CHAPITRE 2 : ANALYSE ET CONCEPTION DU PROJET

2.1 Introduction

Pour chaque développement d'un projet, on doit toujours passé par la phase de conception. Cette phase nous permettra de délimiter le projet mais aussi de déterminer et analyser chaque fonctionnalité qui le composera. La phase de conception nous aidera à donner une fondation stable et solide à notre projet qui s'intitule « UniSphere » et nous permettra de minimiser les problèmes que nous pourrons rencontrés lors de sa réalisation. De ce fait dans ce chapitre, nous allons expliquer la méthode de conception que l'on va utiliser afin de concevoir le projet.

2.2 Notation Unified Modeling Language (UML)

Le développement de notre plateforme UniSphere nécessite une conception minutieuse afin de minimiser les problèmes. Pour cela nous utiliserons l'Unified Modeling Language ou Langage de Modélisation Unifié durant la phase de conception.

2.2.1 Définition de la notation d'Unified Modeling Language (UML)

L'UML (Unified Modeling Language) est un langage universel de modélisation de logiciel construite à l'aide d'objets et de notation, outil de communication visuelle [2.01]. La notation UML est fréquemment employer lors de la phase de conception d'un projet avec ces modélisations visuelle qui offrent différentes perspectives d'un système d'un projet.

2.2.2 Origine de l'UML

La notation de l'UML n'est pas apparue sans préambule. Il est né dans les années 1990 de la fusion de trois méthodes de développement orienté objet : la méthode Booch de Grady Booch, OMT (Object Modeling Technique) de James Rumbaugh et OOSE (Object-Oriente Software Engineering) de Ivar Jacobson [2.02]. Plus précisément, les Three Amigos du génie logiciel, comme on les appelait alors, avait élaboré d'autres méthodologies. Ils se sont associés pour apporter plus de clarté aux programmeurs en créant de nouvelles normes. La collaboration entre Grady, Booch et Rumbaugh a renforcé les trois méthodes et a amélioré le produit final. Les efforts de ces penseurs ont abouti à la publication des documents UML 0.9 et 0.91 en 1996. Il est rapidement devenu évident que des sociétés comme Microsoft, Oracle et International Business Machines (IBM) voyant l'UML comme un élément critique pour leur développement futur. Elles ont donc mis en place des ressources, accompagnées en cela par de nombreuses autres sociétés et personnes, permettant de développer un langage de modélisation complet. Les

Three Amigos ont publié The Unified Modeling Language User Guide en 1999, qui fut suivi d'une mise à jour comportant des informations sur l'UML 2.0 en 2005 [2.03].

Les Trois Amigos et les méthodes qu'ils ont créées :

- Grady Booch depuis 1981, travaillait à la mise au point de formalismes graphiques pour représenter le développement de programme Orienté Objet (OO). Sa méthode Object Oriented Design (OOD) fut conçue à la demande du ministère de la Défense des États-Unis. Il avait mis au point des diagrammes de classe et d'objet, des diagrammes d'état-transition, et d'autres diagrammes de processus, pour mieux visualiser les programmes pendant leur exécution. Ces diagrammes devaient accompagner et améliorer l'organisation et la structure de programmes écrits en ADA et C++ [2.04].
- James Rumbaugh, alors à la Général Electric, fut l'auteur d'un formalisme graphique, précédant et en cela anticipant UML, appelé Object Modeling Technique (OMT). C'est d'OMT qu'UML s'est indéniablement le plus inspiré. Cette inspiration fut puisée dans les langages à objet mais également dans la modélisation conceptuelle appliquée à l'analyse et au stockage des données. La plupart des diagrammes importants faisant partie d'UML se retrouvaient déjà dans OMT, comme le diagramme de classe et autres diagrammes dynamiques et fonctionnels. C'est de fait Rumbaugh qui, chez IBM aujourd'hui, est le plus impliqué dans la maintenance et l'évolution de son bébé. Il accorde aussi énormément d'importance, au-delà de l'utilisation de ces diagrammes, au suivi d'une méthodologie décomposée en plusieurs phases, de l'analyse à l'implémentation, phases qui, au lieu d'être complètement séquentielles, se recouvrent en partie. Un développement logiciel devrait plutôt se dérouler comme une succession de petites itérations, de courte durée, toutes intégrant de l'analyse et du développement, mais le poids de l'analyse et du développement s'inversant graduellement vers la fin. On retrouve ces directives méthodologiques dans RUP (Rational Unified Process) et dans la programmation agile [2.04].
- Ivar Jacobson, auteur de la méthode Object Oriented Software Engineering (OOSE) est surtout connu pour avoir recentré l'analyse et le développement informatique sur les besoins humains et, en conséquence, pour l'apport dans UML de la notion de cas d'utilisation et du diagramme correspondant. Il s'est toujours intéressé à la nécessité première d'une bonne représentation du cahier des charges de l'application, ainsi que d'une bonne stratégie de développement, également basée sur une succession de phases courtes, dans lesquelles l'analyse et le développement varient en importance durant la progression du projet. C'est

essentiellement à lui que l'on doit RUP, la méthodologie de développement logiciel, qui accompagne souvent l'apprentissage d'UML [2.04].

La version de l'UML 1.0 a été adopté comme standard par l'Object Management Group (OMG) en janvier 1997. Des versions successives ont ensuite été validées, la dernière en date étant l'UML 2.5.1. [2.05].

L'Object Management Group (OMG) est un consortium de normes technologiques international, ouvert à tous et sans but lucratif créé en 1989. Les normes de l'OMG sont appliquées par les entreprises, les utilisateurs, le monde universitaire et les agences gouvernementales. Les groupes de travail de l'OMG développent des normes d'intégration aux entreprises pour un grand nombre de technologies et de secteurs industriels. Les normes de modélisation de l'OMG, dont l'UML et le Model Driven Architecture (MDA), permettent la conception, l'exécution et la maintenance de logiciels et d'autres processus d'une façon visuellement efficace [2.03].

L'OMG (Object Management Group) supervise la définition et la maintenance des spécifications UML. Cette surveillance donne aux ingénieurs et aux programmeurs la possibilité d'utiliser un langage à des fins multiples pendant toutes les phases du cycle de vie du logiciel, quelle que soit la taille du système concerné. L'OMG (Object Management Group) supervise la définition et la maintenance des spécifications UML. Cette surveillance donne aux ingénieurs et aux programmeurs la possibilité d'utiliser un langage à des fins multiples pendant toutes les phases du cycle de vie du logiciel, quelle que soit la taille du système concerné [2.03].

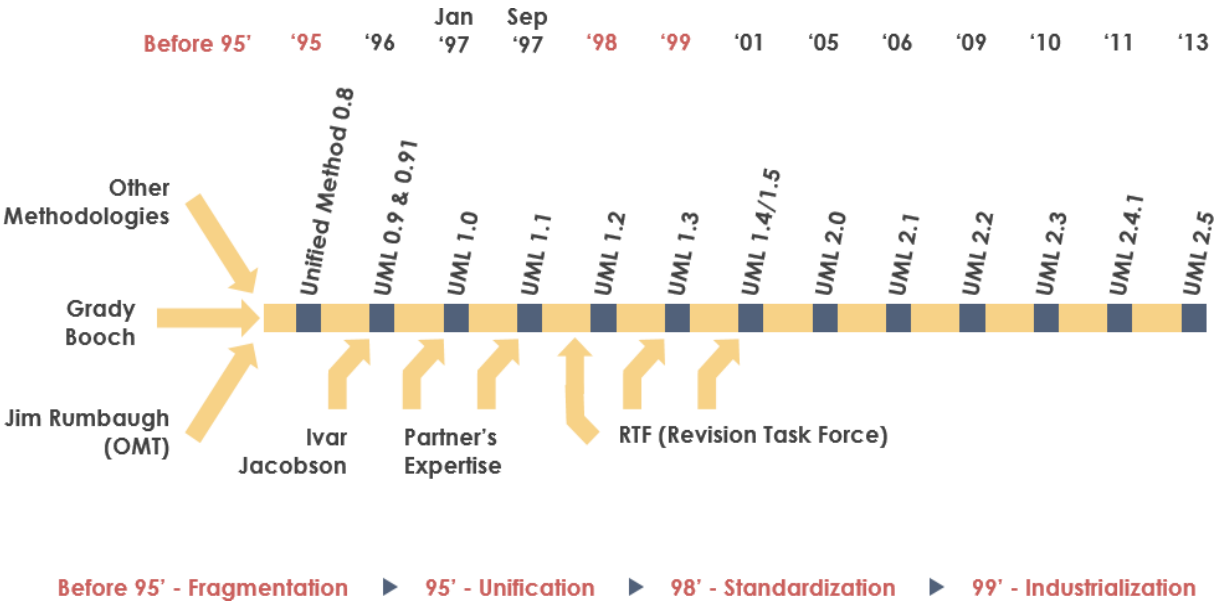

Figure 2.01 : *Ordre Chronologique de l'évolution de l'UML* [2.06]

2.2.3 Objectifs de l'UML

L'Unified Modeling Language a pour objectif principale de présenter un langage de modélisation permettant de représenter de manière visuelle et simplifier la structure et le comportement de système complexe. Facilité la communication entre les différents acteurs du projet comme les développeurs, concepteurs, chef de projet et le client. L'UML permet de réduire l'imprécision et d'amélioré la précision des besoins du client.

L'Object Management Group définit les objectifs de l'UML comme suit :

- Fournir aux concepteurs de systèmes, ingénieurs logiciels et développeurs de logiciels des outils pour l'analyse, la conception et la mise en œuvre de systèmes logiciels, ainsi que pour la modélisation de processus métier et d'autres processus similaires. [2.03]
- Faire progresser l'industrie en permettant l'interopérabilité des outils de modélisation visuelle orientés objet. Toutefois, pour permettre un échange significatif d'informations de modèles entre outils, il est nécessaire de trouver un accord sur la sémantique et la notation. [2.03]

L'UML réponds aux exigences suivantes :

- Fixer une définition formelle d'un métamodèle basé sur une norme Meta-Object Facility (MOF) commune qui spécifie la syntaxe abstraite de l'UML. La syntaxe abstraite définit l'ensemble des concepts de modélisation UML, leurs attributs et leurs relations, ainsi que les règles permettant d'associer ces concepts afin de créer des modèles UML partiels ou complets. [2.03]

- Fournir une explication détaillée de la sémantique de chaque concept de modélisation UML. La sémantique définit, d'une façon indépendante de la technologie, comment les concepts UML doivent être mis en œuvre par les ordinateurs. [2.03]

- Spécifier des éléments de notation lisibles par l'homme pour représenter chaque concept de modélisation UML, ainsi que les règles pour les combiner au sein d'une grande variété de diagrammes correspondant à différents aspects des systèmes modélisés. [2.03]

- Définir des moyens grâce auxquels les outils UML peuvent être mis en conformité avec cette spécification. Ceci est pris en charge (dans une spécification distincte) par une spécification XML des formats d'échange de modèles correspondants (XMI) qui doivent être réalisés par des outils conformes. [2.03]

2.2.4 Avantages de l'UML

L'UML offre de nombreux avantages lors de la phase de conception d'un quelconque projet informatique, qu'ils s'agissent d'améliorer le flux de travail d'un ingénieur logiciel ou de rationaliser la communication au sein d'une équipe. Passons en revue tous les avantages :

- Simplifier les idées et les systèmes complexes : les diagrammes UML simplifient visuellement les idées abstraites et les systèmes logiciels complexes, facilitant ainsi la collaboration entre les ingénieurs logiciels [2.07].
- Visualisation de codes complexes : transformer des lignes de code compliquées en diagrammes visuels rend le développement de logiciels plus simple. Les diagrammes UML donnent une image claire de la manière dont les parties du code sont liées et fonctionnent ensemble, ce qui permet de gagner du temps et de réduire la confusion [2.07].
- Permet aux développeurs d'être sur la même longueur d'onde : UML est un langage visuel standard qui aide les membres de l'équipe à mieux communiquer entre eux, quels que soient leur langue et leur stade de développement. [2.07].
- Permet d'avoir une vue d'ensemble : au cours du processus de développement du logiciel, le fait de pouvoir se référer à un diagramme UML aide les développeurs à rester concentrés sur la conception globale et les objectifs du projet. [2.07].
- Ideal pour les explications non techniques : en plus d'aider les ingénieurs en informatiques, les diagrammes aident les propriétaires de produits, les gestionnaires et les parties prenantes à comprendre les processus et les fonctionnalités des logiciels. Ils comblent le fossé entre les membres techniques et non techniques de l'équipe, favorisant ainsi un meilleur travail d'équipe. [2.07].
- Améliore la collaboration entre les équipes : tous les programmeurs ne comprennent pas et ne se spécialisent pas dans le même type de code et de langage de programmation. En utilisant une notation standard, les diagrammes UML permettent à des programmeurs ayant des compétences différentes de travailler ensemble de manière efficace [2.07].

