains composants d'exploitation en seront concernés que par certaines couches logicielles. Par exemple, l'application SGP_CC est concernée par plusieurs catégories d'analyse (Employé, Période, Position, Carrière). Par contre, elle sera concernée par les couches Présentation et Application. Par contre, le composant d'exploitation Instance de base de données, il sera quant à lui concerné par les couches Métier, Accès aux données, et Stockage de données. A l'issue de ce croisement, il sera possible d'énumérer les catégories au niveau de chaque couche logicielle, avec leurs associations aux composants du modèle d'exploitation.

5. Concevoir les structures de la présentation

Cette étape ne fait pas partie d'UML car concerne les choix ergonomique en matière de présentation de chaque composant concerné par la couche présentation. Cette étape permet de déduire les classes correspondant aux vues de la couche présentation comme les panneaux, les boutons…

6. Organiser la configuration logicielle

La dernière étape de la conception préliminaire consiste à réorganiser le modèle logique de conception constitué de catégories de conceptions en sous-systèmes. Initialement, chaque catégorie de conception donne lieu à un sous-système à développer. Cependant, il est possible d'isoler certaines classes en sous-systèmes afin de permettre la réutilisation.

Exemple : la sélection d'employés est utilisée à chaque fois qu'on veut faire un traitement à un employé. Les classes nécessaires à cette sélection (niveau application et application) peuvent être isolées en sous-système. Aussi, les catégories d'une même catégorie d'analyse relevant des couches application et présentation sont souvent associées car généralement, ces deux couches ne sont pas isolées.

Un sous-système est modélisé sous la forme de paquetage UML. 

Exemple : dans le système SGP, on aura les sous-systèmes Employé, selection_employé, Période, Diplôme…

7. Conclusion

L'étape de conception préliminaire constitue le premier point de rencontre entre la branche fonctionnelle et la branche technique du processus 2TUP. Dans cette étape, le modèle de configuration matérielle donnera lieu au déploiement des postes de travail. Aussi, le modèle d'exploitation est concrétisé par l'identification des applications de découpage du système et les composants distribués. Les aspects d'interface sont également traités par l'énoncé des interface homme machine de chaque application.

Chapitre 9 : Conception détaillée

1. Introduction

L'étape de conception détaillée est la dernière étape avant le codage. A cette étape, on commence à raffiner les choix de conception en incluant les spécificités du langage ou environnement de développement. Les étapes de conception détaillées sont présentées dans ce qui suit.

2. Concevoir le modèle logique

2.1. Concevoir les classes

La conception détaillée des classes est relativement systématique, cas les concepts de l'orienté-objet trouvent leur équivalence au niveau des langages de programmation, comme Java. Cependant, certains concepts ne sont pas explicités dans certains langages de programmation.

2.2. Concevoir les associations

Lorsque le concept d'association n'est pas pris en charge par le langage de codage, il convient de procéder à la transformation de l'association. Cette transformation dépend du type et des multiplicités de l'association.

Exemple : soit les classes représentées par le diagramme ci-dessous et la transformation qui lui est associés.

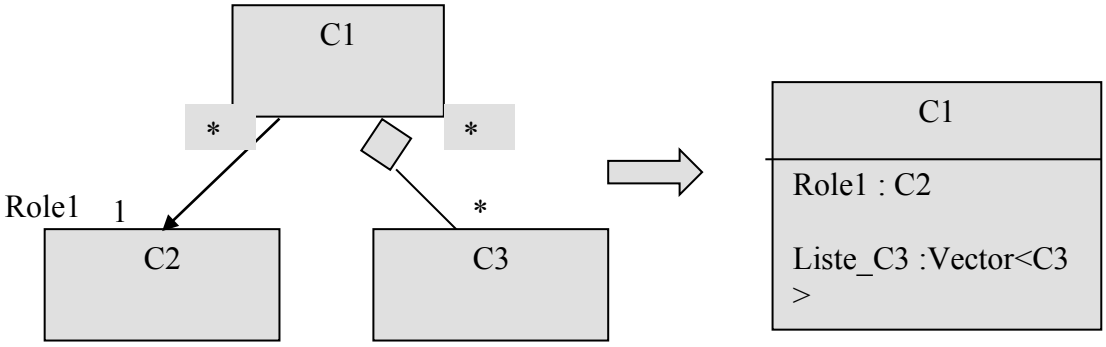


Figure 44. Exemple de transformation d'associations

La conception des associations mène à penser à plusieurs situations représentées dans le diagramme et devant être conservées lors de la transformation de l'association. Parmi les éléments à gérer figurent :

- les multiplicités supérieures à 1 (attributs obligatoires)
- les contraintes *lecture seule* et d'*ordonnancement*
- la gestion des associations de type composition lors de la destruction de l'agrégat
- les associations bidirectionnelles donnent lieur chacune à la copie des attributs et des opérations de part et d'autre. La synchronisation entre les deux classes en termes de valeur est de fait à gérer.
- la classe d'association se transforme d'abord en deux associations puis chaque association sera gérée de sa part par les mêmes principes d'une association normale.
- transformer les relations n-aires en relations binaires.

2.3. Concevoir les attributs

La conception des attributs concerne les opérations suivantes

- Typage avec des types scalaires ou complexes
- Prise en compte des contraintes de type lecture seule ou lecture écriture
- Conservation de la visibilité de
- Ajout, si nécessaire, des méthodes d'accès en lecture *set<nom d'attribut>* ou *get<nom d'attribut>*
- Spécifier les méthodes de mise à jour des attributs dérivés lorsque les attributs de base changent de valeur.

2.4. Concevoir les opérations

La conception des opérations consiste à détailler l'algorithme nécessaire à sa mise en œuvre.

3. Conception des couches logicielles

Les différentes couches logicielles vont subir une conception encore détaillée, que nous ne présentons pas dans ce cours. Toutefois, la couche de stockage mérite d'être étayée à ce niveau. En effet, le diagramme des classes métier est généralement mis en œuvre en relationnel, d'où la nécessité de procéder à une description des données en relationnel.

Le passage du modèle objet au relationnel suit un ensemble de règles illustrées dans le tableau suivant :

Modèle objet	Modèle relationnel
Classe	Table
Attribut de type simple	Colonne
Attribut de type composé	Colonnes ou clé étrangère

Instance	T-uplet
OID	Clé primaire
Association	Clé étrangère ou table de liens
Héritage	Clé primaire identique sur plusieurs tables

Tableau 11. Règles de passage du modèle d'objet au relationnel

4. Conclusion

La conception détaillée représente la dernière étape de conception qui précédé le codage et les tests. C'est durant cette étape que le modèle statique sera détaillé. Il s'agit de concevoir les classes selon l'environnement de mise en œuvre (développement sous Java, par exemple) et d'affiner le contenu des classes en termes d'attributs et d'opérations.

Durant cette étape également, on procède à la traduction du diagramme de classes vers le modèle relationnel vu que les SGBD relationnels sont les plus utilisées actuellement.

Conclusion Générale

Ce support de cours résume les activités de développement d'un système d'information selon le processus 2TUP qui est une concrétisation du processus unifié basé sur UML. Ces activités sont regroupées en étapes et sont organisés selon un processus en Y qui sépare les aspects fonctionnels des aspects techniques du système à développer. Ce support a été organisé, naturellement, selon l'ordre des étapes afin de mieux percevoir l'aspect incrémental du processus dans le sens que chaque étape apporte des incréments aux résultats des étapes précédentes.

Tout au long de la présentation du processus, nous l'avons accompagné d'un exemple représentant la gestion de personnel d'une entreprise typique. L'adoption d'un seul exemple avait comme objectif de fournir à l'étudiant une étude de cas complète. Toutefois, l'étudiant est averti du fait que l'adoption d'un seul exemple de système répond purement à un objectif pédagogique d'enseignement du processus. Les annexes fournissent une panoplie d'autre cas pour l'exercice du processus ainsi. L'assistance aux séances de travaux dirigés reste, toutefois, indispensable afin de maitriser le processus et d'éviter les erreurs dues à une mauvaise compréhension.

Ce support reste ouvert aux améliorations. Les critiques peuvent être destinées à l'auteur à travers le site Web du master SIAD de l'université de Jijel. Aussi, les quatre années d'enseignement de ce module ont permis de déceler les parties délicates qui nécessitent plus d'explications lors des séances de cours et plus d'attention au niveau du support de cours.

Références bibliographiques

[1] https://fr.wikipedia.org/wiki/G%C3%A9nie_logiciel

[2] R. Torlone. *Conceptual Multidimensional Models*. Multidimensional Databases, pages 69–90. 2003.

[3] J. Rambaugh et al, « OMT, Modélisation et Conception Orientées Objet». Ed. Masson 1995.

[4] H. Tardieu, A. Rochfled, R. Colletti. *La méthode Merise, principes et outils*. Eyrolles, 2000.

[5] Object Management Group. *Unified Modeling Language™ (UML®) Resource Page*. www.omg.org.

[6] D. Pilone, UML 2.0 in a nutshell.

[7] Wikipedia. *Unified Process*. http://en.wikipedia.org/wiki/Unified_Process.

[8] P. Roques, F. Vallée. *UML en action*. Eyrolles, 2002.

[9] P. Roques, F. Vallée. *UML 2 en action*. Eyrolles, 2004.

Annexe 1 : Séries d’exercices 2011-2014

1. Année universitaire 2011-2012

1.1. Série n° 1

On veut concevoir le système d'information d'un cabinet médical, en utilisant UML. Le cabinet est composé du médecin généraliste, d'un infirmier assistant le médecin dans diverses tâches (mesure de tension, injections, etc.). Une réceptionniste accueille les patients, enregistre les rendez-vous, et s'occupe de la gestion de la caisse. Le cabinet emploie une femme de ménage à temps partiel et un agent veilleur de nuit. Le système fonctionne comme suit [1]:

Traitement des rendez-vous

Les patients se présentent au cabinet ou téléphonent pour la prise de rendez-vous. La réceptionniste note le nom, prénom du patient ainsi que la date et heure du rendez-vous. Le rendez-vous possède le statut «non réalisé ». Le patient peut annuler le rendez-vous. Dans ce cas, le rendez-vous est maintenu dans le système mais aura le statut « annulé ». Le patient peut demander de reporter son rendez-vous. Dans ce cas, la réceptionniste en modifie la date et/ou l'heure. Le patient peut également ne pas se présenter au rendez-vous. Dans ce cas, la réceptionniste note cela en attribuant le statut « raté » au rendez-vous. Le dernier cas est celui où le patient se présente au rendez-vous. Dans ce cas, son statut est « réalisé ».

Traitement des visites

Le médecin tient un registre des visites sur lequel il note toutes les visites effectuées. Les visites sont numérotées séquentiellement. Le médecin tient aussi une fiche de chaque patient où il note le nom de ce dernier, son prénom, son sexe, sa date de naissances, les éventuelles allergies, les antécédents familiaux ainsi que les visites effectuées. Lorsque le patient se présente pour la première fois, on lui créé une nouvelle fiche. Cette fiche permet le suivi de près de chaque patient et elle est constamment modifiée pour ce qui est des informations générales ou des informations de suivi médical. Pour chaque visite, le médecin note sur le registre des visites la date, le nom et prénom du patient, le type de la visite (consultation ou contrôle). Il note également les différentes mesures effectuées (tension, poids, etc.). Une visite de type consultation est caractérisée par la constatation faite par le médecin sur l'état de santé du patient (diagnostic, symptômes, etc.). Pour une visite de type contrôle, on note l'évolution de l'état de santé du patient (état avant, état après).

Chaque visite donne lieu à une éventuelle prescription médicale. En plus du nom, prénom et âge du patient, la prescription contient l'ensemble des médicaments avec, pour chaque médicament, la quantité et le mode de prise (posologie, fréquence de prise, durée, etc.).

Travail à faire

1. Identifier les cas d'utilisation et illustrer par un diagramme de cas d'utilisation.
2. Elaborer le diagramme de classes correspondant à la description ci-haut.
3. Etayer le diagramme de classes par un diagramme d'objets. Les noms d'instances sont à proposer.
4. Identifier parmi les classes du diagramme élaboré en 3. celle(s) dont les objets sont sujets à des changements d'état. Elaborer les diagrammes d'état-transition.

[1] Le fonctionnement décrit ici est partiel. La gestion des payements n'est pas prise en compte.

Corrigé type série 1

- ***Identification des cas d'utilisation***

1.Acteurs : Médecin, Réceptionniste

2.Cas :

a.Gestion des RDV : inclut les scénarios ajout, report, annulation, réalisation

b.Ajout de personne

c.Consultation de personne

d.Gestion de visite: deux sous-cas dérivés :

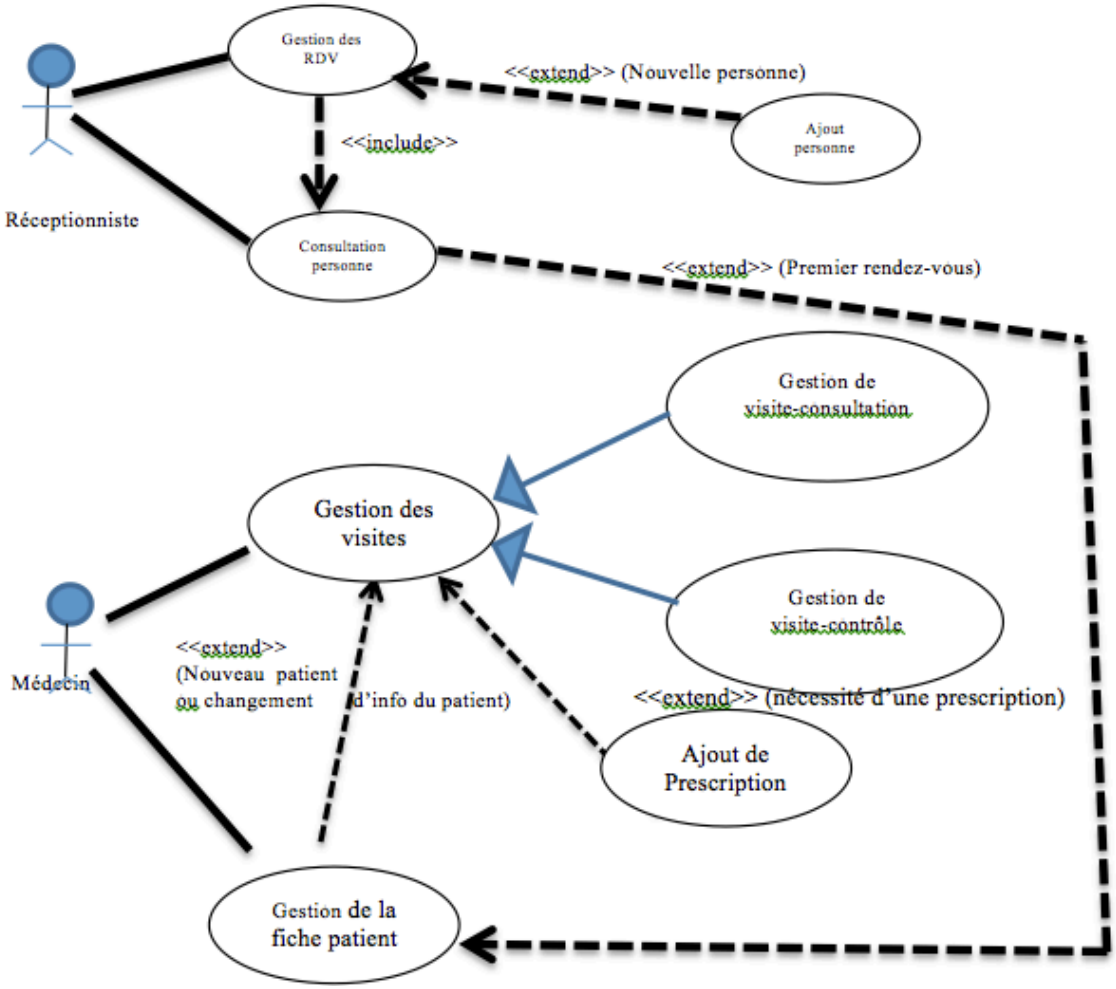
i.Gestion de visite-consultation.

ii.Gestion de visite-contrôle.