2.2.5 Les diagrammes dans l'UML

Afin de pouvoir concevoir un projet à partir de l'UML, plusieurs diagrammes sont utilisés pour représenter visuellement la structure du système du projet, les communications entre les acteurs et le système et présenter de façon concises le projet au client.

L'UML possède 14 types de diagrammes qui sont repartis dans 2 catégories : le diagramme structurel ou statique et le diagramme comportementaux ou dynamique.

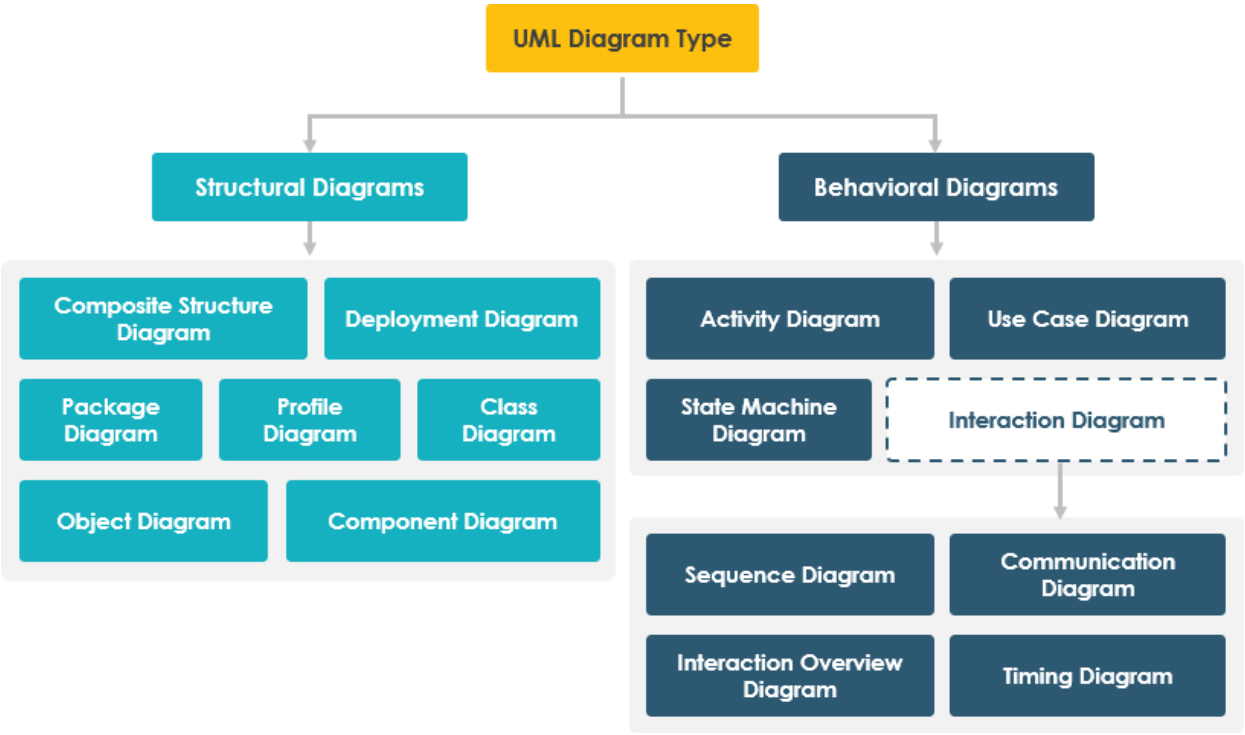

Figure 2.02 : *Hiérarchie Schématique de UML* [2.06]

2.2.5.1 Les diagrammes structurels ou statiques

Les diagrammes structurels sont des représentations visuelles des différentes éléments qui constitue un système et leurs relations statique dans un projet. En d'autres termes, les diagrammes UML structurels, comme leur nom l'indique, illustrent la structure d'un système, notamment les classes, les objets, les paquetages, les composants, etc…, et les relations entre ces éléments [2.08].

- Diagramme de classe : c'est l'une des diagrammes le plus utilisé dans l'UML car ils exposent la structure statique d'un système, notamment les classes, leurs attributs et leurs comportements, ainsi que les liens entre chacune d'elles. Une classe est représentée par un rectangle contenant trois (3) compartiments emplis verticalement. Le compartiment supérieur contient le nom de la classe et est indispensable, tandis que les deux compartiments inférieurs fournissent des détails sur les attributs et les opérations ou comportements de la classe [2.08].

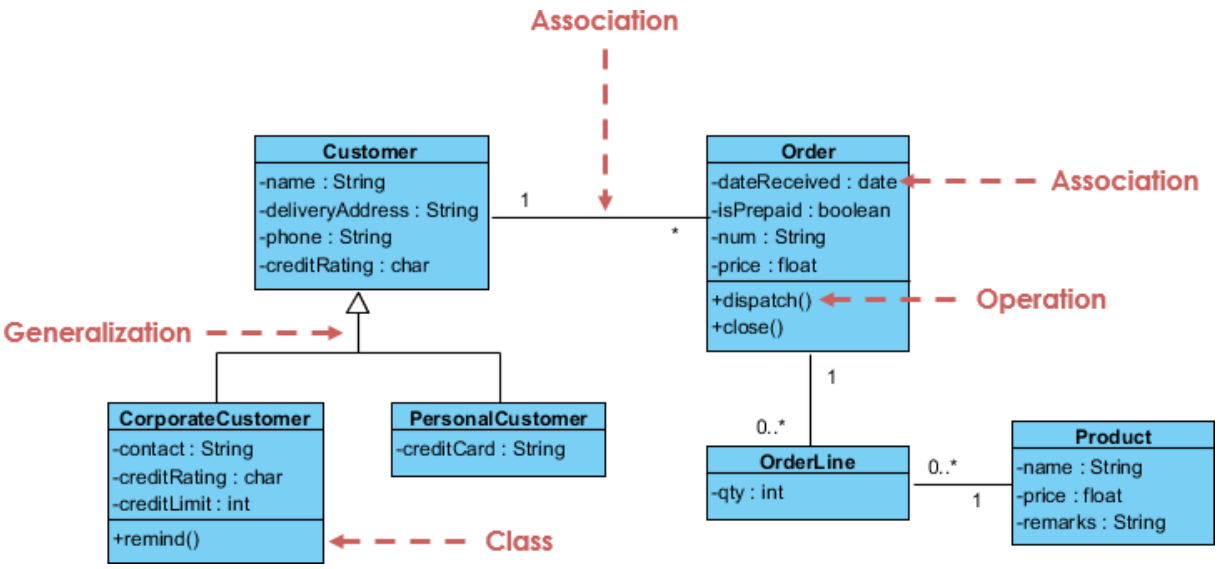

Figure 2.03 : *Exemple de diagramme de Classe* [2.06]

- Diagramme d'objet : les diagrammes d'objets présentent des exemples de structure de données à un moment spécifiques [2.08]. Ces diagrammes sont utilisés comme scenarios de test à un diagramme de classes d'une structure d'un système.

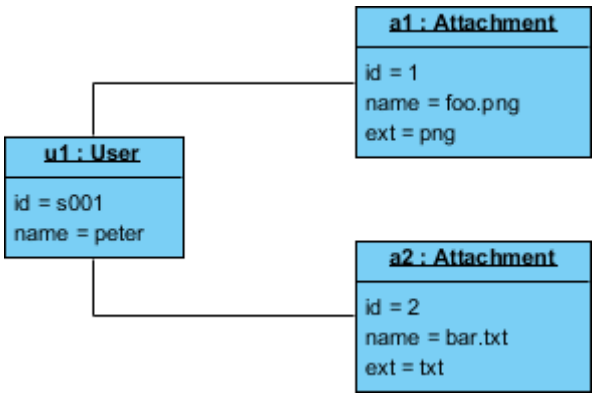

Figure 2.04 : *Exemple de diagramme d'objets* [2.06]

- Diagramme de composant : c'est une version plus spécifique du diagramme de classes, et les mêmes règles de notation s'appliquent aux deux. Ce type de visuel découpe un système complexe en composants de taille réduite et représente les interactions entre ces derniers [2.08].

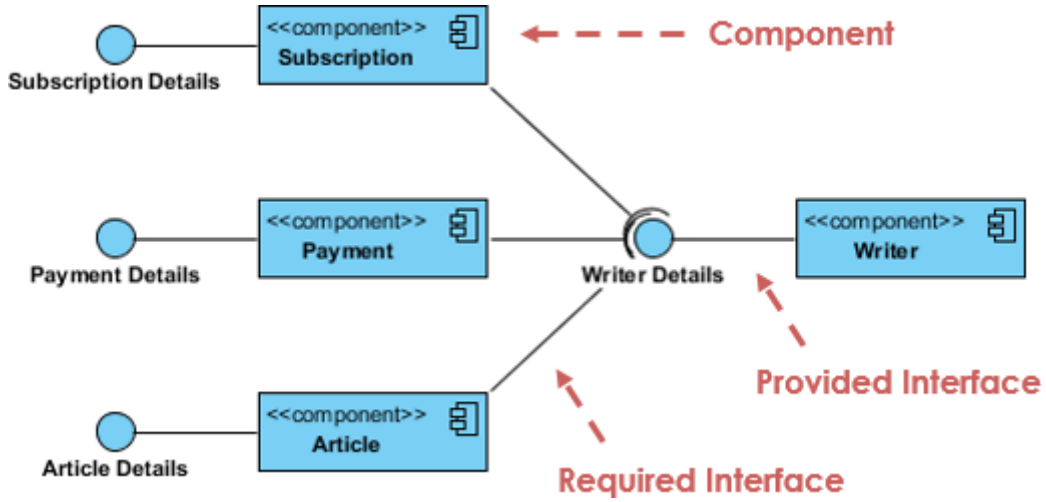

Figure 2.05 : *Exemple de diagramme de composant* [2.06]

- Diagramme de déploiement : c'est la manière dont les logiciels sont déployés sur les composants matériels d'un système. Ces visuels sont particulièrement utiles pour les ingénieurs système et ils illustrent généralement les performances, l'évolutivité, la maintenabilité et la portabilité. Lorsque les composants matériels sont représentés les uns par rapport aux autres, il est plus facile de suivre l'ensemble des infrastructures informatiques et de s'assurer que tous les éléments sont pris en compte lors d'un déploiement [2.08].

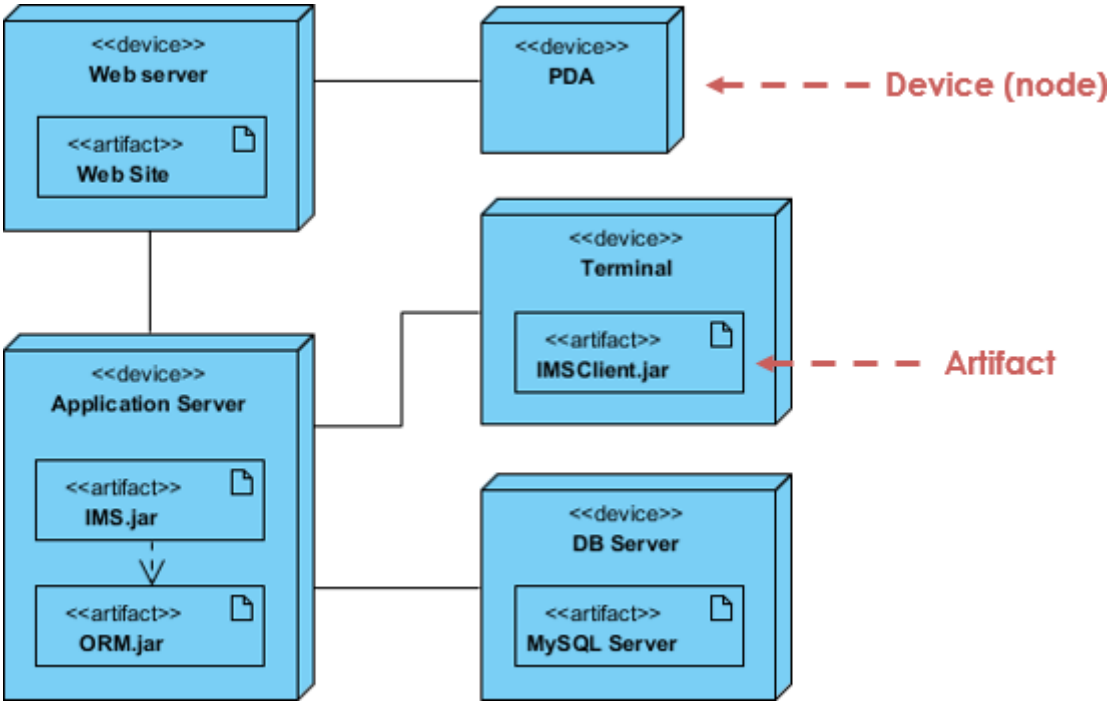

Figure 2.06 : *Exemple de diagramme de déploiement* [2.06]

- Diagramme de paquetage : ces diagrammes sont utilisés pour illustrer les dépendances entre les différents paquetages d'un système. Ces derniers, représentés sous la forme d'un dossier de fichiers, organisent en groupes les éléments de modèle, tels que les cas d'utilisation ou les classes [2.08].

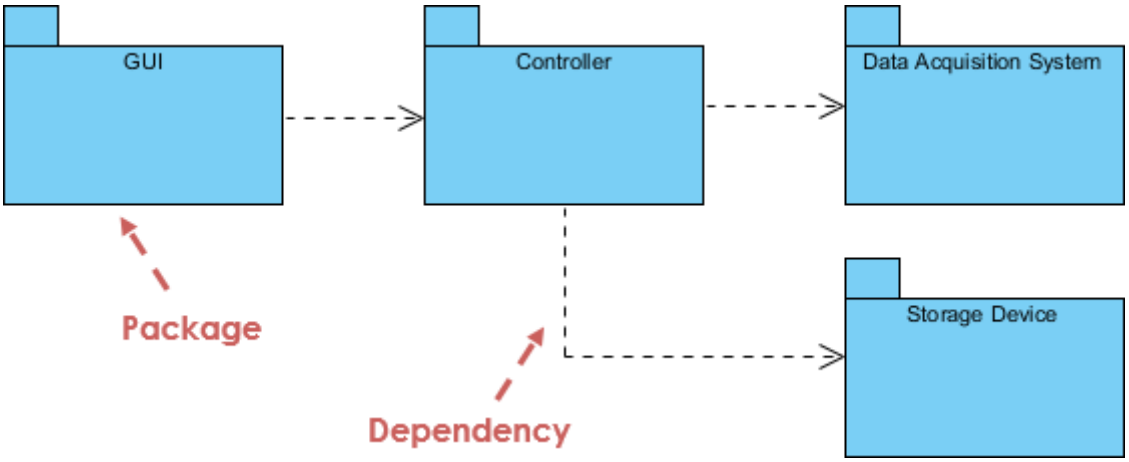

Figure 2.07 : *Exemple de diagramme de paquetage* [2.06]

- Diagramme de structure composite : ce sont des plans de la structure interne d'un classifieur. Ils peuvent également être utilisés pour illustrer le comportement d'une collaboration ou les interactions du classifieur avec son environnement par le biais des ports. Ils permettent de représenter facilement les composants internes de tout type d'équipement pour mieux en comprendre le fonctionnement [2.08].

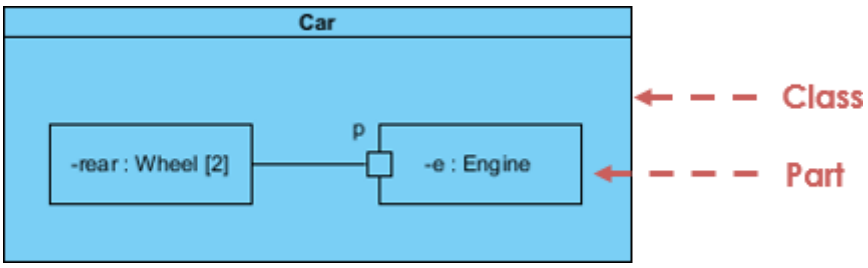

Figure 2.08 : *Exemple de diagramme de structure composite* [2.06]

- Diagramme de profil : récemment ajouté à UML 2.0, les diagrammes de profil sont uniques et rarement utilisés dans les spécifications. Un diagramme de profil est mieux compris comme un mécanisme d'extensibilité pour personnaliser les modèles UML pour des domaines et des plates-formes spécifiques.

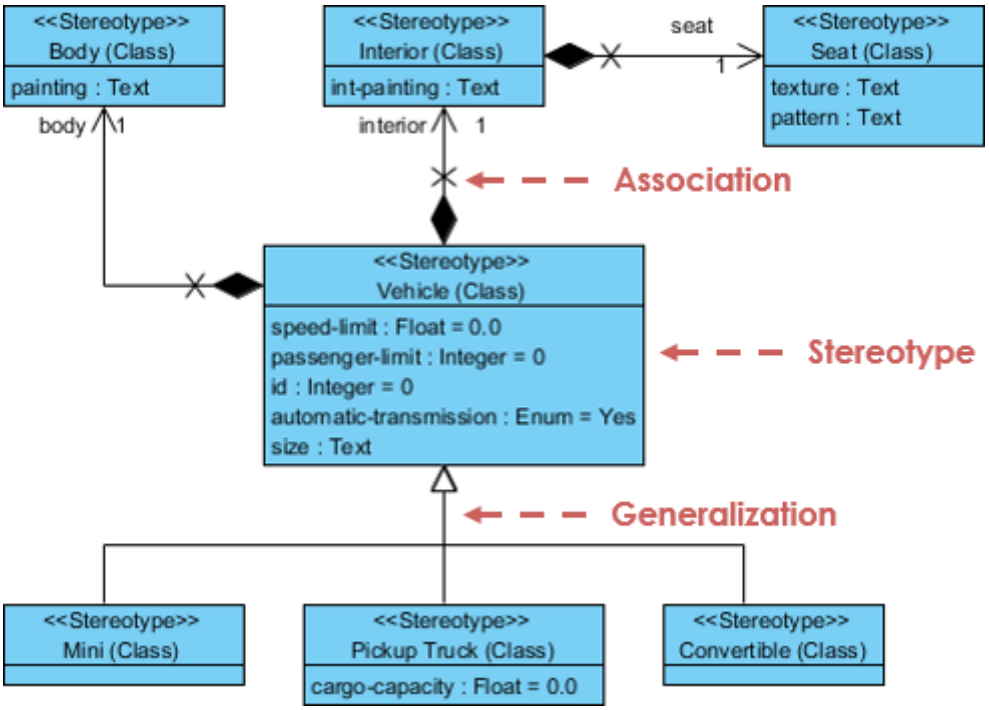

Figure 2.09 : *Exemple de diagramme de profils* [2.09]

2.2.5.2 Les diagrammes comportementaux ou dynamiques

Les diagrammes comportementaux sont des représentations visuelles des comportements et communication entre les différents éléments et le système d'un projet. En d'autres termes, Ces diagrammes UML représentent la manière dont le système se comporte et interagit avec lui-même et avec les utilisateurs, les autres systèmes et les autres entités [2.08].

- Diagramme de cas d'utilisation : les diagrammes de cas d'utilisation modélisent la manière dont les utilisateurs, représentés sous formes de figurine appelées « acteurs », interagissent avec le système. Ce type de diagramme UML est une vue d'ensemble des relations entre les acteurs et les systèmes, ce qui en fait un excellent outil pour présenter un système à un public non technique [2.08].

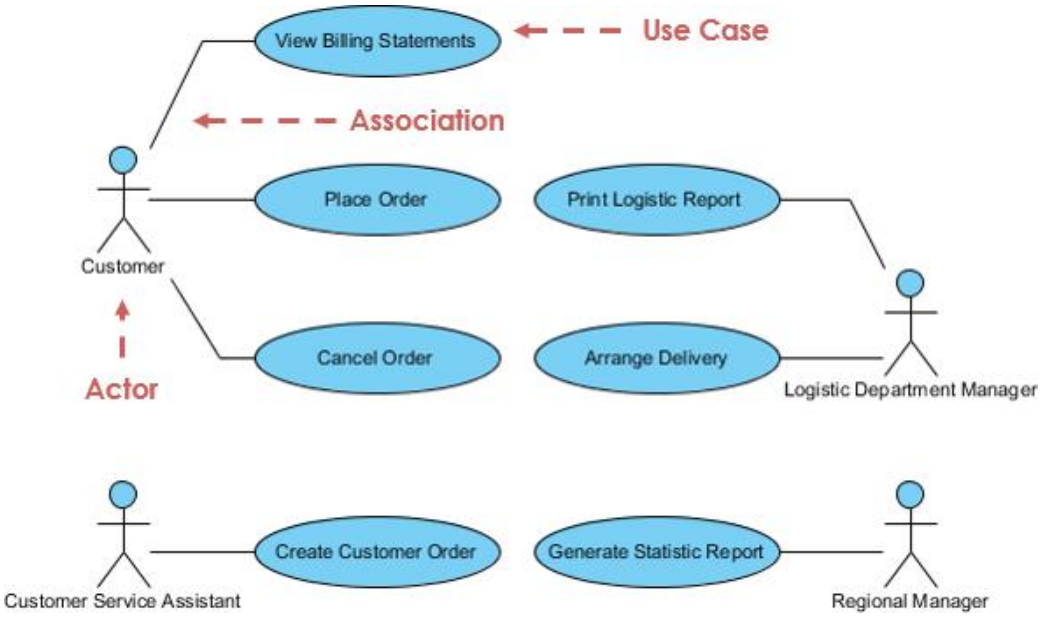

Figure 2.10 : *Exemple de diagramme de cas d'utilisation* [2.06]

- Diagramme d'activité : ces diagrammes représentent les étapes réalisées dans un cas d'utilisation. Les activités peuvent être séquentielles, ramifiées ou simultanées. Ce type de diagramme UML est utilisé pour montrer le comportement dynamique d'un système, mais il peut également être utile dans la modélisation des processus métier [2.08].

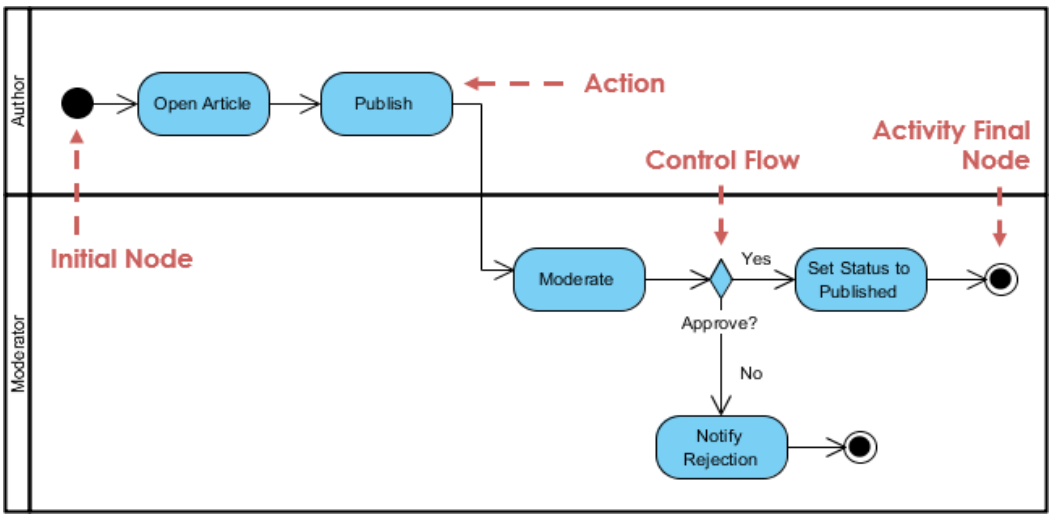

Figure 2.11 : *Exemple de diagramme d'activité* [2.06]

- Diagramme d'état-transition : ils décrivent les états et les transitions. Les états correspond aux différentes combinaisons d'informations qu'un objet peut contenir, et ce type de diagramme UML permet de visualiser tous les états possibles et la manière dont l'objet passe d'un état à l'autre [2.08].