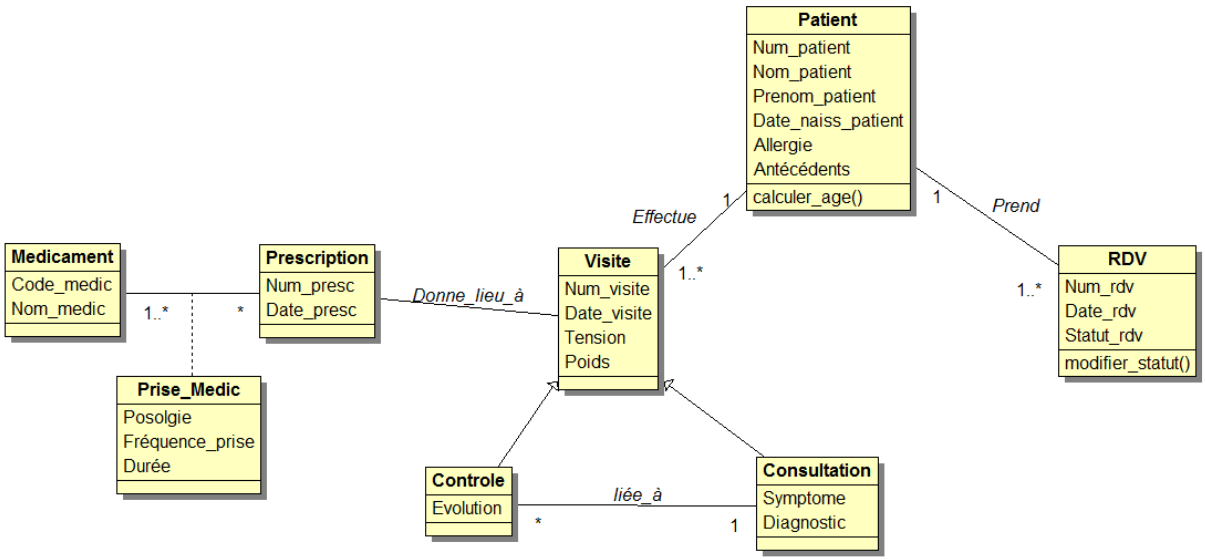
e.Gestion de la fiche patient : inclut les scénarios ajout patient, mise à jour patient

f.Ajout de prescription.

Le diagramme des cas d'utilisation

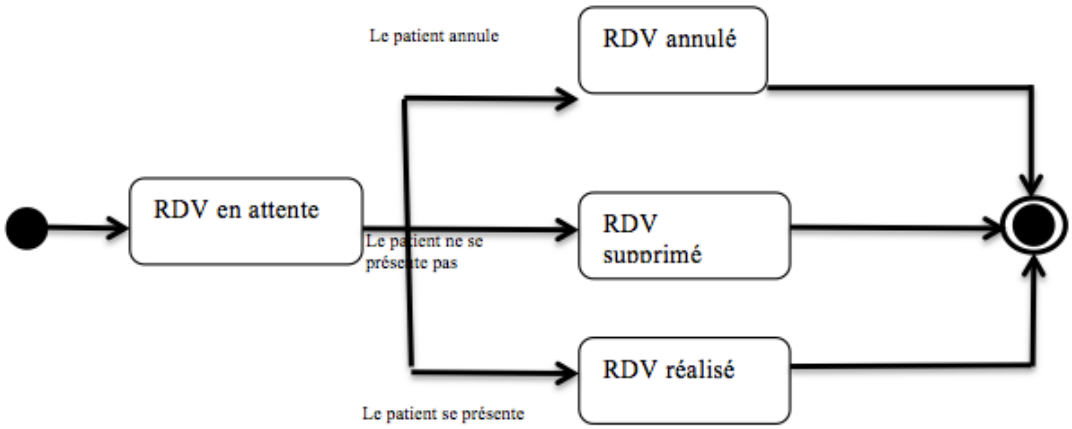


- ***Le diagramme de classes***



- ***Diagramme d'états-transitions***

La classe concernée par des changements d'état est la classe RDV. Le diagramme d'états-transitions est le suivant :



1.2. Série n° 2

Gestion de la bibliothèque *Kitabi*

La bibliothèque *Kitabi* est spécialisée dans le prêt de livres de différents types (livres scolaires, histoire, géographie, sport, etc.). Chaque année, le bibliothécaire dresse la liste des livres à acheter auprès des fournisseurs. Une fois les livres acquis, le bibliothécaire se charge de les classer selon le type, de leur affecter les côtes et de les ranger dans les rayons. Une fois les livres rangés, ils peuvent être prêtés. La bibliothèque prête les livres à des adhérents (prêts externes) ou permet la consultation interne pour les adhérents et pour les non adhérents sur présentation de la carte d'adhésion ou d'une pièce d'identité. Les agents de prêts, au nombre de

six s'occupent des prêts et des consultations. Pour chaque prêt, on note le numéro d'adhérent, la côte du livre ainsi que le numéro d'exemplaire et la date de prêt. Lors de la restitution, l'agent cherche le livre emprunté par sa côte et son numéro d'exemplaire, vérifie la date prévue de restitution et restitue le livre. Si la date prévue de restitution est dépassée, l'adhérent est privé de prêt pour une durée équivalente au double du nombre de jours de retard. Pour une consultation interne, l'agent note le numéro de la carte d'adhérent ou le numéro de la pièce d'identité, son type et le nom et prénom du consultant.

La bibliothèque est chapeautée par un directeur qui veille à son bon fonctionnement. Aussi, par souci de sa modernisation, il vous sollicite afin de lui mettre en place un système d'information pour gérer sa bibliothèque. Le directeur vise à travers le système d'information à faire connaître sa bibliothèque au grand public en permettant à toute personne de consulter par Internet la liste des livres et leur disponibilité. Il souhaite que cette consultation soit également possible à l'intérieur de la bibliothèque en déployant des ordinateurs dans le hall. Ainsi, une fois qu'un adhérent ou personne externe aurait identifié le livre souhaité et vérifié sa disponibilité, il s'adresse à l'agent qui effectue le prêt en le notant sur machine. On voudrait qu'un adhérent puisse réserver un livre indisponible et d'annuler la réservation. Aussi, on veut renforcer la gestion en sanctionnant automatiquement les retardataires dans les restitutions (ne pas permettre le prêt jusqu'à la fin de la période de sanction).

Par ailleurs, la recherche de livres ne requiert pas d'authentification. Par contre, pour pouvoir gérer la réservation d'un livre, l'adhérent doit s'authentifier avec un nom d'utilisateur et mot de passe. L'attribution des login /mot de passe est du ressort des agents lors de l'ajout de l'adhérent au système ou du renouvellement d'adhésion. Un mot de passe aléatoire est donné à l'adhérent à la fin de son inscription. Cependant, ce dernier doit être en mesure de modifier son mot de passe. Dans le cas de perte du mot de passe, l'adhérent s'adressera à l'agent de prêt pour le lui réinitialiser. De même, afin de garder une trace des opérations de prêts, chaque agent doit s'authentifier pour pouvoir gérer les prêts. L'attribution des coordonnées de connexion au système pour les agents revient au bibliothécaire. Le principe de changement et réinitialisation des mots de passe des agents est le même que celui des adhérents.

Le souci de satisfaction des adhérents et consultants est majeur, surtout par rapport à la disponibilité des livres. Pour cela, le bibliothécaire suit la gestion des prêts pour connaître les livres trop demandés, les titres souvent demandés mais indisponibles (à travers l'historique des recherches), les livres au nombre insuffisant d'exemplaires et ce, pour planifier les achats de livres. A un autre niveau, le directeur de la bibliothèque souhaite suivre l'évolution de l'activité de sa bibliothèque à travers différentes mesures (évolution du nombre d'adhérents, taux de renouvellement des adhésions chaque année, taux de prêts de livres, etc.).

Travail demandé

1. Identifier les besoins fonctionnels et les besoins opérationnels du cas ci-dessus ;
2. Identifier les acteurs et leur affecter les rôles ;
3. Modéliser le contexte du système à développer.

Corrigé de la série 2

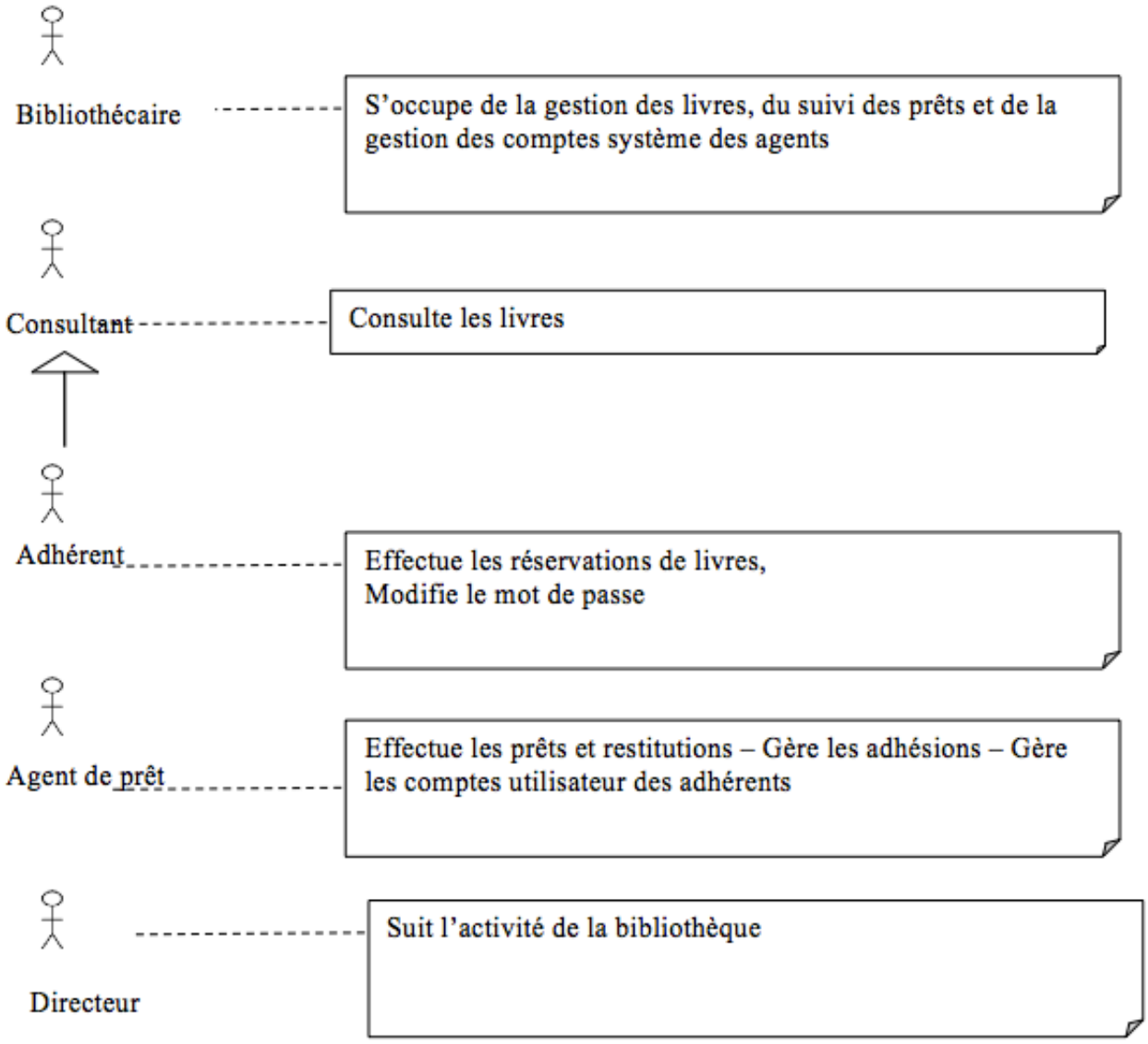
Identification des besoins fonctionnels et des besoins opérationnels

- Besoins fonctionnels
  - Gestion des adhérents (ajouter les adhérents, renouveler les adhésions, …)
  - Gestion des prêts (ajouter les prêts, prolonger les prêts, restituer les livres,

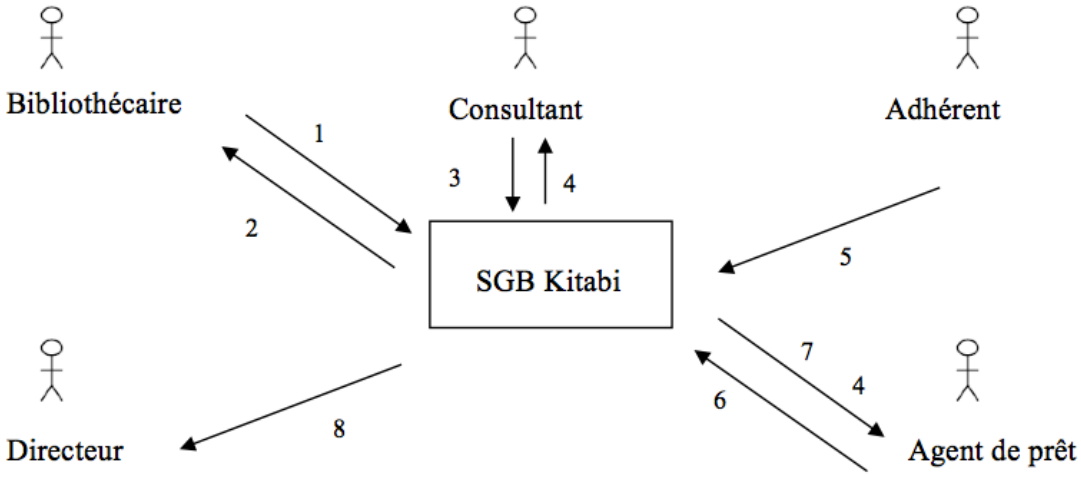
gérer les sanctions...)
- Gestion des livres
- Suivi des prêts
- Suivi de l'activité de la bibliothèque
- Consultation de livres

- Besoins opérationnels
  - Gérer les mots de passe (créer les utilisateurs techniques / mots de passe, réinitialiser les mots de passe, modifier les mots de passe)
  - Gérer l'historique des recherches
  - Accéder au système en interne (hall de la bibliothèque) et de l'extérieur (Internet).