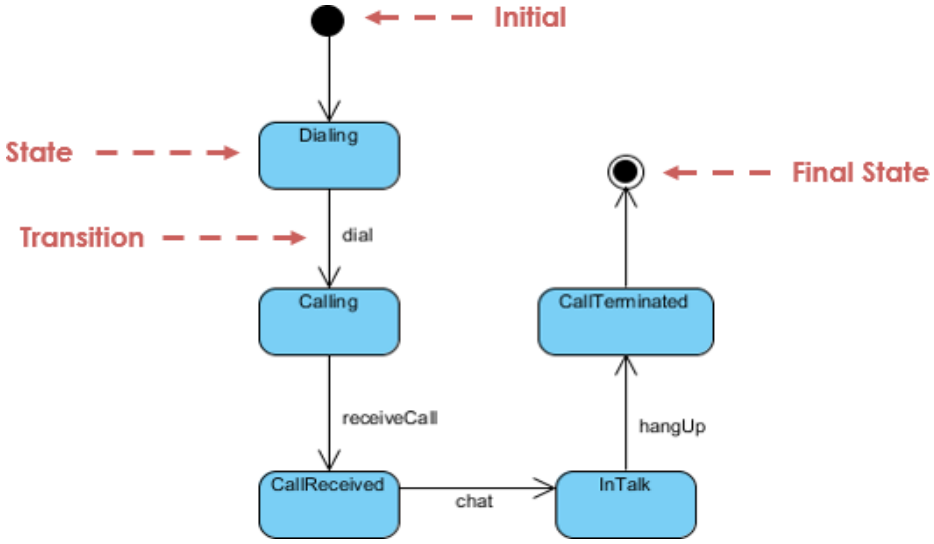

Figure 2.12 : *Exemple de diagramme d'état-transition* [2.06]

- Diagramme de séquence : un diagramme parfois appelé diagramme d'événements ou scénario d'événements, montre l'ordre dans lequel les objets interagissent. Ils nous permettent ainsi de représenter visuellement des scénarios d'exécution simples [2.08].

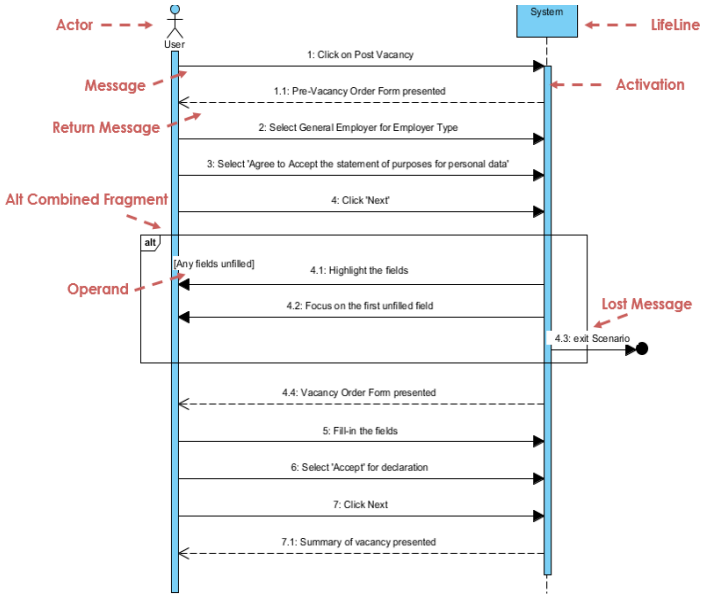

Figure 2.13 : *Exemple de diagramme de séquence* [2.06]

- Diagramme d'interaction : ce diagramme donne un aperçu du flux de contrôle entre des nœuds en interaction. Ceux-ci incluent les nœuds initiaux, les nœuds finaux de flux, les nœuds finaux d'activité, les nœuds de décision, les nœuds de fusion, les nœuds de bifurcation et les nœuds de jonction [2.08].

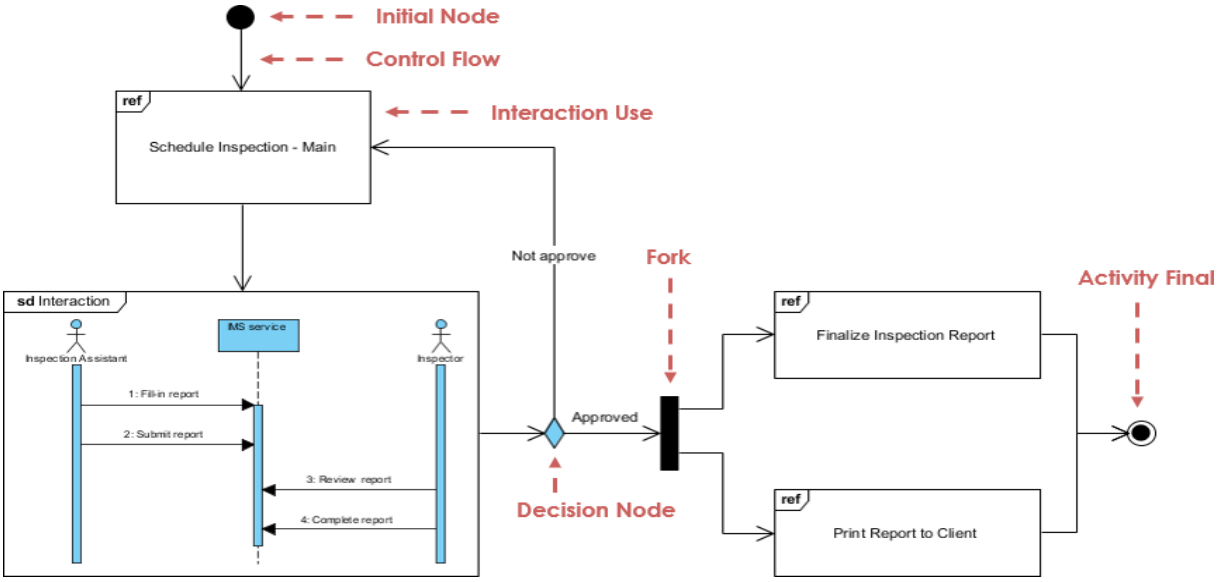

Figure 2.14 : *Exemple de diagramme d'interaction* [2.06]

- Diagramme de communication : ce diagramme, autrefois appelé diagramme de collaboration, illustrent les liens entre les objets. Ils modélisent la manière dont ces derniers s'associent et se connectent par le biais de messages dans le cadre de la conception architecturale d'un système. Ils peuvent également représenter des scénarios alternatifs dans des cas d'utilisation ou des opérations qui nécessitent la collaboration de différents objets et interactions [2.08].

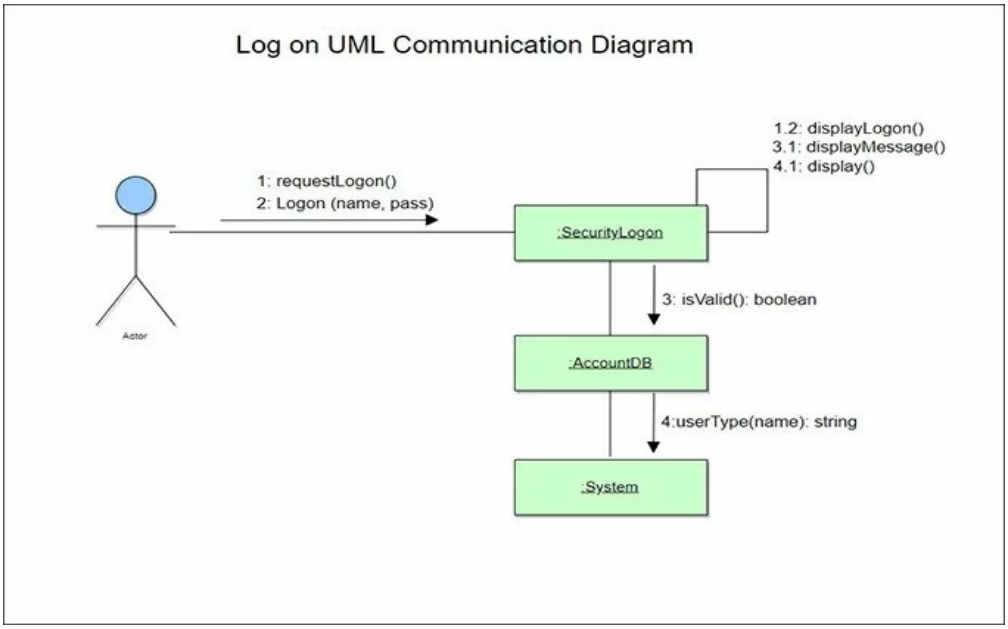

Figure 2.15 : *Exemple de diagramme de communication* [2.10]

- Diagramme de temps : souvent décrit comme un diagramme de séquence inversé, un diagramme de temps montre comment les objets interagissent entre eux dans un laps de temps donné [2.08].

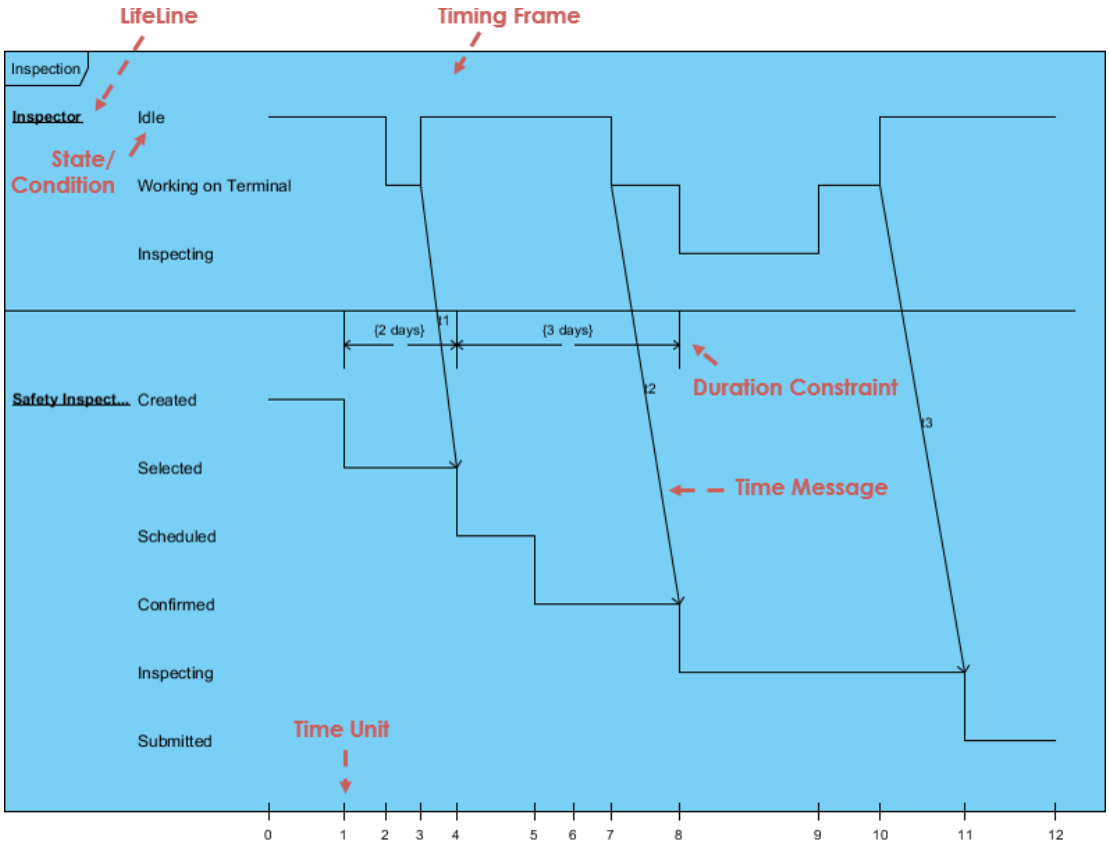

Figure 2.16 : *Exemple de diagramme de temps* [2.06]

2.3 Conception du projet

Dans le contexte de notre projet, nous emploierons l'Unified Modeling Language lors de sa conception pour sa faciliter de compréhension, son adaptabilité et ces modélisations visuels. Pour cette conception, nous emploierons les diagrammes suivants :

- Diagramme de cas d'utilisation
- Diagramme de séquence
- Diagramme d'activité
- Diagramme de classe

2.3.1 Diagramme de cas d'utilisation

Ce diagramme est utilisé pour décrire les fonctionnalités d'un système d'un point de vue utilisateur sous la forme d'actions et de réaction.

Un diagramme de cas d'utilisation se compose des éléments suivants :

- Acteur

C’est le rôle joue par une entité extérieure au système, ayant une interaction directe avec le système, peut-être de type humain ou de type non humaine comme matériel ou logiciel.

Figure 2.17 : *Acteur*

- Système

Le système permet de délimiter le sujet de l’étude, et est constitué de cas d’utilisation

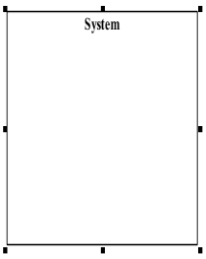

Figure 2.18 : *Système*

- Cas d’utilisation

C’est un moyen de représenter les différentes possibilités d’utiliser un système. Il exprime toujours une suite d’interaction entre un acteur et un système.

Figure 2.19 : *Cas d’utilisation*

Notre diagramme de cas d’utilisation est représenté dans la figure 2.20 :

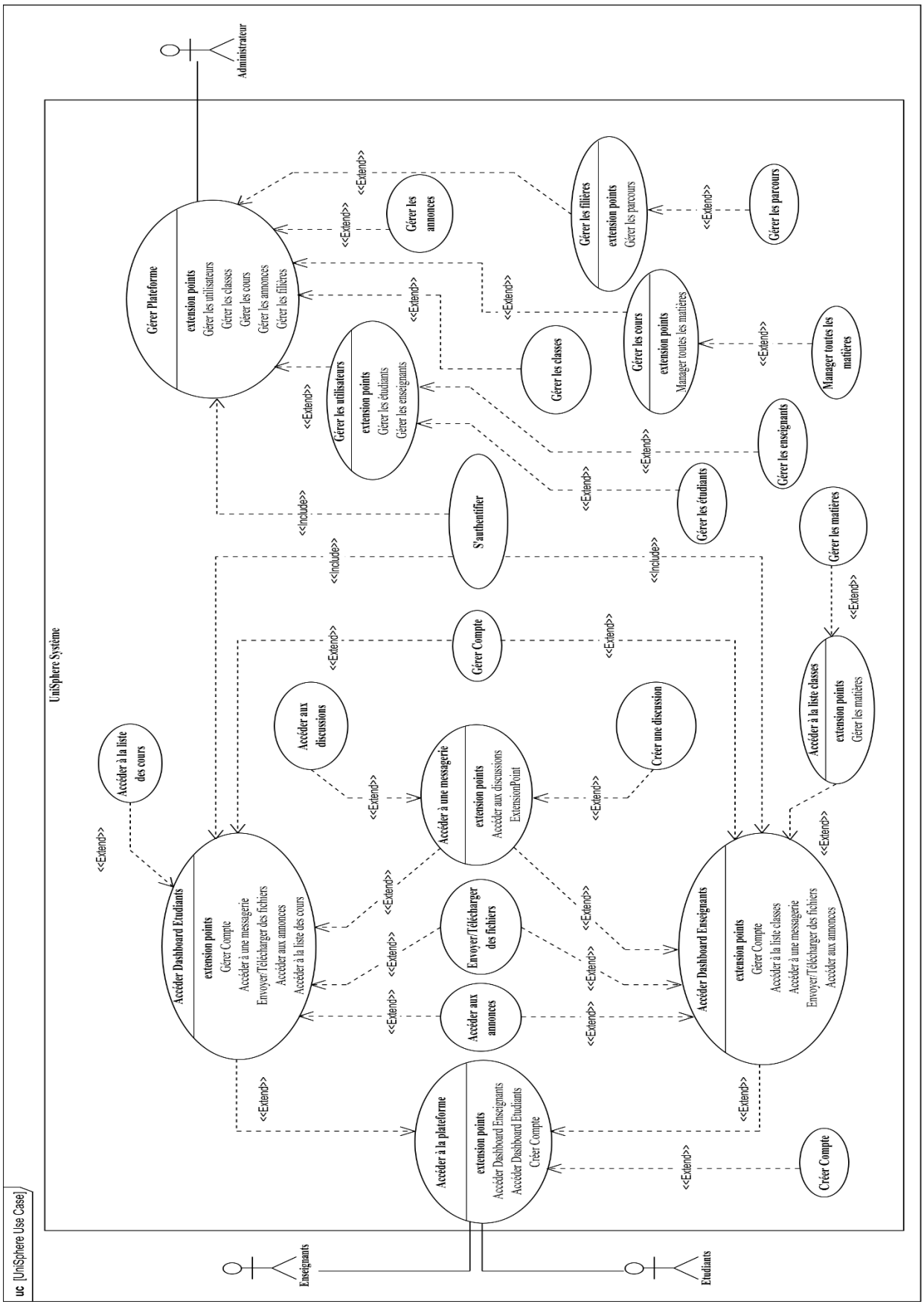

Figure 2.20 : *Diagramme de cas d'utilisation de UniSphere*

2.3.2 Description textuelle

Cette étape de la conception nous permettra de décrire avec plus de détails les cas d'utilisation présentent dans notre diagramme de cas d'utilisation. La description textuelle nous permettra de clarifier les démarches entamer afin d'exécuter une fonctionnalité.

Une description textuelle comprend les éléments suivants :

- Les préconditions : ce sont les conditions nécessaires déterminer si le cas d'utilisation peut fonctionner
- Les acteurs : nous permettent de déterminer quel acteur peut utiliser cette fonctionnalité
- Le scenario nominal : offre la description du fonctionnement du cas d'utilisation dans le cas d'un scénario le plus probable.
- Le scenario alternatif : décrit un fonctionnement du cas d'utilisation pour une partie du scénario différent du scénario le plus probable.

2.3.2.1 Description textuelle de : « S'authentifier »

a- Généralité

- Nom : S'authentifier
- Acteur : Administrateurs, Etudiants, Enseignants
- Précondition : accès à l'internet, plateforme active, possède déjà un compte

b- Scénario nominal

Tableau 2.01 : *S'authentifier (1)*

N°	Action
1	Accéder à la plateforme
2	La plateforme affiche la page d'accueil
3	L'acteur se dirige vers la page d'authentification
4	Affichage de la page d'authentification
5	L'acteur saisit les informations demandées
6	La plateforme vérifie et valide les informations entrées
7	La plateforme redirige l'acteur vers le tableau de bords qui lui correspond

c- Scénario alternatif

Tableau 2.02 : *S'authentifier (2)*

N°	Action
6	La plateforme invalide les informations saisit
6.1	La plateforme affiche un message d'erreur et attend que l'acteur saisisse à nouveau les informations demandées. Reprise au point 5

2.3.2.2 Description textuelle de : « Créer un Compte »

a- Description

- Nom : Créer un compte
- Acteur : Etudiants, Enseignants
- Précondition : accès à l'internet, plateforme active

b- Scénario nominal

Tableau 2.03 : *Créer un compte (1)*

N°	Action
1	Accéder à la plateforme
2	La plateforme affiche la page d'accueil
3	L'acteur se dirige vers la page d'inscription
4	La plateforme affiche la page d'inscription
5	L'acteur saisit les informations demandées
6	La plateforme vérifie et valide les informations entrées
7	La plateforme redirige l'acteur vers la page d'authentification

c- Scénario alternatif

Tableau 2.04 : *Créer un compte (2)*

N°	Action
6	La plateforme invalide les informations entrées
6.1	La plateforme affiche la page d'inscription avec un message d'erreur et attend que l'acteur saisisse à nouveau les informations demandées. Reprise au point 5

2.3.2.3 Description textuelle de : « Accéder à une messagerie »

a- Description

- Nom : Accéder à une messagerie
- Acteur : Etudiants, Enseignants
- Précondition : accès à l'internet, plateforme active, être connecter sur la plateforme

b- Scénario nominal

Tableau 2.05 : *Accéder à une messagerie (1)*

N°	Action
1	Inclusion du cas d'utilisation « S'authentifier »
2	La plateforme affiche le tableau de bords qui correspond à l'acteur
3	L'acteur se dirige vers la page de messagerie
4	La plateforme affiche la page de messagerie
5	L'acteur accède aux fonctionnalités de la messagerie : « Créer une discussion », « Accéder à une discussion »

2.3.2.4 Description textuelle de : « Envoyer/Télécharger des fichiers »

a- Description

- Nom : Envoyer/Télécharger des fichiers
- Acteur : Administrateur, Etudiants, Enseignants
- Précondition : accès à l'internet, plateforme active, être connecter sur la plateforme

b- Scénario nominal

Tableau 2.06 : *Envoyer/Télécharger des fichiers (1)*

N°	Action
1	Inclusion du cas d'utilisation « S'authentifier »
2	Inclusion du cas d'utilisation « Accéder à une messagerie »
3	Inclusion du cas d'utilisation « Accéder à une discussion »
4	La plateforme affiche une discussion privée ou groupée
5	L'acteur appuie sur le bouton envoyer fichier (image ou document)
6	La plateforme affiche un formulaire
7	L'acteur choisi un fichier et envoie sur la plateforme
8	La plateforme vérifie et valide le fichier
9	La plateforme affiche le fichier dans la discussion privée ou groupée

c- Scénario alternatif

Tableau 2.07 : *Envoyer/Télécharger des fichiers (2)*

N°	Action
8	La plateforme invalide le fichier
8.1	La plateforme affiche un message d'erreur et attend que l'acteur fasse une action. Reprise au point 6

2.3.2.5 Description textuelle de : « Créer des discussions »

a- Description

- Nom : Créer des discussions
- Acteur : Etudiants, Enseignants
- Précondition : accès à l'internet, plateforme active, être connecter sur la plateforme

b- Scénario nominal

Tableau 2.08 : *Créer des discussions (1)*

N°	Action
1	Inclusion du cas d'utilisation « S'authentifier »
2	Inclusion du cas d'utilisation « Accéder à une messagerie »
3	La plateforme affiche la page de messagerie
4	L'acteur appuie sur le bouton de création de discussion
5	La plateforme affiche une liste des utilisateurs (étudiants et enseignants)
6	L'acteur sélectionne un ou plusieurs utilisateurs pour participer à la discussion
7	La plateforme crée une discussion dans avec les membres sélectionnés par l'acteur

c- Scénario alternatif

Tableau 2.09 : *Créer des discussions (2)*

N°	Action
6	L'acteur ne sélectionne aucun utilisateur pour la discussion
6.1	La plateforme ne crée pas de discussion. Reprise au point 3

2.3.2.6 Description textuelle de : « Gérer les utilisateurs »

a- Description

- Nom : Gérer les utilisateurs
- Acteur : Administrateur
- Précondition : accès à l'internet, plateforme active, être connecter sur la plateforme

b- Scénario nominal

Tableau 2.10 : *Gérer les utilisateurs (1)*

N°	Action
1	Inclusion du cas d'utilisation « S'authentifier »
2	La plateforme affiche le tableau de bords de l'administrateur
3	L'acteur se dirige vers la page de gestion d'utilisateur
4	La plateforme affiche la page de gestion d'utilisateur : - Gestion d'étudiants - Gestion d'enseignants
5	L'acteur appuie sur le bouton de gestion d'étudiants
6	La plateforme affiche la page de gestion d'étudiants

c- Scénario alternatif

Tableau 2.11 : *Gérer les utilisateurs (2)*

N°	Action
5	L'acteur appuie sur le bouton de gestion d'enseignants
5.1	La plateforme affiche la page de gestion d'enseignants

2.3.2.7 Description textuelle de : « Gérer les classes »

a- Description

- Nom : Gérer les classes
- Acteur : Administrateur
- Précondition : accès à l'internet, plateforme active, être connecter sur la plateforme

b- Scénario nominal

Tableau 2.12 : *Gérer les classes (1)*

N°	Action
1	Inclusion du cas d'utilisation « S'authentifier »
2	La plateforme affiche le tableau de bords de l'administrateur
3	L'acteur se dirige vers la page de gestion de classes
4	La plateforme affiche une liste des classes
5	L'acteur sélectionne une classe
6	La plateforme affiche les informations de la classe

2.3.3 Diagramme de séquence

Ce diagramme représente les interactions entre objets selon un point de vue temporel, et on y met l'accent sur la chronologie des envois de messages, montre les réactions du système aux actions des utilisateurs, la création et la manipulation des objets. Il document un ou plusieurs scenarios d'un cas d'utilisation. C'est utile en phase d'analyse et de test.