Identification des acteurs et affectation des rôles



Modélisation du contexte



Numéro du message	Nature des messages
1	Information sur les livres Informations sur les agents de prêt
2	Livres trop demandés Livres aux exemplaires insuffisants
3	Livres recherchés
4	Informations sur les livres et leurs disponibilités
5	Informations de réservation (livre, date de réservation) – annulation de réservation
6	Informations sur les adhérents – informations sur les prêts et restitutions – informations sur les comptes d'utilisateurs
7	Adhérents sanctionnés, informations détaillées sur les prêts
8	Taux de renouvellement, taux de prêt de livres

1.3. Série n° 3

Reprendre le cas « Kitabi » et faire le travail suivant :

1. Recenser les cas d'utilisation fonctionnels ;
2. Classer les cas d'utilisation par acteur et mentionner les envois et réception d'informations ;
3. Elaborer le diagramme de cas d'utilisation ;
4. Pour chaque cas d'utilisation recensé en 3

   - Décrire le cas textuellement ;
   - Elaborer une fiche structurant le cas, comme vu dans le cours ;

5. Illustrer l'activité de prêt d'un livre par un diagramme d'activités, ensuite par un

diagramme de séquences ;
6. Illustrer la restitution d'un livre par un digramme de séquences ;
7. Illustrer les changements d'état d'un livre et ceux d'un adhérent chacun par un diagramme d'état-transition ;
8. Pour chaque cas d'utilisation recensé en 3

- Elaborer un diagramme de classes participantes ;
- Décrire chaque classe par sa responsabilité.

Corrigé exemplaire partiel de la série 3

1. *Recensement des cas d'utilisation :*

- Gestion des adhérents
- Gestion des adhésions
- Gestion des prêts
- Gestion des livres
- Elaboration de statistiques de suivi des prêts
- Elaboration des statistiques de suivi de la bibliothèque
- Gestion de consultations internes des livres

2. *Classement des cas d'utilisation*

Cas d'utilisation	**Acteurs**	**Envoie / réception d'information**
Gestion des adhérents	Agent de prêts	Envoie : informations détaillées sur les adhérents Reçoit : Liste des adhérents
Gestion des adhésions	Agent de prêts	Envoie : Les informations d'adhésion (ex. date) et sur les renouvellements Reçoit : la liste des adhésions (surtout pour renouvellement)
Gestion des prêts	Agent de prêts	Envoie : informations sur les prêts et restitutions Reçoit : disponibilité des livres, adhérents sanctionnés
	Adhérent	Envoie : réservation de livres Reçoit : disponibilité des livres, livres réservés
	Consultant	Reçoit : disponibilité des livres
Gestion des livres	Bibliothécaire	Envoie : information détaillée sur les livres Reçoit : liste des livres
Elaboration de statistiques des prêts	Bibliothécaire	Reçoit : statistiques de suivi des prêts
Elaboration des statistiques de la bibliothèque	Directeur	Reçoit : statistiques de suivi de la bibliothèque
Gestion de consultations des livres	Agent de prêt	Envoie : informations sur la consultation (livre, consultant) Reçoit : livres en cours de consultations (en particulier les coordonnées du consultant)

1.4. Série n° 4

Exercice 1 : *Capture des besoins techniques*

Reprendre la description du cas de la bibliothèque *kitabi* (série n°2) et faire le travail suivant

1. Spécifier l'architecture du système du point de vue matériel ;
2. Spécifier l'architecture du système du point de vue logiciel (composants) ;
3. A partir de l'expression des besoins techniques, élaborer le diagramme de cas d'utilisation techniques.

Exercice 2 : *Analyse*

1. A partir des classes identifiées dans les diagrammes de classes participantes dans la série 3, procéder à un découpage en catégories et illustrer les dépendances entre catégories par un diagramme de packages ;
2. Pour le cas d'utilisation *gestion des prêts*

    a. identifier les différents scénarios du cas en montrant les scénarios nominaux et éventuellement les scénarios alternatifs, aux limites et d'erreur ;

    b. choisir un scénario et illustrer par un diagramme de séquences en ajoutant les objets.

2. Année universitaire 2012-2013

2.1. Série d'exercices n° 1

Exercice 1 : Le système à mettre en place concerne la réservation en ligne de tables dans un restaurant. Un client qui souhaite réserver une ou plusieurs tables doit créer un compte et se connecter. Il peut visualiser la liste des tables disponibles pour une date donnée. Il peut réserver une ou plusieurs tables, annuler une réservation, ou modifier le nombre de tables. Le client doit confirmer son arrivée au plus tard 3 heures avant le repas en ligne ou par téléphone. Le gérant peut consulter la liste des tables réservées par date. Il peut effectuer la réservation sur place pour des clients qui se présentent physiquement au restaurant. Lorsque le client se présente au repas, après son identification physique, le gérant met à jour le statut de la réservation comme «présence». Sinon, si le client ne se présente pas ou vient en nombre insuffisant, le système enregistre respectivement «absence» et «sous-exploitation». A la fin de chaque mois, le gérant élabore les statistiques sur les mauvais clients, les habitués, etc. Le système peut refuser la réservation à un mauvais client au delà d'un certain seuil de réservations absentes ou sous-exploitées.

Exercice 2 : Le système à mettre en place concerne une vidéothèque de prêt de DVD de films. La réservation se fait au sein de la vidéothèque. Toute personne désireuse de prendre un film doit d'abord consulter la liste des films sur des écrans placés à l'entrée de la vidéothèque. Le système affiche la liste des films et pour chacun, le nombre d'exemplaires disponibles. Une fois le film choisi, le gérant de la vidéothèque ou son assistant sélectionne le film, le numéro d'exemplaire, et les dates de prêt et de restitution avec le type de pièce d'identité remise par le client et son numéro. La restitution se fait par la recherche du film avec numéro d'exemplaire ou par la recherche du client par sa pièce d'identité. Le système doit permettre différents

affichage des films (par date de parution, par genre, par acteur, etc.) Enfin, le système doit effectuer un archivage automatique des films qui n'ont pas été demandés pendant deux années.

Exercice 3 : Le système à mettre en place concerne la gestion de la caisse d'un centre de formation. La caisse est tenue par le réceptionniste du centre ou par l'agent de permanence dans les week-end. Les payements des formations se font régulièrement par les stagiaires (chaque mois). A la connexion au système, le réceptionniste ou agent s'authentifie. A chaque payement, on sélectionne le stagiaire, la formation qu'il suit, et on ajoute le montant. Le système peut à la demande du stagiaire indiquer le montant qui lui reste à payer. Aussi, lors de l'inscription à une formation, le stagiaire effectue une avance de payement. Cette avance est remboursable si le stagiaire annule son inscription, ce qui se traduit par une sortie d'argent de la caisse. A la fin de chaque journée, l'agent consulte le solde de la caisse qui doit être égal au solde de la journée de travail précédente, plus le total des entrées d'argent moins le total des sorties d'argent. Enfin, le système doit effectuer régulièrement la journalisation des opérations d'entrée / sortie dans un fichier log (journal) caché et accessible seulement par le directeur du centre. Ce fichier permet de savoir qui a effectué n'importe quelle opération d'entrée ou de sortie d'argent de la caisse (réceptionniste ou agent).

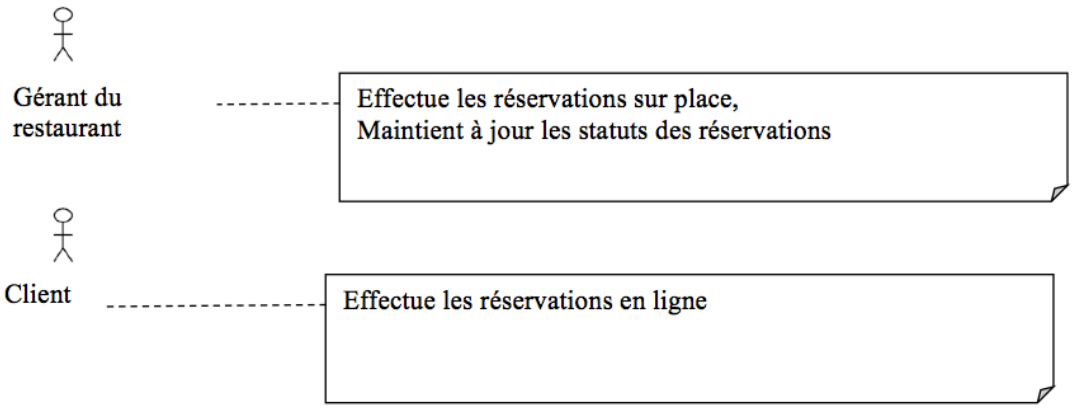
Travail demandé : il est demandé pour chacun des cas ci-haut, d'

1. Identifier les besoins fonctionnels et les besoins opérationnels ;
2. Identifier les acteurs et leurs responsabilités ;
3. Modéliser le diagramme de contexte dynamique.

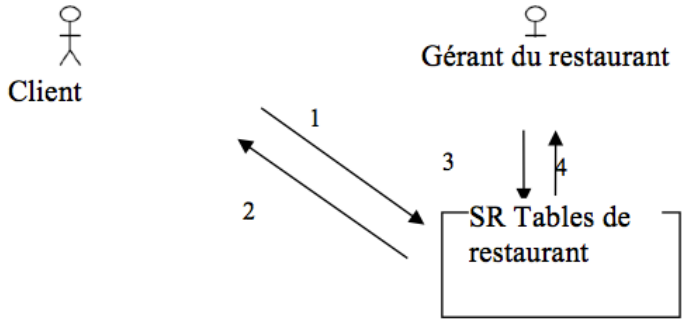
Corrigé type du TD 1

Exercice 1 :

- Identification des besoins
    - besoins fonctionnels
        - connaître le nombre de tables disponibles pour une date et plage horaire données,
        - réserver, modifier ou annuler une réservation,
        - connaître les mauvais clients et les habitués,
        - interdire la réservation aux mauvais clients.
    - besoins techniques
        - besoin de personnalisation des sessions à travers un compte pour accéder aux réservations,
        - accès en ligne (besoin de disponibilité du serveur Web).
- Identification des acteurs



- Diagramme de contexte dynamique



Numéro du message	Nature des messages
1	Informations de réservation (nombre de tables, date et horaire)
2	Liste des tables disponibles par date et par horaire Confirmation d'arrivée
3	Statut de réservation
4	Liste de tables réservées par horaire et date Statistiques de mauvais clients et habitués

Exercice 2 : Vidéothèque

–Identification des besoins

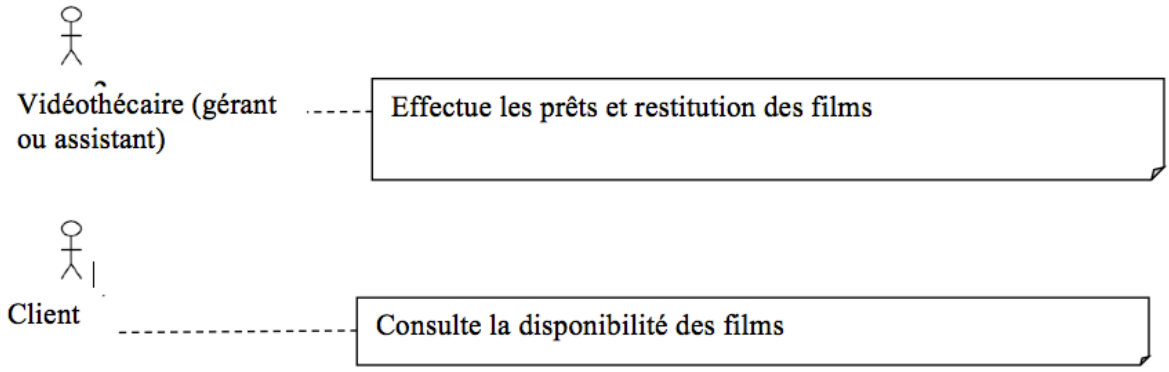
1.besoins fonctionnels :

1.connaître les films disponibles
2.prêter les films et les restituer

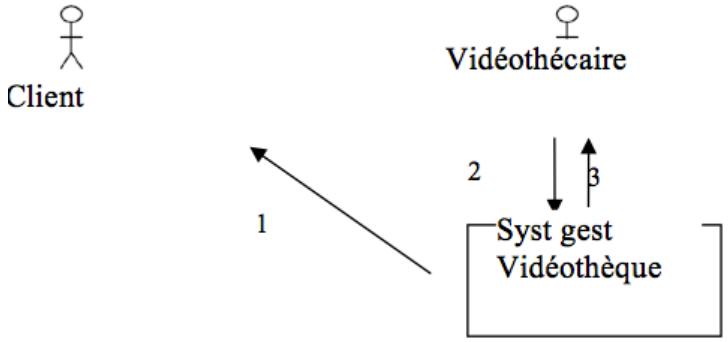
2.besoins techniques

1.archivage automatique des films
2.accès en consultation sur les écrans

–Identification des acteurs



–Diagramme de contexte dynamique



Numéro du message	Nature des messages
1	Liste des films disponibles
2	Informations sur les prêts (nom & prénom, infos sur le film), informations de restitutions (date)
3	Liste des films réservés, disponibles

Exercice 3 :

- Identification des besoins

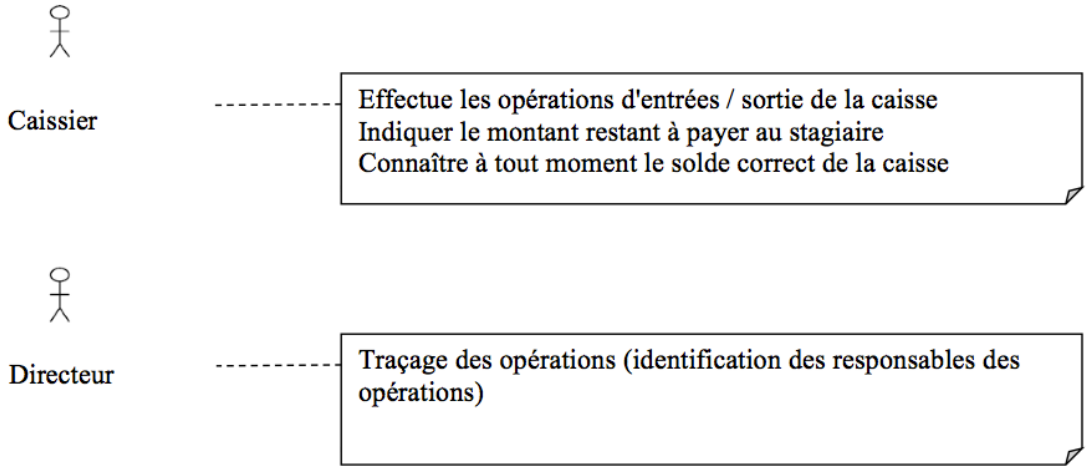
–besoins fonctionnels

–effectuer des opérations d'entrée / sortie
–indiquer le montant restant à payer par stagiaire

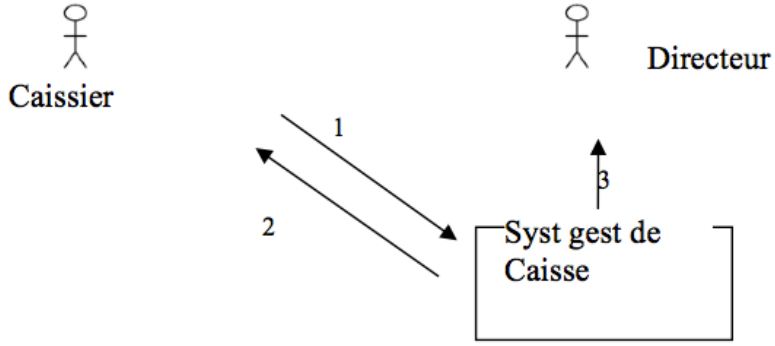
–besoins techniques

–sécurisation à travers la connexion
–traçage des opérations (fichier log)
–sécurisation de l'accès au fichier log

- Identification des acteurs



- Diagramme de contexte dynamique



Numéro du message	Nature des messages
1	Informations sur l'opération : nature, montant, stagiaire concerné
2	Solde Montant restant à payer
3	Traces des opérations Solde

2.2. Série n° 2

NB : Cette série d'exercices complète la série précédente.

Exercice 1 (cas réservation de tables de restaurant)

1. Identifier les cas d'utilisation en les associant aux acteurs respectifs
2. Parmi les cas identifiés précédemment,

   - Quel est le cas qui représente l'activité de base ?
   - Identifier les messages en entrée et en sortie de ce cas,
   - Elaborer une fiche descriptive de ce cas.

Exercice 2 (cas de la vidéothèque)

Nous avons identifié les cas d'utilisation fonctionnels suivants : (1) Gestion des prêts de films, (2) Consultation de la liste des films

1. Associer chacune des cas au(x) acteur(s) respectifs,
2. Elaborer le diagramme d'activité du cas « gestion des prêts »,
3. Elaborer le diagramme de séquences du cas précédent en incluant les différents enchainements possibles,
4. Elaborer le diagramme de classes candidates du cas « Consultation de la liste des films »,
5. Identifier les classes sujettes à des changements d'état et élaborer pour chacune un diagramme d'état/transition.

Exercice 3 (cas de la gestion de la caisse du centre de formation)

Nous avons identifié les cas d'utilisation fonctionnels suivants : (1) Ajout d'opération, (2) Ajout d'entrée en caisse, (3) Ajout de sortie de caisse, (4) Consultation du solde de la caisse, (5) Consultation du montant restant à payer.

1. Associer chacune des cas au(x) acteur(s) respectifs,
2. Elaborer le diagramme de cas d'utilisation préliminaire,
3. Restructurer les cas précédent en identifiant les éventuelles relations d'inclusion, d'extension et de généralisation / spécialisation.

2.3. Série n° 3

Exercice 1 : *pour les trois exercices de la série 1, faire le travail suivant*

1. Elaborer le modèle de spécification du point de vue matériel,
2. Elaborer le modèle d'architecture en identifiant les composants,
3. Identifier les acteurs et cas d'utilisation techniques et dessiner le diagramme des cas d'utilisation techniques.

Exercice 2 : *Gestion de courrier d'une entreprise*

On veut développer un système de gestion de courrier pour le directeur d'une entreprise. Le directeur est aidé par trois secrétaires qui se chargent entre autres de gérer le courrier. On veut gérer le courrier à travers une application déployée au niveau du secrétariat. Le système doit permettre de stocker le courrier entrant et sortant, de trier le courrier par date, par expéditeur/destinataire, par degré d'urgence et de rechercher les courriers par mot-clé. D'autre part, chaque secrétaire est responsable, à travers le système, du courrier qu'il envoie ou reçoit. Aussi, le système doit permettre au directeur, à travers un accès sécurisé, de consulter le courrier et d'effectuer des analyses du courrier (fréquence d'envoie, fréquence de réception, etc). Le système dispose d'une aide sur toutes les fonctionnalités du système.

Travail demandé

1. Identifier les acteurs et les cas d'utilisation techniques et dessiner le diagramme de cas d'utilisations techniques;
2. Elaborer le modèle de spécification du point de vue matériel,

3. Elaborer le modèle d'architecture,
4. A cause de ses déplacements, le directeur souhaite être notifié par email du courrier urgent et souhaite pouvoir consulter le détail du courrier sur le navigateur de appareil portable (téléphone ou ordinateur). Modifier le modèle de spécification matérielle et le modèle d'architecture pour tenir compte de ces nouveaux besoins.

2.4. Série n° 4

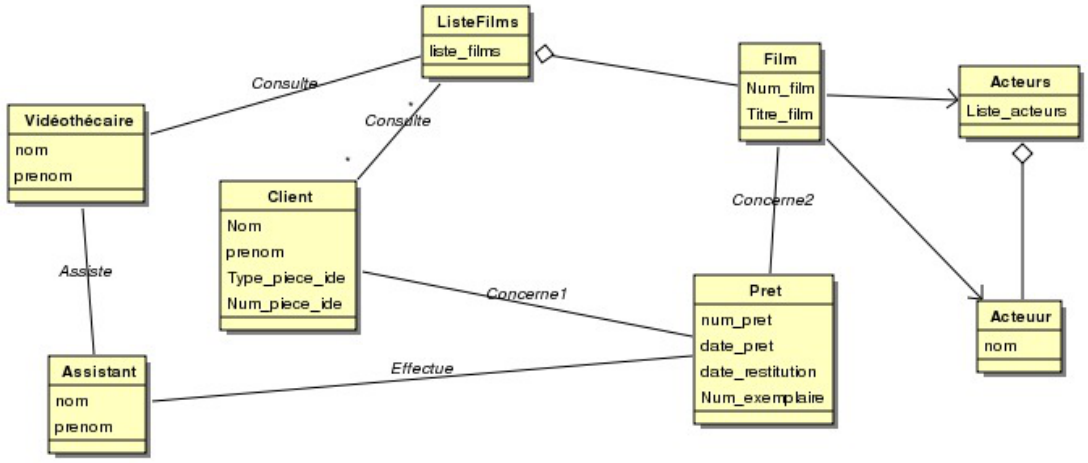
Exercice 1 : *Découpage en catégories*

1. Elaborer le diagramme de classes global de la réservation des tables de restaurant
2. Regrouper les classes en catégories en spécifiant

- Le critère de découpage
- Les dépendances entre catégories

Exercice 2 : *Développement du modèle statique*

Soit le diagramme de classes suivants représentant les prêts dans la vidéothèque et issu du regroupement des classes candidates :



- Détecter et corriger les anomalies et compléter les multiplicités du diagramme ci-dessus,
- Ajouter les contraintes nécessaires à la bonne gestion des prêts,
- Compléter le diagramme pour permettre de connaître le nombre d'exemplaires disponibles d'un film et ne pas les calculer à chaque fois.

Exercice 3 : *Développement du modèle dynamique*

Reprendre le diagramme de cas d'utilisation structuré de la gestion de caisse. Pour chaque cas

1. Identifier et classer les différents scénarios,
2. Pour chaque scénario identifié en 1, développer le modèle dynamique à travers un diagramme de séquences (acteur, objets).

2.5. Série n° 5

Exercice 1

1. Reprendre le modèle à 5 couches logicielles vu dans le cours,
2. A partir du diagramme de classes global de la réservation des tables de restaurant, proposer un dictionnaire des termes techniques associés à chaque couche logicielle et donner des exemples pour chaque couche à partir du système de réservation des tables.

Exercice 2

1. Reprendre le diagramme de cas d'utilisation techniques de la vidéothèque,
2. Elaborer le(s) framework(s) technique(s) permettant de répondre aux besoins techniques en donnant un exemple des instances des classes techniques identifiées
3. Elaborer le modèle logique de conception technique en associant le(s) nouveau(x) framework(s) technique(s) à ceux du modèle à 5 couches logicielles.

Exercice 3

1. Reprendre le diagramme de cas d'utilisation techniques de la caisse,
2. Elaborer le modèle d'exploitation de la conception technique en montrant le(s) nouveaux composants d'exploitation qui répondent aux besoins techniques.

E**xercice 4**

1. Rependre le diagramme de cas d'utilisation techniques du suivi du courrier,
2. Elaborer le(s) framework(s) technique(s) permettant de répondre aux besoins techniques en donnant un exemple des instances des classes techniques identifiées,
3. Elaborer le modèle logique de conception technique en associant les nouveaux framework techniques à ceux du modèle à 5 couches logicielles,
4. Elaborer le modèle d'exploitation de la conception technique en montrant le(s) nouveaux composants d'exploitation qui répondent aux besoins techniques.

3. Année universitaire 2013-2014

3.1. Série n° 1

Exercice 1 : On veut mettre en place un système d'informations pour gérer les réservations en ligne de tables dans un restaurant. Un client qui souhaite réserver une ou plusieurs tables doit créer un compte et se connecter. Il peut visualiser la liste des tables disponibles pour une date donnée. Il peut réserver une ou plusieurs tables, annuler une réservation, ou modifier le nombre de tables. Le client doit confirmer son arrivée au plus tard 3 heures avant le repas en ligne ou par téléphone. Le gérant peut consulter la liste des tables réservées par date. Il peut effectuer la réservation sur place pour des clients qui se présentent physiquement au restaurant. Lorsque le client se présente au repas, après son identification physique, le gérant met à jour le statut de la réservation comme «présence». Sinon, si le client ne se présente pas ou vient en nombre insuffisant, le système enregistre respectivement «absence» et «sous-exploitation». A la fin de chaque mois, le gérant élabore les statistiques sur les mauvais clients, les habitués, etc. Le système peut refuser la réservation à un mauvais client au delà d'un certain seuil de réservations absentes ou sous-exploitées.

1. Donner le diagramme de classes de ce cas en renseignant le minimum d'attributs et de méthodes des classes.
2. Valider le digramme de classes par un diagramme d'objets.

Exercice 2 : Le système d'information à mettre en place concerne une vidéothèque de prêt de DVD de films. La réservation se fait au sein de la vidéothèque. Toute personne désireuse de prendre un film doit d'abord consulter la liste des films sur des écrans placés à l'entrée de la vidéothèque. Le système affiche la liste des films et pour chacun, le nombre d'exemplaires disponibles. Une fois le film choisi, le gérant de la vidéothèque ou son assistant sélectionne le film, le numéro d'exemplaire, et les dates de prêt et de restitution avec le type de pièce d'identité remise par le client et son numéro. La restitution se fait par la recherche du film avec numéro d'exemplaire ou par la recherche du client par sa pièce d'identité. Le système doit permettre différents affichage des films (par date de parution, par genre, par acteur, etc.) Enfin, le système doit effectuer un archivage automatique des films qui n'ont pas été demandés pendant deux années.

1. Représenter le fonctionnement du système par un ou plusieurs diagrammes d'activités
2. Identifier la ou les classes sujettes à des changements d'états. Représenter ces changements par des diagrammes d'états/transitions.

Exercice 3 : Le système à mettre en place concerne la gestion de la caisse d'un centre de formation. La caisse est tenue par le réceptionniste du centre ou par l'agent de permanence dans les week-ends. Les payements des formations se font régulièrement par les stagiaires (chaque mois). A la connexion au système, le réceptionniste ou agent s'authentifie. A chaque payement, on sélectionne le stagiaire, la formation qu'il suit, et on ajoute le montant. Le système peut à la demande du stagiaire indiquer le montant qui lui reste à payer. Aussi, lors de l'inscription à une formation, le stagiaire effectue une avance de payement. Cette avance est remboursable si le stagiaire annule son inscription, ce qui se traduit par une sortie d'argent de la caisse. A la fin de chaque journée, l'agent consulte le solde de la caisse qui doit être égal au solde de la journée de travail précédente, plus le total des entrées d'argent moins le total des sorties d'argent. Enfin, le système doit effectuer régulièrement la journalisation des opérations d'entrée / sortie dans un fichier log (journal) caché et accessible seulement par le directeur du centre. Ce fichier permet de savoir qui a effectué n'importe quelle opération d'entrée ou de sortie d'argent de la caisse (réceptionniste ou agent).

Travail demandé : Refaire les questions des exercices précédents pour ce cas.