Un diagramme de séquence se compose :

- Les acteurs : permet de préciser les utilisateurs qui effectue les actions durant le scénario
- Les objets : qui sont représenté par un rectangle contenant le nom de l'objet
- Ligne de vie : c'est une ligne verticale pointillé dirigée vers le bas à partir de chaque objet, elle symbolise une durée qui dépend du scénario et du comportement modélisée. Elle représente aussi l'ensemble des opérations exécutées par un objet.

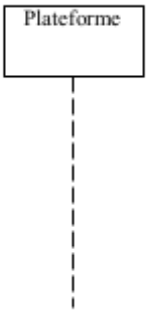

Figure 2.21 : *Ligne de vie*

- Barre d’activation : c’est un rectangle qui remplace la ligne de vie
- Message : c’est généralement un appel, un signal ou une réponse qui est représenter par des flèches horizontales relient la ligne de vie de l’objet émetteur à la ligne de vie de l’objet récepteur

2.3.3.1 Diagramme de séquence de : « S’authentifier »

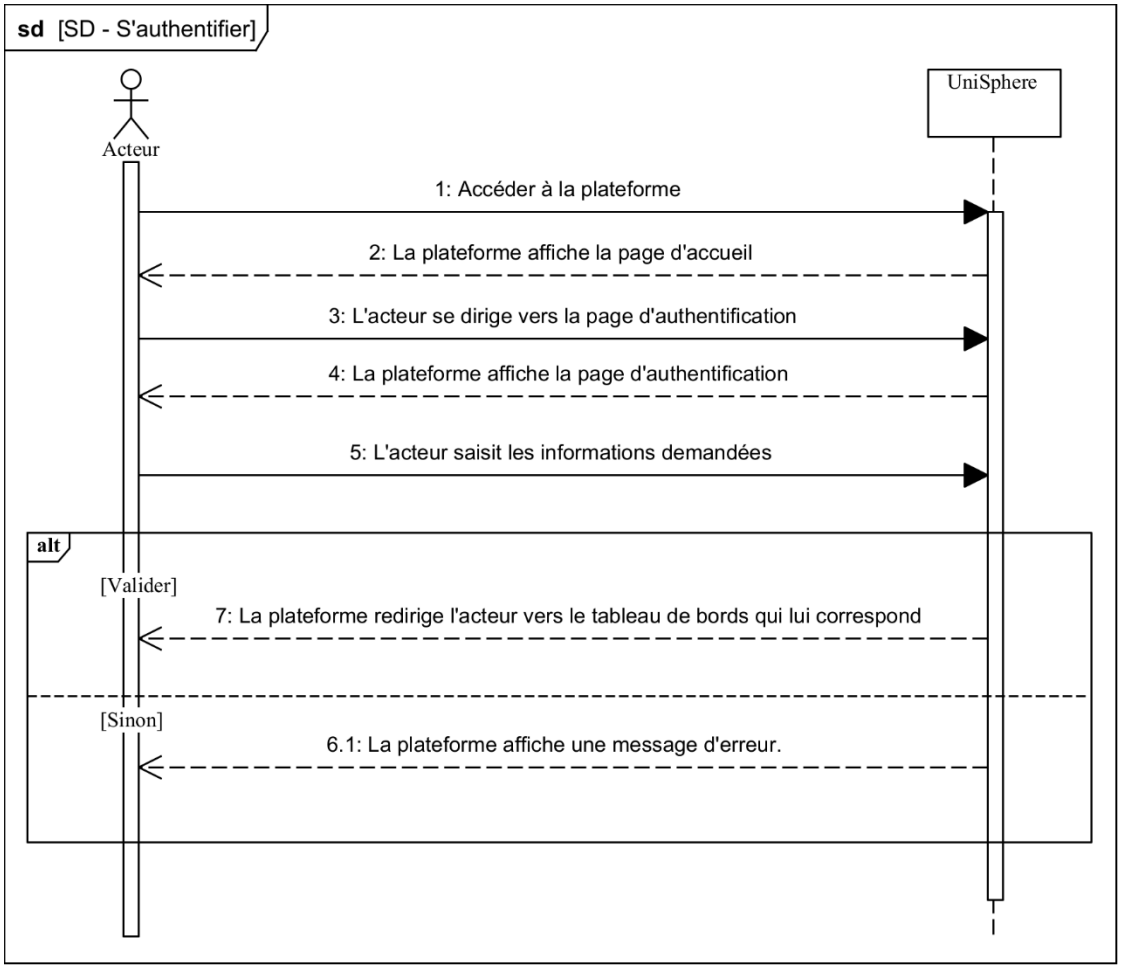

Figure 2.22 : *Diagramme de séquence de « S’authentifier »*

2.3.3.2 Diagramme de séquence de : « Créer un compte »

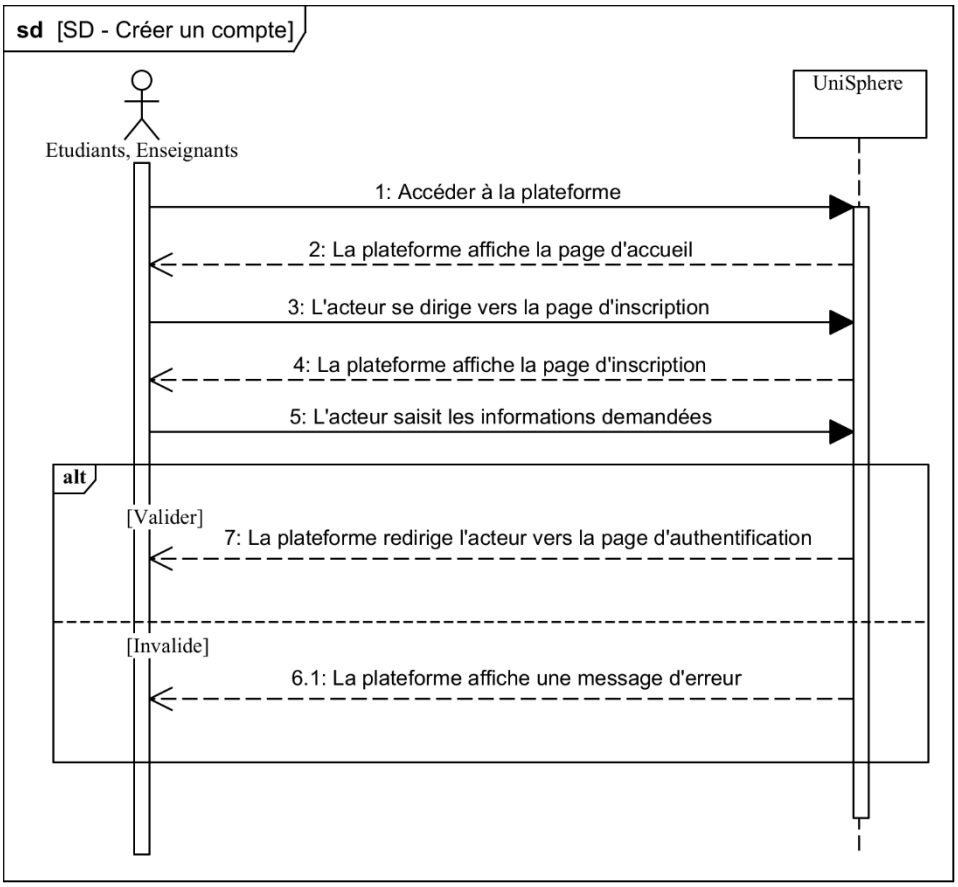

Figure 2.23 : *Diagramme de séquence de «* Créer un compte *»*

2.3.3.3 Diagramme de séquence de : « Accéder à une messagerie »

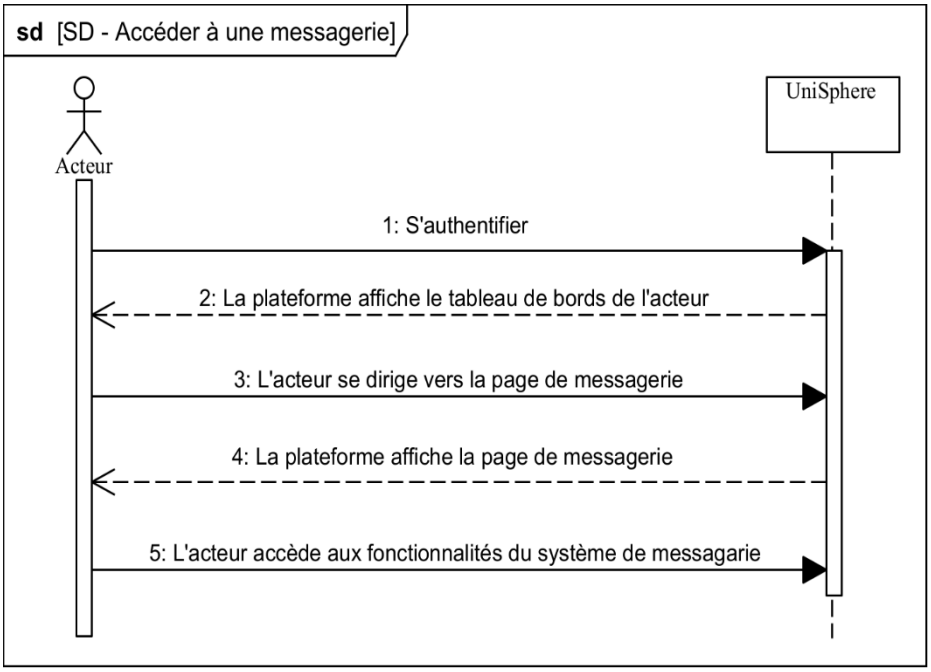

Figure 2.24 : *Diagramme de séquence de «* Accéder à une messagerie *»*

2.3.3.4 Diagramme de séquence de : « Envoyer/Télécharger des fichiers »

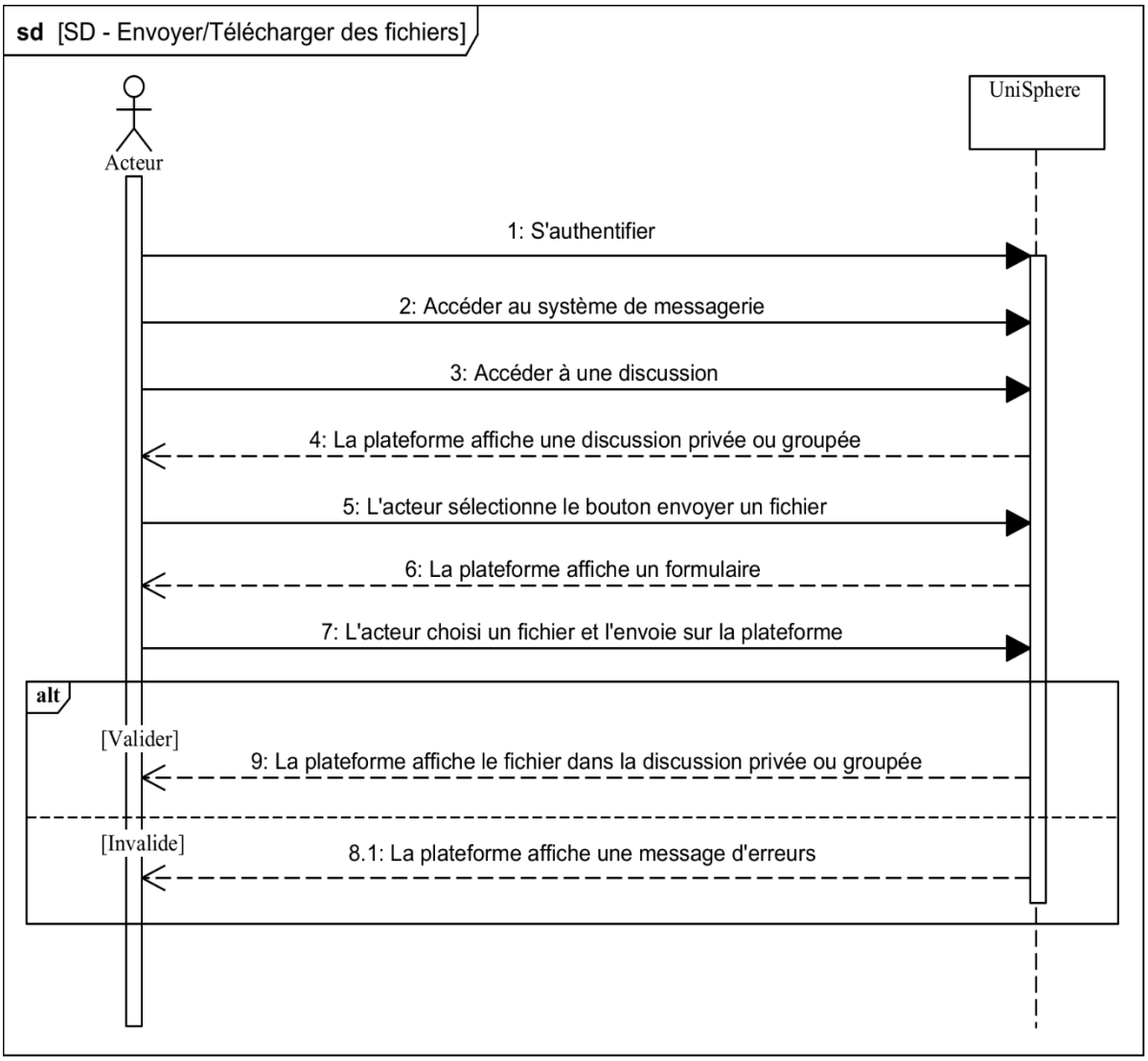

Figure 2.25 : *Diagramme de séquence de «* Envoyer/Télécharger des fichiers *»*

2.3.3.5 Diagramme de séquence de : « Créer une discussion »

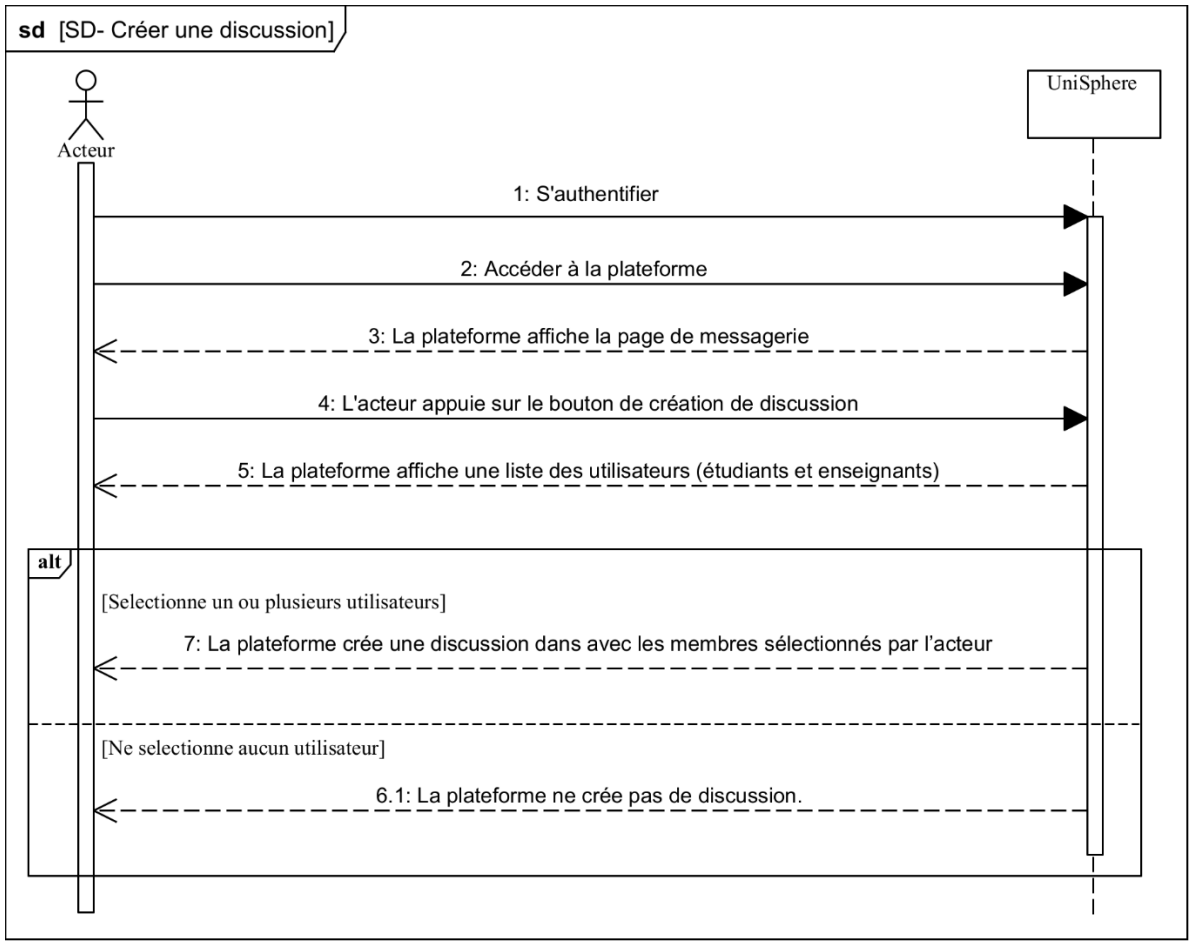

Figure 2.26 : *Diagramme de séquence de «* Créer une discussion *»*

2.3.3.6 Diagramme de séquence de : « Gérer les utilisateurs »

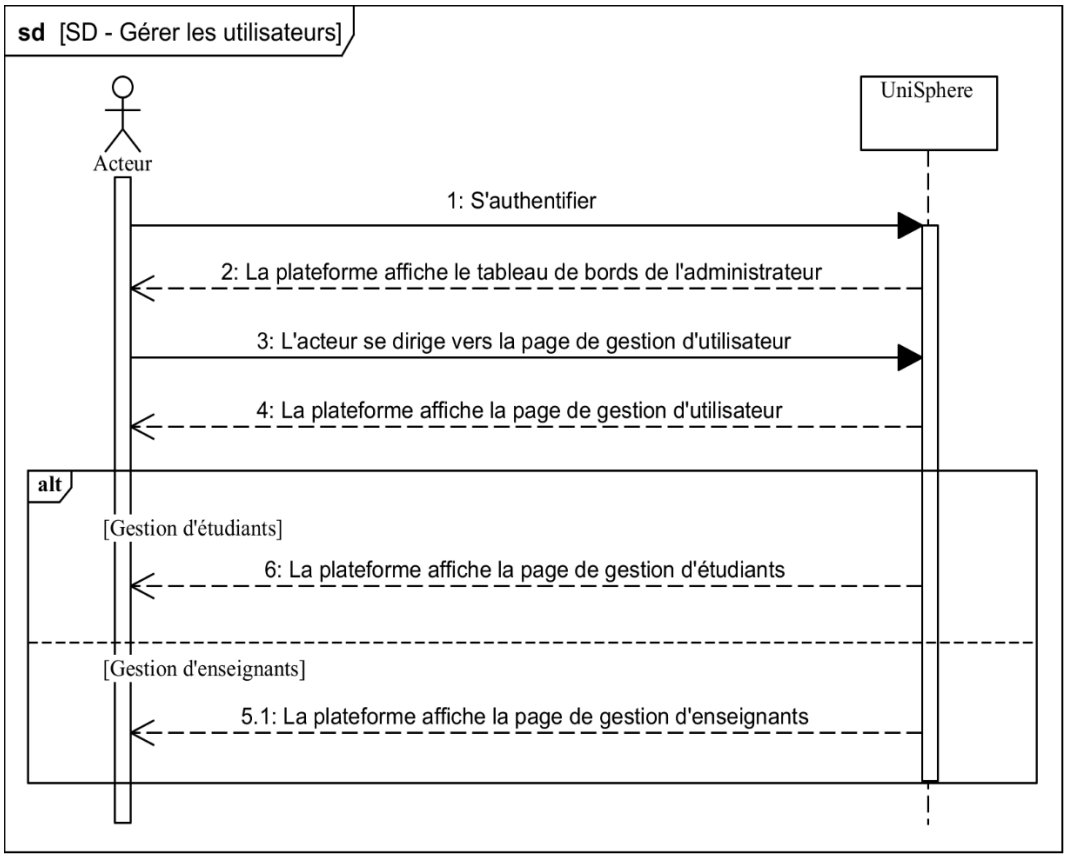

Figure 2.27 : *Diagramme de séquence de «* Gérer les utilisateurs *»*

2.3.3.7 Diagramme de séquence de : « Gérer les classes »

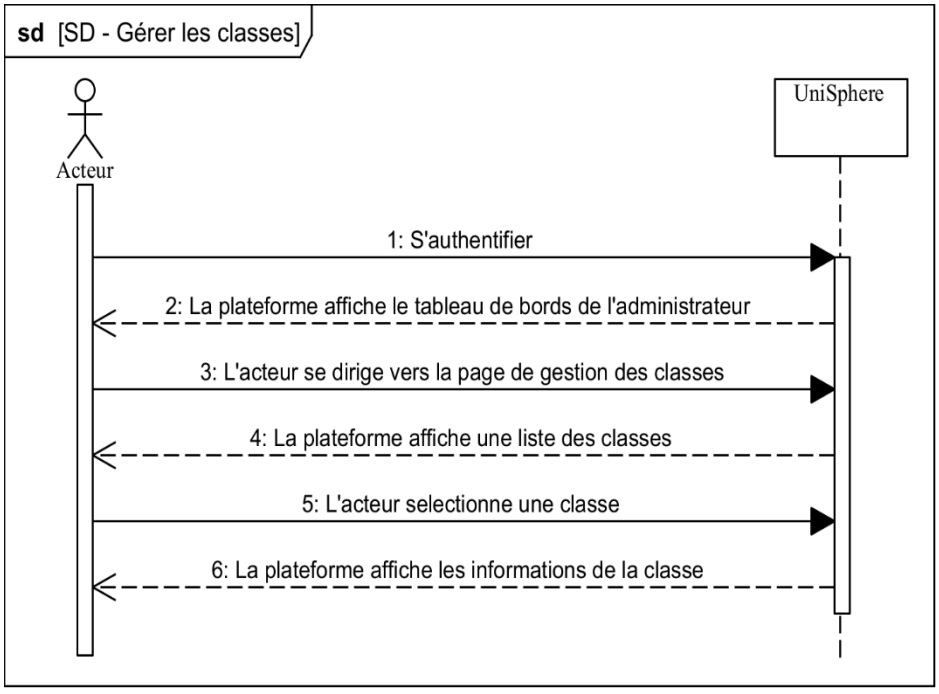

Figure 2.28 : *Diagramme de séquence de «* Gérer les classes *»*

2.3.4 Diagramme d'activité

C'est une variante du diagramme d'état transition, permettant de représenter le comportement interne d'un cas d'utilisation ou processus et prend en charge les comportements parallèles.