3.2. Série n° 2

QuickMail (courrier rapide) est une entreprise de délivrance rapide de courrier ; elle possède une branche dans chaque ville. Pour envoyer un courrier rapide, le client le dépose à QuickMail (QM). Le système à mettre en œuvre doit permettre à l'agent de QM d'enregistrer le nouveau courrier et de délivrer au client un bordereau contenant un numéro et un code barre. Le client doit pouvoir connaître le statut de son courrier à l'aide du numéro ou code barre. Le client peut également se renseigner auprès de l'agent par téléphone sur son courrier. Dans ce cas, l'agent recherche le courrier par son numéro ou par d'autres critères (nom et prénom, date de dépôt, etc). Dans tous les cas, le système doit sauvegarder ces critères et les valeurs de recherche pour permettre la réindexation manuelle de la base de données.

Chaque courrier déposé est programmé le lendemain matin. Le système affichera pour cela la liste des courriers à programmer. Le programmateur complète ou rectifie l'adresse d'envoi du courrier. Pour cela, il utilise un système externe pour la localisation géographique. Ce système permet de corriger une adresse approximative et de la positionner sur une carte géographique. Si une adresse est incorrecte, le courrier est retiré de la liste et reçoit le statut automatique de « en attente de correction d'adresse». Une fois les adresses rectifiées, le programmateur demande au système externe de les regrouper et de générer une liste de trajets (suite de plusieurs ville enchainées). Pour chaque trajet, le programmateur affecte un chauffeur et un véhicule disponibles à partir d'une liste de disponibilité. A la fin de la programmation, les courriers programmés auront le statut « programmé ».

Au départ des courriers, l'agent modifie leur statut à « en route » et celui des chauffeurs et véhicules affectés à « en mission ». Au retour des missions, l'agent libère le chauffeur et véhicule et modifie le statut du courrier soit à « délivré » ou « non délivré ».

Le statut d'un courrier doit être consultable à tout moment (24h/24, 7j/7). Aussi, pour éviter les problèmes techniques, un spécialiste bases de données est sollicité régulièrement pour sauvegarder la base de données et la récupérer en cas de perte.

Enfin, pour augmenter la performance de la recherche de courrier, le spécialiste en base de données visualise à chaque visite les critères de recherche les plus utilisés et effectue une réindexation des tables de la base de données. Il peut aussi utiliser les critères de recherche pour modifier et réorganiser le schéma de la base de données.

Travail demandé :

1. Identifier les besoins fonctionnels et les besoins opérationnels**.**

2. Identifier les acteurs du système.

3. Schématiser le diagramme de contexte dynamique.

3.3. Série n° 3

Exercice 1 : On veut concevoir le système d'information d'un cabinet médical. Le cabinet est composé du médecin généraliste, d'un infirmier assistant le médecin dans diverses tâches (mesure de tension, injections, …). Une réceptionniste accueille les patients, enregistre les rendez-vous, et s'occupe de la gestion de la caisse. Les patients se présentent au cabinet ou téléphonent pour la prise de rendez-vous. La réceptionniste note le nom, prénom du patient ainsi que la date et heure du rendez-vous. Le patient peut demander d'annuler ou de reporter le rendez-vous. Le médecin tient une fiche de chaque patient où il note les informations sur ce dernier et les visites effectuées. Lorsque le patient se présente pour la première fois, on lui créé une nouvelle fiche. Cette fiche permet le suivi de près de chaque patient et elle est constamment modifiée pour ce qui est des informations générales ou des informations de suivi médical. Pour chaque visite, le médecin note sur le registre des visites la date, le nom et prénom du patient, le type de la visite (consultation ou contrôle). Il note également les différentes mesures effectués (tension, poids,…). Une visite de type consultation est caractérisée par la constatation faite par le médecin sur l'état de santé du patient (diagnostic, symptômes,…). Pour une visite de type contrôle, on note l'évolution de l'état de santé du patient (état avant, état après). Chaque visite donne lieu à une éventuelle prescription médicale. En plus du nom, prénom et âge du patient, la prescription contient l'ensemble des

médicaments avec, pour chaque médicament, la quantité et le mode de prise (posologie, fréquence de prise, durée,…).

Travail demandé : Elaborer le diagramme de cas d'utilisation fonctionnels en illustrant les liens entre les cas.

Exercice 2 : Reprendre l'énoncé de l'exercice 2 de la série 1. Nous avons identifié les cas d'utilisation fonctionnels suivants : (1) Gestion des prêts de films, (2) Consultation de de la liste des films.

Travail demandé : Pour chacun des cas d'utilisation précédents

1. Identifier les différents scénarios d'exécution
2. Elaborer une fiche descriptive.

Exercice 3 : Reprendre l'exercice 3 de la série 1. Nous avons identifié les cas d'utilisation fonctionnels suivants : (1) Ajout d'opérations, (2) Ajout d'entrée en caisse, (3) Ajout de sortie de caisse, (4) Consultation du solde de la caisse, (5) Consultation du montant restant à payer, (6) consultation des informations sur les stagiaires. Les cas sont liés comme suit : (2) et (3) spécialisent (1), (3) inclut (4), (2) inclut (6) et (5) inclut (6).

Travail demandé : pour chacun des cas précédents

1. Elaborer le diagramme de classes candidates en mentionnant le rôle de chaque classe.
2. Quelle est l'influence des liens entre les cas d'utilisation sur les diagrammes de classes candidates ?

3.4. Série n° 4

Exercice 1 : On veut mettre en place un système informatisé pour la gestion d'une compétition annuelle nationale de natation à différents niveaux : wilaya, région (centre, est, ouest), central (Alger). Les compétions par wilaya s'effectuent en mois de Mars, celles des régions en mois de Mai, et la compétition finale a lieu en mois de Juillet. Chaque wilaya dispose d'une base de données locale pour gérer ses nageurs, laquelle est manipulée par une application distribuée commune et spécifique à chaque région. La liste des nageurs sélectionnés par wilaya est répliquée en début Avril dans les bases de données régionales. Ces dernières sont manipulées par une application centrale (Alger) distribuée. La liste des nageurs sélectionnés par région est répliquée en début Juin vers la base de données centrale. Cette dernière est manipulée par une application centrale unique.

Travail demandé :

1. *Elaborer le modèle de déploiement à l'étape de capture des besoins techniques.*
2. *Elaborer le modèle de composants à l'étape de capture des besoins techniques.*

Exercice 2 : Une banque met à disposition de ses clients des cartes magnétiques pour le retrait d'argent de ses distributeurs. On veut mettre en place un système pour la gestion de ces cartes. Chaque carte est identifée par un numéro unique. Lors de la première délivrance, le

client doit changer son code d'accès. Pour retirer l'argent, le client doit s'authentifier par son code secret. Si le code secret est renseigné correctement, le système délivre l'argent que le client retirera. Sinon, au bout de trois mauvaises introductions du code secret, le système bloque la carte qui devient inutilisable temporairement. Le client doit se faire débloquer la carte par l'agent correspondant à la banque. Aussi, par mesure de sécurité, si le client ne retire pas sa carte au bout de 15 secondes ou l'oublie, le lecteur avale cette dernière. Le client devrait demander sa carte auprès du maintenancier du distributeur. Ce dernier s'occupe d'alimenter le distributeur par l'argent, de le vider par mesure de sécurité selon les circonstances, de retirer les cartes avalées et de mettre à niveau le système logiciel du distributeur.

Travail demandé : *Identifier les cas d'utilisation techniques et élaborer le diagramme de cas d'utilisation techniques.*

Exercice 3 : Reprendre l'énoncé de QuickMail (série 2).

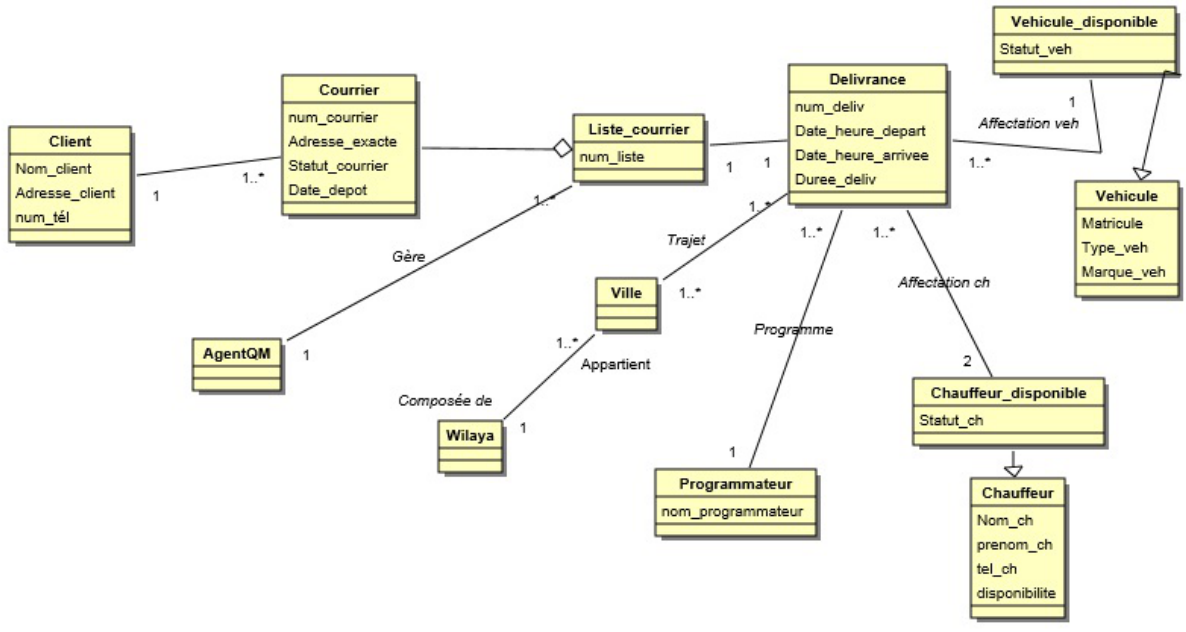
Travail demandé:

1. *Elaborer le diagramme de cas d'utilisation techniques*
2. *A quel type d'architecture appartient QuickMail. Elaborer le modèle de déploiement corresondant.*
3. *Elaborer le diagramme de composants à l'étape de capture des besoins techniques.*

3.5. Série n° 5

Exercice 1 : Soit le diagramme de classes global de QuickMail (série 2) issu de l'intégration des diagrammes de classes candidates.

Travail demandé : Détecter et corriger les anomalies contenues dans ce diagramme de classes.



Exercice 2 : Un responsable pédagogique veut élaborer le programme d'un emploi du temps pour les trois promotions d'étudiants en licence. Il dispose d'une liste de modules à

programmer, d'un ensemble de créneaux horaires et d'un ensemble de salles disponibles, la liste d'étudiants créditaires par module ainsi que de la liste d'enseignants de sureveillance. On commence par programmer les examens de la classe de 1ère année, puis celle de 2ème année puis celle de 3ème année. Le responsable commence par distribuer les modules sur les salles, puis par affecter les enseignants de surveillance. L'emploi du temps est d'abord consulté par les enseignants. Chaque enseignant a le droit de demander à changer une séance de surveillance au plus, en justifiant sa demande. Aussi, le système doit permettre de gérer les conflits de salles, celui des surveillances et aussi le conflit entre les modules (pour gérer les cas des étudiants créditaires) et de respecter le nombre limite de surveillances par enseignant.

Travail demandé:

1. Identifier les différents cas d'utilisation fonctionnels.
2. Pour chaque cas d'utilisation, identifier les différents scénarios (nominal, alternatif, terminal, d'erreur) en mentionnant les liens entre les scénarios.

Exercice 3 : Reprendre le cas de la gestion de la caisse du centre de formation.

Travail demandé : Développer le modèle dynamique du système de gestion de la caisse en élaborant un diagramme de séquences pour chaque cas d'utilisation fonctionnel.

4. Année universitaire 2014- 2015

4.1. Série 1

Exercice 1 : On veut développer une application de lecture de livres électroniques (ebook reader). L'application doit fonctionner comme suit : Lors de la première utilisation, la bibliothèque des livres est vide. Le lecteur peut enrichir la bibliothèque par des livres depuis sont support de stockage vers la bibliothèque. Il peut également chercher des livres sur Internet à partir de l'application et les télécharger. Les livres téléchargés sont automatiquement ajoutés à la bibliothèque. Pour lire un livre, le lecteur peut soit le chercher dans la bibliothèque, soit y accéder directement à partir des bookmarks (point de repère). Lors de la lecture, le lecteur peut ajouter des bookmarks, ou peut extraire des phrases et les sauvegarder sous forme de *passages favoris* en format texte. Le lecteur pourra consulter la liste des passages favoris pour les lire. Le ebook reader doit offrir des fonctionnalités d'accessibilité en tenant compte de la vision des personnes en offrant différentes tailles de la police de caractère (minime, petite, moyenne, grande, très grande). Pour rendre la lecture agréable plus conviviale, le lecteur doit pouvoir personnaliser la couleur de fond des pages.
Le lecteur doit pouvoir organiser sa bibliothèque en supprimant les livres déjà lus ou inintéressants. Pour faciliter cette suppression, le système doit afficher, en dessous du titre de chaque livre, le pourcentage de la partie lue. Pour les grands livres qui ne tiennent pas en mémoire vive du lecteur, permettre de préparer en mémoire les futures pages et garder les plus récentes en cas de retour vers elles.

Travail demandé : Identifier les besoins métier et les besoins opérationnels.

Exercice 2 : On veut développer une application Web qui permet à une équipe d'employés d'organiser des promenades en week-end. On désigne parmi les participants un organisateur et un cuisinier. L'organisateur se charge d'ajouter l'ensemble des dates de week-ends avec

une case à cocher sous chaque date. Le cuisinier se charge d'ajouter l'ensemble des plats qu'il peut cuisiner avec une case à cocher sous chaque plat. Chaque participant peut choisir une ou plusieurs dates qui le conviennent selon sa disponibilité. Aussi, il peut choisir les plats qu'il a envie de manger. On fixe une date pour remplir les choix de dates et de plats. Au terme de cette date, l'organisateur et le cuisinier consultent les choix faits par les participants. Pour les dates, l'organisateur valide les dates qui ont été choisies par au moins 2/3 des participants et envoie un message de confirmation aux participants de chaque date. Pour les plats, le cuisinier classe les plats selon le nombre de choix et sélectionne (valide) les 4 premiers. Il envoie également un message de confirmation aux participants avec la somme à payer pour chaque promenade. Dans le cas où un empêchement majeur se produit à un participant, ce dernier peut informer les autres participants via le système en mentionnant la cause de l'absence. Après chaque promenade effectuée, chaque participant peut noter son déroulement avec une note entre 0 et 5. Ces notes permettent d'évaluer la promenade en calculant la note moyenne.

Travail demandé :

1. Identifier les acteurs du système avec leurs responsabilités.
2. Modéliser le contexte de ce système par un diagramme de collaboration en montrant les messages envoyés et reçus.

4.2. Série 2

Exercice 1 : reprendre l'énoncé de l'exercice 1 de la série n°1.

Travail demandé

1. Elaborer le diagramme de cas d'utilisation fonctionnels en montrant les acteurs et leurs liens avec les cas.

2. Pour chaque cas, élaborer une fiche descriptive en mettant l'accent sur les enchainements.

3. Affiner le diagramme précédent en montrant les éventuels liens entre les cas.

Exercice 2 : *Gestion d'un agenda électronique*

On veut développer une application informatique pour la gestion d'un agenda électronique d'un responsable. L'agenda doit permettre à ce dernier d'organiser ses rendez-vous et activités qu'on appelèra *événements*. L'agenda est tenu par le responsable ou pas sa secrétaire. Les événements sont affichés sur un calendrier croisé : en colonnes les jours et en lignes les heures de 00h00 à 23h. A la demande du responsable, la secrétaire ajoute des événements en choisissant la date, l'heure et la durée. Pour cela, elle consulte d'abord le calendrier pour choisir un crénaux convenable mais elle doit avoir la confirmation du responsable avant de valider l'ajout. Si le responsable est disponible, la secrétaire demande la confirmation verbalement. Sinon, elle met en attente le choix et envoie un mail au responsable. A la lecture du mail, le responsable consulte l'agenda et confirme les choix convenables mis en attente de la secrétaire. Pour les choix non convenables, il peut demander à la secrétaire de les modifier ou il peut lui-même les modifier et les confirmer directement.

Dans plusieurs cas, lorsque les événements sont urgents ou ont des dates et heures fixées, la secrétaire peut modifier des crénaux existants par d'autres plus urgents ou fixés. Dans ce cas, l'événement urgent ou fixé remplace l'événement existant et ce dernier sera déplacé à un autre

crénau. La programmation d'un événement fixé n'attend pas la confirmation du responsable. Par contre, celle des événements urgents et les déplacements d'événements existants doivent avoir la confirmation du responsable.

Dans certains cas, le responsable peut décider d'annuler des évènements et confie à sa secrétaire la tâche d'apporter ces annulations à l'agenda. Enfin, au début de chaque journée, la secrétaire met à jour le statut des événements de la veille en mentionnant sa réalisation ou pas.

Travail demandé

1. Elaborer le diagramme de cas d'utilisation fonctionnels en montrant les liens entre les cas
2. Pour chaque cas, élaborer le diagramme de classes participantes en indiquant la responsabilité de chaque classe.

4.3. Série 3

Exercice 1 : On veut développer un système pour gérer les données des élections législatives. Les élections sont gérées au niveau du ministère de l'Intérieur, des Wilayas et des bureaux de vote. Chaque wilaya dispose d'un ensemble de fiches de candidats en format word. Ces fiches sont accessibles via un serveur de fichier à partir de terminaux mis à la disposition des électeurs dans chaque bureau de vote. Après fermeture des bureaux de vote à 20h00, un agent de bureau saisit les données de vote sur un terminal à travers une application de saisie centralisée par wilaya. Les données saisies sont stockées directement dans la base de données de la wilaya. À 22h00, les données de chaque wilaya sont répliquées dans la base de données du ministère. Pour des mesures de sécurité, une copie de ces données est stockée dans une base de données clone, hébergée sur une machine séparée. Enfin, les résultats des élections peuvent être analysées soit au niveau de chaque wilaya ou soit niveau du ministère (résultat global) à travers des applications spécifiques.

Travail demandé :

1. Elaborer le modèle de configuration matérielle à l'aide d'un diagramme de déploiement UML.
2. Elaborer le modèle de configuration logicielle à l'aide d'un diagramme de composants UML.

Exercice 2 : On veut développer une application pour gérer une superette. Les achats sont saisis par plusieurs caissiers avec un système de rotation par caisse. Avant de commencer le travail, chaque caissier doit d'abord s'identifier. L'attribution et la gestion des identifiants et des codes d'accès se fait par le gérant de la superette à travers l'espace confidentiel de ce dernier. Avant d'ouvrir la caisse, le caissier doit s'assurer que l'argent liquide disponible correspond au solde affiché.
A la présentation de l'acheteur devant la caisse, le caissier enregistre chaque article acheté par le biais du lecteur de code barre. Dans certains cas, lorsque le code barre n'est pas fonctionnel ou inexistant, le caissier bascule vers la saisie manuelle du produit. La lecture par code barre est le mode par défaut. Le caissier peut à tout moment mettre en pause l'enregistrement des articles, par exemple pour renseigner un autre acheteur sur le pris d'un produit, puis il reprend l'enregistrement.
Après la saisie de tous les produits d'un acheteur, ce dernier procède au payement. Le mode

de paiement par défaut est le payement cash (argent liquide). L'acheteur peut également payer en utilisant une carte de crédit. Dans ce cas, le caissier bascule vers le payement par carte. L'acheteur introduit sa carte dans un lecteur de carte lié au réseau de banques. L'acheteur saisit ensuite le code secret et valide. Le numéro de carte et le code introduit sont lus par le lecteur : si le code est correct, le caissier valide le payement. Sinon, après trois tentatives échouées du code de la carte bancaire, et si l'acheteur ne dispose pas d'argent, le montant des achats est sauvegardé comme « dette ». Par mesure de garantie (en cas d'acheteur inconnu), le numéro de la carte bancaire du client est à nouveau lu pour être associé à la dette et le caissier fixe un délai de payement. Une fois le payement effectué, la dette est effacée.

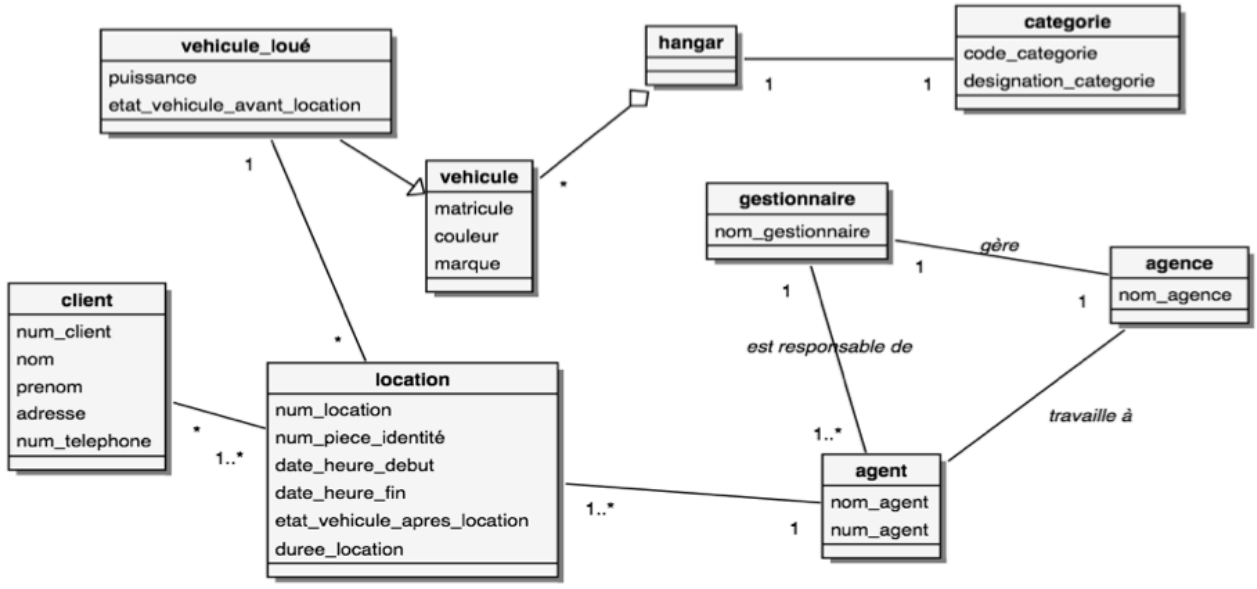
Travail demandé : élaborer le diagramme de cas d'utilisations techniques.

Exercice 3 : reprendre l'énoncé de l'exercice 1 de la série 1.

Travail demandé : élaborer le diagramme de cas d'utilisations techniques.

4.4. Série 4

Exercice 1 : ***Développement du modèle statique*** - Soit le diagramme de classes global suivant à l'étape d'analyse permettant de représenter la location des voitures d'une agence dirigée par un gestionnaire. L'agence possède plusieurs véhicules appartenant à différentes catégories (utilitaire, tourisme, ...). Les véhicules de chaque catégorie sont garées dans un hangar. Chaque location se fait à un seul client par un agent de l'agence. Ces agents sont payés selon le nombre de véhicules qu'ils ont loués. Ces agents sont sous la responsabilité directe du gestionnaire.



Travail demandé :

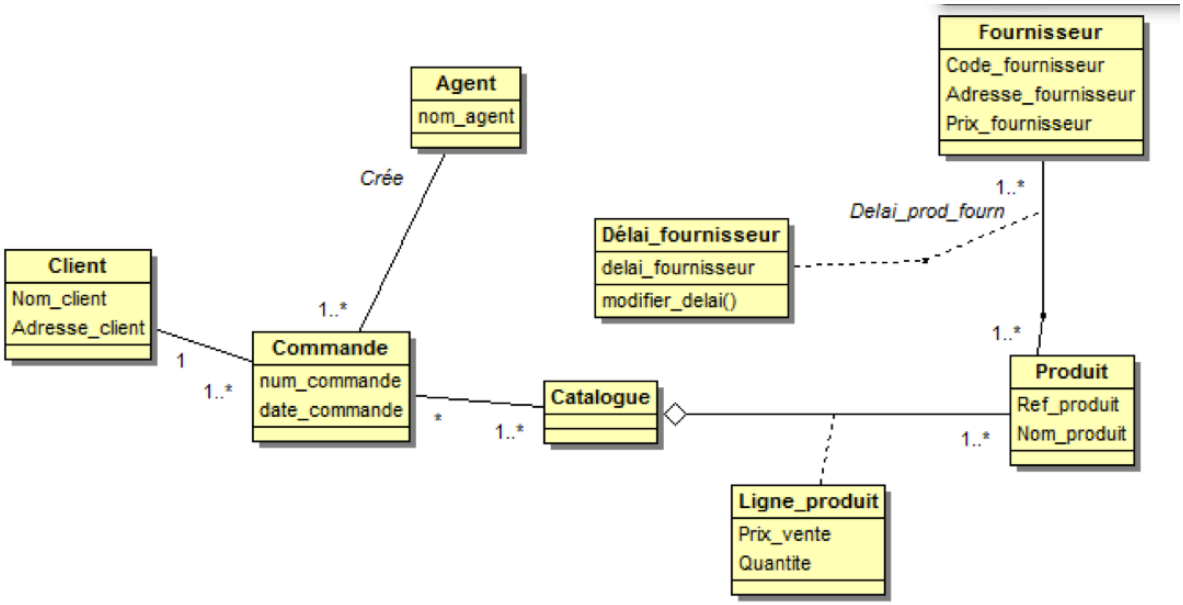
1. Identifier les anomalies contenues dans ce diagramme et proposer une correction à chacune.
2. Corriger les anomalies et compléter le diagramme par les opérations.

Exercice 2 : *Développement du modèle dynamique / Scénarios* - Soit l'activité de gestion de stock de produits dans une entreprise de vente. On s'intéresse au cas d'utilisation d'ajout d'une opération (sortie ou entrée) de produits du ou vers le stock. A chaque nouvelle opération, le gestionnaire de stock saisit les informations sur l'opération (date, quantité, type d'opération) et délivre les produits au demandeur (en cas de sortie) ou stocke les produits en cas d'entrée. Dans les deux cas, le gestionnaire note le justificatif de l'opération (numéro de bon d'achat en cas d'entrée et numéro de commande client en cas de sortie). Dans certains cas, le gestionnaire ajoute des opérations de sortie particulières à la suite de détérioration des produits. Dans ce cas, l'opération n'est associée à aucun numéro mais le gestionnaire la justifie par la cause de « détérioration ».

Travail demandé : identifier les différents scénarios du cas d'ajout d'une opération de stock (remarque : vu la simplicité de traitement, on considère un seul cas d'utilisation).

Exercice 3 : *Développement du modèle dynamique / diagrammes de séquences*

Soit le diagramme de classes suivant pour le traitement des commandes de clients par un agent commercial d'une entreprise.



On s'intéresse au cas d'utilisation d'ajout d'une commande décrit comme suit : l'agent commercial
renseigne les informations de la commande (numéro, date, etc.), ensuite il associe l'ensemble des produits commandés à partir d'un catalogue de produits. Pour chaque produit, l'agent choisit un des fournisseurs qui fournissent le produit. Ce choix est basé sur deux critères qui sont mis à jour automatiquement: le prix du produits proposé et le délai de livraison. Une fois le fournisseur de chaque produit choisi, l'agent renseigne la quantité commandée et le prix de la vente (après application d'un pourcentage de bénéfice). Enfin, l'agent établit le lien entre la commande et le client qui est choisi à partir d'une liste.

1. Détecter et corriger les anomalies contenues dans le diagramme ci-dessus.
2. Développer le diagramme de séquences du cas d'utilisation *ajout de commande* sur la base du diagramme de classes corrigé.

Annexe 2 : Interrogations 2011- 2014

1. Interrogation 2011-2012

Une petite entreprise veut mettre en place un système informatisé pour gérer sa messagerie interne (comme la messagerie de gmail, yahoo!, et hotmail). On affecte à chaque employé de l'entreprise une adresse email unique et un code secret pour se connecter. L'affectation est faite par l'informaticien de l'entreprise. Les employés ne peuvent pas modifier leurs codes d'accès mais lorsqu'un employé perd son code, l'informaticien lui en crée un nouveau.

Une fois connecté, l'employé peut écrire un nouveau message, il saisit l'objet du message, le texte et sélectionne un ou plusieurs destinataires. Il peut aussi joindre un ou plusieurs fichiers selon la capacité du système. Lorsque le message est prêt à être envoyé, l'employé a le choix soit de le sauvegarder comme brouillon, soit de l'annuler, soit de l'envoyer. Le message brouillon peut être ouvert plus tard et devient un message en cours de rédaction. L'employé peut donc le supprimer, le modifier puis l'envoyer ou le sauvegarder une autre fois comme brouillon.

Pour faciliter la sélection des destinataires des messages, le système les sauvegarde dans une liste. L'employé peut ajouter de nouveaux contacts avec pour chaque contact, un nom, un prénom et un alias en plus de l'adresse email existe déjà. Le système doit inviter l'employé à ajouter les destinataires de messages comme contacts après l'envoi d'un message si le destinataire n'existe pas déjà dans la liste des contacts. Sinon, l'ajout se fait de manière indépendante. Aussi, l'employé peut changer les informations d'un contact ou supprimer certains contacts. Un contact supprimé ne sort pas automatiquement dans la liste des destinataires lorsqu'on écrit un message.

Enfin, pour mieux gérer l'espace de stockage, le système doit permettre de nettoyer les boites de réception et d'envoi lorsqu'elles commencent à se remplir. L'employé peut afficher le contenu de chaque boite, classer les messages par date ou par envoyeur pour la boite de réception et par destinataire pour la boite d'envoi. Les messages que l'employé juge inintéressants peuvent être sélectionnés puis supprimés. Noter que la suppression est définitive.

Travail demandé

1. Identifier les acteurs fonctionnels et les cas d'utilisation fonctionnels et les représenter par un diagramme de cas d'utilisation en montrant les liens entre cas s'ils existent ;
2. Sélectionner un cas d'utilisation **fonctionnel** parmi ceux identifiés en 1et identifier les messages en entrée et en sortie.

2. Interrogation 2012-2013

Reprendre l'énoncé du cas QuickMail

Travail demandé

1. Identifier les besoins fonctionnels et les besoins opérationnels **(4 pts = 2+2)**
2. Elaborer le diagramme de cas d'utilisation fonctionnels en mentionnant les liens entre cas **(6pts)**

3. Choisir un cas d'utilisation fonctionnel et identifier les messages en entrées et en sortie **(2pt)**
4. Elaborer le diagramme de cas d'utilisation techniques **(3pts)**

3. Interrogation 2013-2014

Reprendre le cas du Ebook reader

Travail à faire:

1. Elaborer le diagramme de contexte dynamique du système (6 pts).
2. Elaborer le diagramme des cas d'utilisation fonctionnels en indiquant éventuellement les liens entre les cas (8 pts).

4. Interrogation 2014-2015

Cas RestoRapido

Le responsable d'un restaurant est soucieux de la qualité des repas qu'il offre à ses clients mais aussi de la rapidité de service. Il sollicite une boite informatique pour lui développer un système d'information qui lui permet de bien gérer son restaurant. Après l'étude préliminaire, on lui proposa le fonctionnement suivant :

A son entrée au restaurant, chaque client doit d'abord choisir une table disponible. Ensuite, il se dirige vers la caisse pour payer à l'avance son repas. Le caissier crée une nouvelle commande avec comme identifiant la date et heure d'entrée et le numéro de table. Il saisit le contenu de la commande et affiche au client le montant à payer. Après payement, la commande aura le statut « payée ». Au niveau de la cuisine, les cuisiniers disposent d'un écran géant tactile sur lequel est affichée la liste des commandes payées en attente de préparation. Dès qu'un cuisinier est disponible, il prend en charge la première commande selon l'ordre FIFO. Il met à jour le statut de la commande de « payée » à « en cours de préparation ». Une fois le repas préparé, le cuisinier met à jour le statut de la commande à « prête », la commande prête disparaît de l'écran. Le cuisinier imprime un ticket sur lequel est noté le numéro de la commande. Il dépose le plat sur une table avec son ticket.

La délivrance des commandes aux clients est prise en charge par les serveurs. Chaque serveur dispose d'une tablette connectée au système sur laquelle les plats prêts son affichés. Pour chaque plat qu'il récupère et sert, il met à jour son statut vers « servi ».

Dans certaines situations, des clients peuvent quitter le restaurant avant leur service (service lent, urgence, etc.). Ils se présentent au niveau de la caisse pour demander le remboursement. Le caissier cherche la commande et procède au remboursement. Si le plat est déjà prêt, il demande au client s'il veut le prendre (à emporter) ou renoncer à quitter et lui signale que le remboursement est impossible. Sinon, si le plat est juste payé ou en cours de préparation, le caissier annule la commande et rembourse le client. Une alerte sonore est déclenchée sur l'écran de la cuisine contenant le numéro et le contenu de la commande annulée si la commande est en cours de préparation et cette dernière est supprimée automatiquement de

l'affichage en cuisine.

Travail demandé

1. Identifier les acteurs du système **(2 pts)**
2. Modéliser le contexte dynamique du système pour **un** acteur de votre choix **(2 pts)**.
3. Elaborer le diagramme de cas d'utilisation fonctionnels **(8 pts)**.
4.

Annexe 3 : Examens 2011-2014

1. Examen ordinaire 2011-2012

On veut mettre en place un système d'informations pour gérer le bureau d'un avocat. On s'intéresse seulement à la gestion de la caisse. La gestion de la caisse est assurée conjointement par l'avocat et par son secrétaire. La caisse fait l'objet de deux opérations de bases : dépôt d'argent et dépense.

Quelle que soit l'opération, le secrétaire note sa date et heure, son montant (valeur). Par contre, lorsque l'opération correspond à un dépôt, il peut s'agir d'un payement fait par un client ; dans ce cas, le secrétaire note le nom et prénom du client. Le dépôt peut également être fait par l'avocat. Dans ce cas, le secrétaire mentionne la cause du dépôt, par exemple « caisse vide ». Pour les dépenses, il s'agit soit de remboursement fait pour un client soit d'achats divers (journaux, café, produits de nettoyage, etc.). Dans le premier cas, le secrétaire note le nom et prénom du client remboursé. Dans le deuxième cas, il note la liste des achats. Noter qu'on ne veut pas gérer la liste des clients à part entière.

Dans tous les cas, le système doit permettre de connaître à tout moment le solde réel de la caisse (ce que contient la caisse) en faisant la différence entre la somme des dépôts et la somme des dépenses.

Noter que la modification des informations sur les opérations déjà enregistrées dans la base de données est interdite et la même chose pour la suppression d'opérations. Si le secrétaire a fait un erreur sur le montant d'une opération, il identifie l'opération à corriger à partir de la liste, il note le montant de rectification et il fait le lien entre cette nouvelle opération et l'opération à corriger. La rectification doit être de même type que l'opération qu'elle corrige. Le montant de rectification peut dans le cas (et seulement dans ce cas) être positif ou négatif selon l'erreur faite.

Le système doit permettre à l'avocat, à travers un navigateur Web similaire à celui du secrétaire, d'accéder en lecture seule à la liste des opérations. En plus, il doit pouvoir effectuer des analyses pour connaître la fréquence de dépôts faits par l'avocat, le taux de remboursement faits pour les clients, etc.

Exercice 1 (5 pts: 2+3)

1. Sur quel type d'architecture matérielle reposera le système à mettre en place? Représenter cette architecture graphiquement.
2. Selon la description du cas, dans quel sens peut-on considérer l'avocat et le secrétaire comme des acteurs techniques?

Exercice 2 (15 pts : 6+5+4)

1. Elaborer le diagramme des cas d'utilisation illustrant les liens entre cas s'ils existent.
2. Choisir un seul cas à partir du diagramme des cas d'utilisation. Le cas ne doit pas être abstrait et doit permettre d'alimenter le système par des données. Pour ce cas choisi,

    - Elaborer le diagramme de classes participantes en mentionnant les attributs et les méthodes spécifiques;
    - Elaborer un diagramme de séquences illustrant l'interaction système - acteur(s).

2. Examen de rattrapage 2011-2012

Questions de cours (10 pts : 2+2+3+3)

1. Répondre par vrai ou faux et corriger dans le cas « faux » : *la relation d'inclusion permet d'inclure dans un cas les différents scénarios de son exécution*.
2. Compléter la phrase suivante par un seul mot: *dans le processus 2TUP, la capture des besoins fonctionnels a pour objectif de collecter et organiser les besoins liés au …..... des utilisateurs*.
3. Durant la capture des besoins fonctionnels, en plus de la description textuelle d'un cas d'utilisation, citer un diagramme permettant de mieux décrire un cas d'utilisation et expliquer son rôle.
4. Dans le processus de développement 2TUP, quel est l'avantage de la séparation dans les premières étapes entre les aspects fonctionnels et les aspects techniques?

Problème (10 pts : 5 + 5)

On veut développer un système de gestion d'un distributeur automatique de billets d'une banque. Chaque client de la banque possède une carte bancaire ayant un numéro et un code secret. A l'introduction de la carte, le système demande au client de saisir le code secret. Si le code est correct, le client accède aux fonctionnalités du système. Sinon, le client doit le ressaisir. Au bout de trois saisies erronées du code secret, le système bloque l'accès au système par cette carte. Le client doit contacter l'agent de la banque qui s'occupe de créer les numéros de carte et codes secrets pour qu'il réactive la carte et qu'il donne le code secret au client sur présentation d'une pièce d'identité. Dans ce cas, le système doit obliger le client à modifier son code à la prochaine utilisation de la carte.

Le distributeur permet aux clients de consulter le solde ou de retirer l'argent. Pour retirer l'argent, le client peut soit choisir un montant prédéfini, soit saisir le montant manuellement. Dans ce dernier cas, le client peut valider le montant ou le corriger. Le système vérifie si le montant dépasse l'argent disponible dans le distributeur ou si le solde du client n'est pas suffisant. Dans les deux cas, le système rejette l'interaction et redemande au client de la refaire. Dans le cas contraire (argent suffisant dans le distributeur et solde du client suffisant), le système délivre l'argent au client tout en affichant le détail de l'opération : date, solde avant opération, solde après opération.

Travail demandé

1. Identifier les cas d'utilisation fonctionnels et les cas d'utilisation techniques en les associant aux acteurs correspondants (illustrer par deux diagrammes).
2. Représenter le diagramme de contexte dynamique correspondant aux aspects fonctionnels du système.

3. Examen ordinaire 2012-2013

Exercice 1 (10 pts = 4 + 4 + 2) :

Une entreprise de vente de produits est organisée en une direction générale, 4 directions régionales (est, ouest, centre, sud) et une unité de vente dans chaque wilaya. L'entreprise veut

mettre en place un système permettant de saisir les ventes à chaque unité. Les données saisies chaque jour sont stockées temporairement dans un fichier Excel, qui alimente la base de données de sa direction régionale en fin de journée. En plus, chaque mois, les nouvelles ventes de chaque région sont ajoutées à la base de données centrale. Aussi, chaque directeur régional veut analyser les ventes de sa région pour connaître la meilleure wilaya. De même, le directeur général veut connaître les meilleurs produits vendus et les meilleures régions.

1. Elaborer le modèle de spécification du point de vue matériel,

2. Elaborer le modèle d'architecture en identifiant les composants.
3. Quelle modification apporter au deux modèles précédents pour permettre au directeur général d'effectuer ses analyses même en cours de déplacement?

Exercice 2 (10 pts = 3 + 3 + 4) :

La figure suivante montre le diagramme de classes global d'une entreprise de vente de produits **sur commandes**.

1. Détecter les anomalies contenues dans le diagramme ci-haut en structurant votre réponse selon le tableau suivant

Anomalie	Elément concerné	Correction

2. Donner le diagramme de classes global après correction des anomalies,
3. On s'intéresse au cas d'utilisation d'ajout d'une commande décrit comme suit: l'agent commercial renseigne les informations de la commande (numéro, date, etc), ensuite il associe l'ensemble des produits commandés à partir d'un catalogue de produits. Pour chaque produit, l'agent choisit un des fournisseurs qui fournissent le produit. Ce choix est basé sur deux critères qui sont mis à jour automatiquement: le prix du produits proposé et le délai de livraison. Une fois le fournisseur de chaque produit choisi, l'agent renseigne la quantité commandée et le prix de la vente (après application d'un pourcentage de bénéfice). Enfin, l'agent établit le lien entre la commande et le client qui est choisi à partir d'une liste.

   Sur la base du diagramme de classes corrigé de la question 2, élaborer le diagramme d'interaction pour développer ce cas d'utilisation.

4. Examen de rattrapage 2012-2013

Questions de cours (4 pts)

1. Compléter la phrase suivante: «Un(e) …. permet d'identifier les comportements anormaux et d'interrompre l'exécution d'un cas d'utilisation ».

2. Répondre par vrai ou faux et corriger si la phrase est fausse : «Un acteur fonctionnel est un acteur qui exécute les tâches liées au métier».

3. Quel est le mécanisme qui permet d'utiliser une classe d'analyse en dehors de sa catégorie ?

4. Quel est le lien entre un cas d'utilisation et un scénario d'exécution ?

Exercice 1 (10 = 6+4 pts)

On veut modéliser les aspects fonctionnels et techniques du service de rédaction d'un journal. Le service est composé de plusieurs journalistes, d'un rédacteur-en-chef, d'un infographe et du directeur du journal. Les articles sont écrits par des journalistes dans différents domaines (sport, politique, économie). Chaque article est mis en attente pour vérification par le rédacteur-en-chef pour s'assurer de la bonne qualité de rédaction. Si l'article est validé, le rédacteur-en-chef décide de la date de sa publication. Sinon, le journaliste est invité soit à modifier l'article, soit à le supprimer. A 15h de chaque jour de travail, l'ensemble des articles à paraître le lendemain doit être prêt. Sur la base de cet ensemble, l'infographe commence la composition du journal en choisissant l'endroit où placer chaque article. Si l'espace est insuffisant, l'infographe demande au journaliste concerné de raccourcir l'article. Une fois le journal composé, l'infographe génère une version PDF qu'il envoie au directeur pour validation. Le directeur peut demander de modifier la composition du journal. Une fois la composition finale validée, l'infographe publie le journal en ligne ainsi que la version PDF.

Les lecteurs peuvent accéder au contenu du journal gratuitement à partir du site Web du journal. Mais, pour télécharger la version PDF, chaque lecteur doit créer un compte et se connecter. Un compte de lecteur ne peut être fonctionnel qu'après sa validation par l'administrateur du journal. Chaque lecteur peut, après la lecture d'un article et des commentaires, ajouter un commentaire à l'article ou commenter des commentaires. Chaque journaliste peut lire les commentaires faits sur son article. Il peut supprimer les commentaires qui comportent des insultes. A la demande d'un journaliste, l'administrateur peut désactiver les commentaires d'un lecteur sur la base de son adresse IP ou supprimer les comptes de mauvais lecteurs. Enfin, l'administrateur procède chaque jour à la suppression des articles les moins lus depuis une semaine et de ceux qui datent de plus d'un mois.

1. Modéliser le diagramme de cas d'utilisations fonctionnels, en montrant les liens entre les cas.
2. Modéliser le diagramme de cas d'utilisation techniques.

Exercice 2 (6 = 3+3 pts)

On veut mettre en place un système d'informations pour gérer les sujets de master et de doctorat au niveau des universités et au niveau du ministère de l'enseignement supérieur. Après chaque soutenance, on saisit les informations sur le sujet soutenu à travers une application spécifique au niveau du sujet (master ou doctorat). Les données saisies sont stockées dans une seule base de données par département. A la fin de l'année universitaire, les données sur les sujets soutenus de l'année en cours sont extraites de la base de données de chaque département et fusionnées dans un fichier XML qui sera stocké au niveau du doyen de chaque faculté. Ce dernier peut analyser le contenu du fichier XML à travers une application unique à l'université et gérée par le rectorat.

A la fin de l'année, les fichiers XML sont exportés vers la deux bases de données centrales au niveau du ministère : une base de données sera utilisée par le service des sujets de master et

l'autre sera utilisée par le service des sujets de doctorat. Chaque service possède pour cela une application analytique plus complexe que celle des facultés.

1. Elaborer le modèle de spécification du point de vue matériel
2. Elaborer le modèle d'architecture.

5. Examen ordinaire 2013-2014

Questions de cours (6 pts : 1,5+1+1,5+1+1)

1. Choisir plusieurs réponses possibles : A quelle(s) étape(s) du processus 2TUP utilise-t-on le diagramme de classes UML ? (mauvaise réponse = - 0,25)

A. *Conception générique*. **B.** *Etude préliminaire*. **C.** *Capture des besoins fonctionnels*.

D. *Capture des besoins techniques*. **E.** *Analyse*.

2. Choisir une seule bonne réponse : Quel est le sens d'un scénario terminal ?

A. L'*exécution du cas d'utilisation se termine normalement*. **B.** *L'exécution du cas se termine avec erreur*. **C.** *Le cas de ne peut s'exécuter*. **D.** *La prochaine exécution du cas se termine par une erreur*.

3. Répondre par vrai ou faux et corriger si nécessaire : « Lors du développement du modèle dynamique d'un système, toutes les classes du diagramme de classes sont impliquées dans l'exécution de chaque cas d'utilisation. »
4. Compléter la phrase suivante : « un framework technique contient un ensemble de …..... techniques nécessaires (à la / au) …..... du système.

4. Dans quel cas, un acteur fonctionnel peut-il figurer dans le diagramme de classes d'un système à développer ?

Exercice 1 (8 pts = 4+4) : On veut développer un système pour gérer les données des élections législatives. Les élections sont gérées au niveau du ministère de l'Intérieur, des Wilayas et des bureaux de vote. Chaque wilaya dispose d'un ensemble de fiches de candidats en format Word. Ces fiches sont accessibles via un serveur de fichier à partir de terminaux mis à la disposition des électeurs dans chaque bureau de vote. Après fermeture des bureaux de vote à 20h00, un agent de bureau saisit les données de vote sur un terminal à travers une application de saisie centralisée par wilaya. Les données saisies sont stockées directement dans la base de données de la wilaya. À 22h00, les données de chaque wilaya sont répliquées dans la base de données du ministère. Pour des mesures de sécurité, une copie de ces données est stockée dans une base de données clone, hébergée sur une machine séparée. Enfin, les résultats des élections peuvent être accessibles soit au niveau de chaque wilaya soit niveau du ministère (résultat global) à travers des applications spécifiques.

1. Elaborer le modèle de configuration matérielle à l'aide d'un diagramme de déploiement UML.

2. Elaborer le modèle de configuration logicielle à l'aide d'un diagramme de composants UML.

Exercice 2 (6 pts)

On veut développer une application pour gérer une superette. Les achats sont saisis par plusieurs caissiers avec un système de rotation par caisse. Avant de commencer le travail, chaque caissier doit d'abord s'identifier. L'attribution et la gestion des identifiants et des codes d'accès se fait par le gérant de la superette à travers l'espace pesonnel de ce dernier. Avant d'ouvrir la caisse, le caissier doit s'assurer que l'argent liquide disponible correspond au solde affiché.

A la présentation de l'acheteur devant la caisse, le caissier enregistre chaque article acheté par le biais du lecteur de code barre. Dans certains cas, lorsque le code barre n'est pas fonctionnel ou s'il est illisible le caissier bascule vers la saisie manuelle du produit. La lecture par code barre est le mode par défaut. Le caissier peut à tout moment mettre en pause l'enregistrement des articles, par exemple pour reseigner un autre acheteur sur le pris d'un produit, puis il reprend l'enregistrement.

Après la saisie de tous les renseignements d'un acheteur, ce dernier procède au payement. Le mode de paiement par défaut est le payement cash (argent liquide). L'acheteur peut également payer en utilisant une carte de crédit. Dans ce cas, le caissier bascule vers le payement par carte. L'acheteur introduit sa carte dans un lecteur de carte lié au réseau de banques. L'acheteur saisit ensuite le code secret et valide. Le numéro de carte et le code introduit sont lus par le lecteur : si le code est correct, le caissier valide le payement. Sinon, après trois tentatives échouées du code de la carte bancaire, et si l'acheteur ne dispose pas d'argent, le montant des achats est sauvegardé comme « dette ». Par mesure de garantie (en cas d'acheteur inconnu), le numéro de la carte bancaire du client est à nouveau lu pour être associé à la dette et le caissier fixe un délai de payement. Une fois le payement effectué, la dette est supprimée.

- Elaborer le diagramme de cas d'utilisation techniques de ce système en illustrant les éventuels liens entre les cas.

6. Examen de rattrapage 2013-2014

On veut développer une application informatique pour la gestion d'un agenda électronique d'un responsable. L'agenda doit permettre à ce dernier d'organiser ses rendez-vous et activités qu'on appellera *événements*. L'agenda est tenu par le responsable ou pas sa secrétaire. Les événements sont affichés sur un calendrier croisé : en colonnes les jours et en lignes les heures de 00h00 à 23h. A la demande du responsable, la secrétaire ajoute des événements en choisissant la date, l'heure et la durée. Pour cela, elle consulte d'abord le calendrier pour choisir un créneau convenable mais elle doit avoir la confirmation du responsable avant de valider l'ajout. Si le responsable est disponible, la secrétaire demande la confirmation verbalement. Sinon, elle met en attente le choix et envoie un mail au responsable. A la lecture du mail, le responsable consulte l'agenda et confirme les choix convenables mis en attente de la secrétaire. Pour les choix non convenables, il peut demander à la secrétaire de les modifier ou il peut lui-même les modifier et les confirmer directement.

Dans plusieurs cas, lorsque les événements sont urgents ou ont des dates et heures fixes, la secrétaire peut modifier des créneaux existants par d'autres plus urgents ou fixes. Dans ce cas,

l'événement urgent ou fixé remplace l'événement existant et ce dernier sera déplacé à un autre créneau. La programmation d'un événement fixé n'attend pas la confirmation du responsable. Par contre, celle des événements urgents et les déplacements d'événements existants doivent avoir la confirmation du responsable.

Dans certains cas, le responsable peut décider d'annuler des évènements et confie à sa secrétaire la tâche d'apporter ces annulations à l'agenda. Enfin, au début de chaque journée, la secrétaire met à jour le statut des événements de la veille en mentionnant sa réalisation ou pas.

Travail demandé:

1. Elaborer le diagramme de cas d'utilisation fonctionnels en indiquant éventuellement les liens entre les cas.
2. Elaborer le diagramme de contexte dynamique de ce système.

7. Examen ordinaire 2014-2015

Questions de cours (4 pts / 1+1+2)

1. Compléter la phrase suivante par un seul mot : « durant l'étape d'analyse du processus 2TUP, un bon découpage en catégorie est celui qui ……. Les dépendances entre les catégories. »

2. Choisir une ou plusieurs bonnes réponses : dans la capture des besoins techniques, un besoin technique peut être assuré par

A. du matériel B. un cas d'utilisation C. un composant métier D. du logiciel

3. Expliquer la différence d'utilisation d'un diagramme de séquences pour décrire les cas d'utilisation durant l'étape de capture de besoins fonctionnels et durant l'étape d'analyse.

Exercice 1 (8 pts / 4+4)

Une école de formation de haut niveau organise chaque année un concours pour l'accès au doctorat au profit des étudiants en Master des 5 meilleures universités de l'année. L'école prend de chaque université trois (03) étudiants sur un total des dix (10) meilleurs classés qui passent le concours.

Le jour du concours, les dix candidats de chaque université sont regroupés dans une salle et chacun sera affecté à un terminal Web. Le sujet d'examen est électronique et s'affiche sous la forme d'une application Web interactive servie à partir d'une machine située au niveau de l'école. Les réponses se font à travers l'application. Une fois terminé, le candidat valide ses réponses qui seront enregistrées sous la forme d'un fichier avec l'extension « .frp » sur une machine unique située à l'université. En fin de journée, les fichiers des cinq universités sont répliqués tels quels sur une machine unique au niveau de l'école. Le lendemain, le responsable des concours lance les corrections automatiques sur un PC de correction lié au PC qui héberge les fichiers FRP.

Les résultats (candidats retenus, classement, corrigé-type, copies corrigées) sont stockés en

format PDF sur le PC de correction. Ces fichiers sont accessibles via Internet à travers le site Web de l'école. Aussi, un module applicatif permet à chaque candidat d'accéder à travers le site de l'école à sa copie corrigée. Pour cela, il doit d'abord introduire le code et le mot de passe que son université lui a fourni.

1. Elaborer le schéma de configuration matérielle à l'aide d'un diagramme de déploiement.
2. Elaborer le schéma de configuration logicielle à l'aide d'un diagramme de composants.

Exercice 2 (8 pts)

Une agence de dépôt de demandes de visa veut informatiser la gestion de son service d'accueil. Le service fonctionne comme suit :

A l'entrée du service, un agent d'accueil vérifie l'identité du demandeur de visa (visiteur). Ce dernier doit présenter le coupon de rendez-vous sur lequel un code-barres est imprimé et aussi son passeport. Si le code-barres n'est pas lisible, l'agent d'accueil saisit le numéro de passeport sur un PC et vérifie que le visiteur a réellement un rendez-vous.

Par la suite, le visiteur dépose tous ses objets (portable, montre, etc.) dans un bac et le fait passer par un scanner contrôlé par un agent de contrôle. Ce dernier doit reconnaître tous les objets. Si un objet n'est pas reconnu, l'agent utilise un appareil manuel pour vérifier que l'objet n'est pas dangereux. Les objets du visiteur sont conservés et ce dernier se présente devant l'agent de vérification des documents. Dans le cas où des photocopies de documents manquent, le visiteur doit les reproduire en utilisant un photocopieur-scanner : le mode par défaut est la photocopie, mais si la qualité de la photocopie n'est pas bonne, le visiteur peut scanner puis imprimer ses documents.

Par la suite, si le dossier est complet, le visiteur se place devant un PC pour remplir le formulaire. Il a le choix de saisir ses données directement puis imprimer le formulaire rempli ou d'imprimer le formulaire vide et le remplir manuellement. Une fois le formulaire rempli, le visiteur l'ajoute au dossier, et retire un ticket d'un distributeur puis se dirige vers la salle d'attente pour attendre son tour.

***Travail demandé* :**

- Identifier les acteurs techniques de ce système. Pour chaque acteur, préciser s'il s'agit d'un acteur purement technique et préciser ses responsabilités métier dans le cas contraire.
- Elaborer le diagramme de cas d'utilisation techniques.

8. Examen de rattrapage 2014-2015

Exercice 1 : Questions de cours (10 pts : 2*5)

1. Répondre par vrai ou faux et corriger si nécessaire: *la relation d'inclusion permet d'inclure dans un cas les différents scénarios de son exécution*.
2. Compléter la phrase par les mots qui manquent : « *A l'étape de développement du modèle dynamique (analyse), le système est remplacé par un ensemble de(') ……..... qui interagissent entre eux ou avec le(s) ……..... du système »*.

3. Durant la capture des besoins fonctionnels, en plus de la description textuelle d'un cas d'utilisation, citer un diagramme permettant de mieux décrire un cas d'utilisation et expliquer son rôle.
4. Parmi les phrases suivantes, quelles sont celles qui traduisent des besoins opérationnels: ***a.*** Je veux accélérer les temps de réponse, ***b.*** je veux connaître les clients qui n'ont pas encore payé, ***c.*** Je veux afficher les prix sans virgule. ***D.*** Je veux archiver les données non utilisées.
5. Choisir une seule bonne réponse : « l'organisation d'un diagramme de cas d'utilisation permet d'identifier : ***a.*** les acteurs en plus ou en moins, ***b.*** les messages entre les cas, ***c.*** les liens entre les cas, ***d.*** les liens entre les acteurs ».

Exercice 2 : Problème (10 pts : 5 + 5)

On veut développer un système de gestion d'un distributeur automatique de billets d'une banque. Chaque client de la banque possède une carte bancaire ayant un numéro et un code secret. A l'introduction de la carte, le système demande au client de saisir le code secret. Si le code est correct, le client accède aux fonctionnalités du système. Sinon, le client doit le ressaisir. Au bout de trois saisies erronées du code secret, le système bloque l'accès au système par cette carte. Le client doit contacter l'agent de la banque qui s'occupe de créer les numéros de carte et codes secrets pour qu'il débloque101 la carte et qu'il donne le code secret au client sur présentation d'une pièce d'identité. Dans ce cas, le système doit obliger le client à modifier son code à la prochaine utilisation de la carte.

Le distributeur permet aux clients de consulter le solde ou de retirer l'argent. Pour retirer l'argent, le client peut soit choisir un montant inférieur, soit saisir le montant manuellement. Dans ce dernier cas, le client peut valider le montant ou le corriger. Le système vérifie si le montant dépasse l'argent disponible dans le distributeur ou si le solde du client n'est pas suffisant. Dans les deux cas, le système rejette l'interaction et redemande au client de la refaire. Dans le cas contraire (argent suffisant dans le distributeur et solde du client suffisant), le système délivre l'argent au client tout en affichant le détail de l'opération : date, solde avant opération, solde après opération.

Travail demandé

1. Identifier les cas d'utilisation fonctionnels et les cas d'utilisation techniques en les associant aux acteurs correspondants (illustrer par deux diagrammes).
2. Représenter le diagramme de 101contexte dynamique correspondant aux aspects fonctionnels du système.