Un diagramme d'activité comprend les éléments suivants :

- Action : c'est une étape dans l'activité où les acteurs ou logiciel exécutent une tâche donnée ;
- Etat initial : c'est un élément qui marque le début de l'activité ;

Figure 2.29 : *Etat initial*

- Etat final : c'est un élément qui marque la fin de toute activité ;

Figure 2.30 : *Etat final*

- Activités : représentent le comportement d'un objet ;

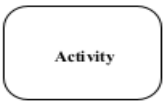

Figure 2.31 : *Activité*

- Nœuds de test décision : permet de faire un choix entre plusieurs flots sortant en fonction des conditions de garde de chaque flot. Il ne possède qu'une seule entrée et peut utiliser deux (2) flots de sortie, le premier correspondant à la condition vérifié et l'autre traitant le cas contraire ;

Figure 2.32 : *Nœuds de test décision*

- Nœuds de bifurcation ou fourche : permet la création de plusieurs flots concurrents en sortie de la barre de synchronisation à partir d'un flot unique entrant ;

Figure 2.33: *Nœuds de bifurcation*

Pour notre plateforme, les diagrammes d'activités sont les suivants :

2.3.4.1 Diagramme d'activité de : « S'authentifier »

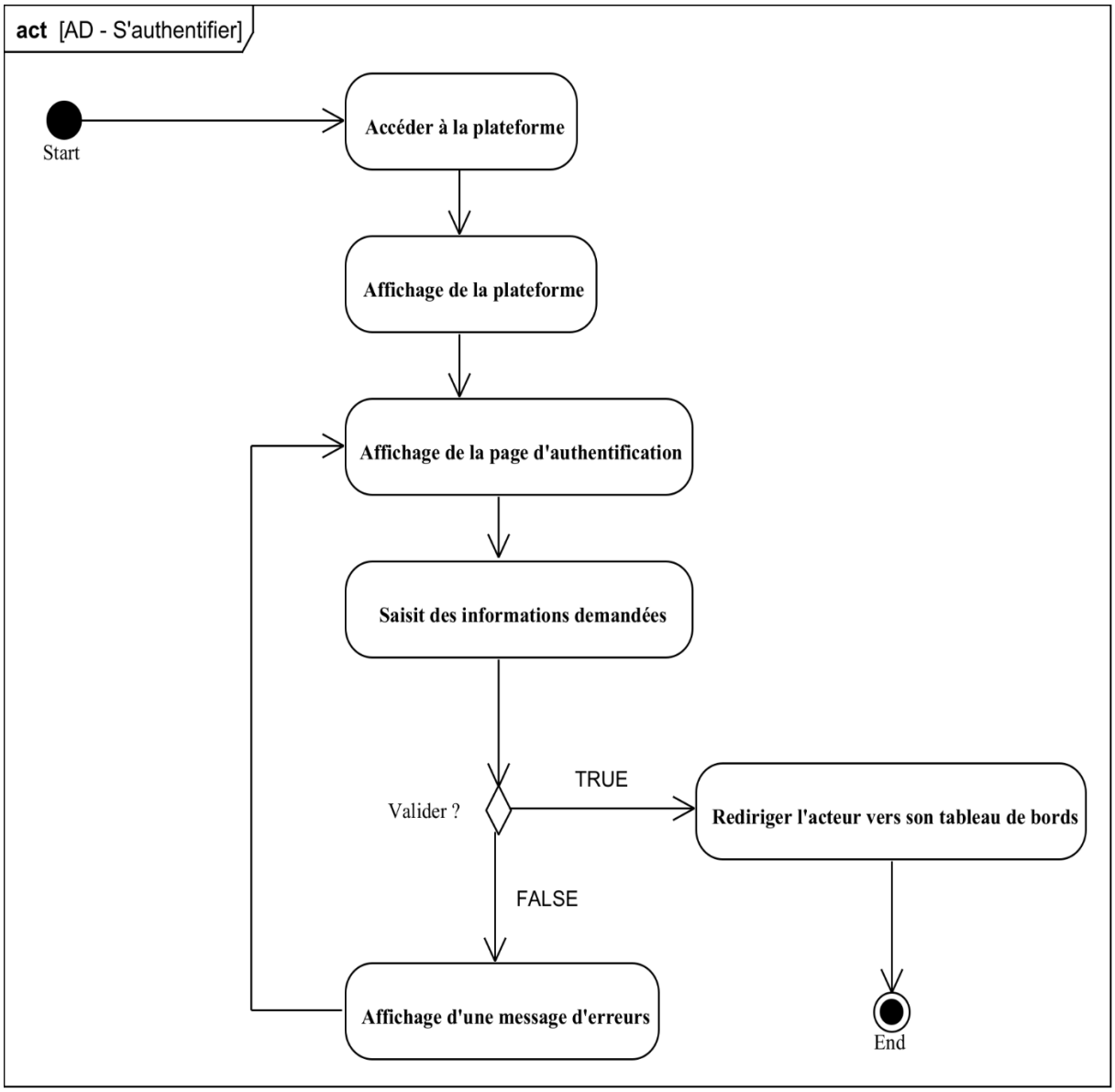

Figure 2.34 : *Diagramme d'activité de «* S'authentifier *»*

2.3.4.2 Diagramme d’activité de : « Créer un compte »

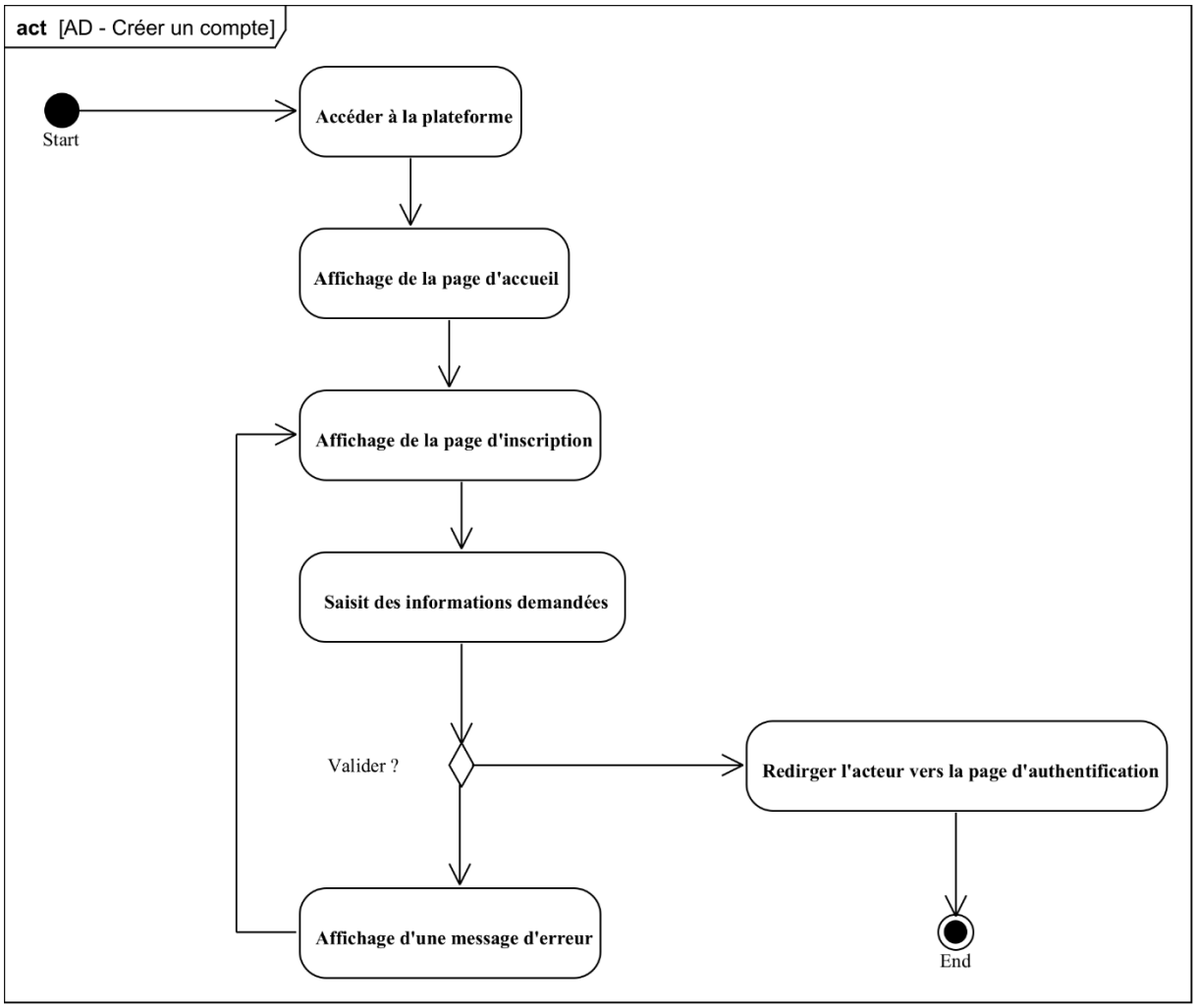

Figure 2.35 : *Diagramme d’activité de «* Créer un compte *»*

2.3.4.3 Diagramme d’activité de : « Accéder à une messagerie »

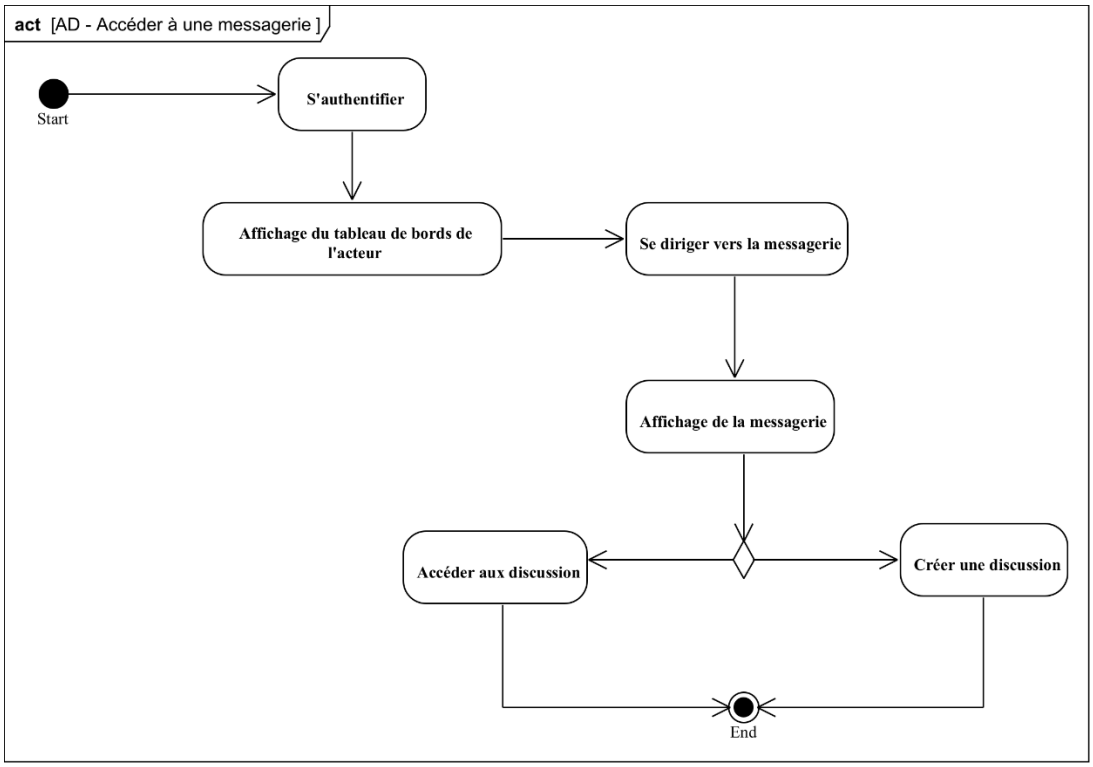

Figure 2.36 : *Diagramme d'activité de « Accéder à une messagerie »*

2.3.4.4 Diagramme d'activité de : « Envoyer/Télécharger des fichiers »

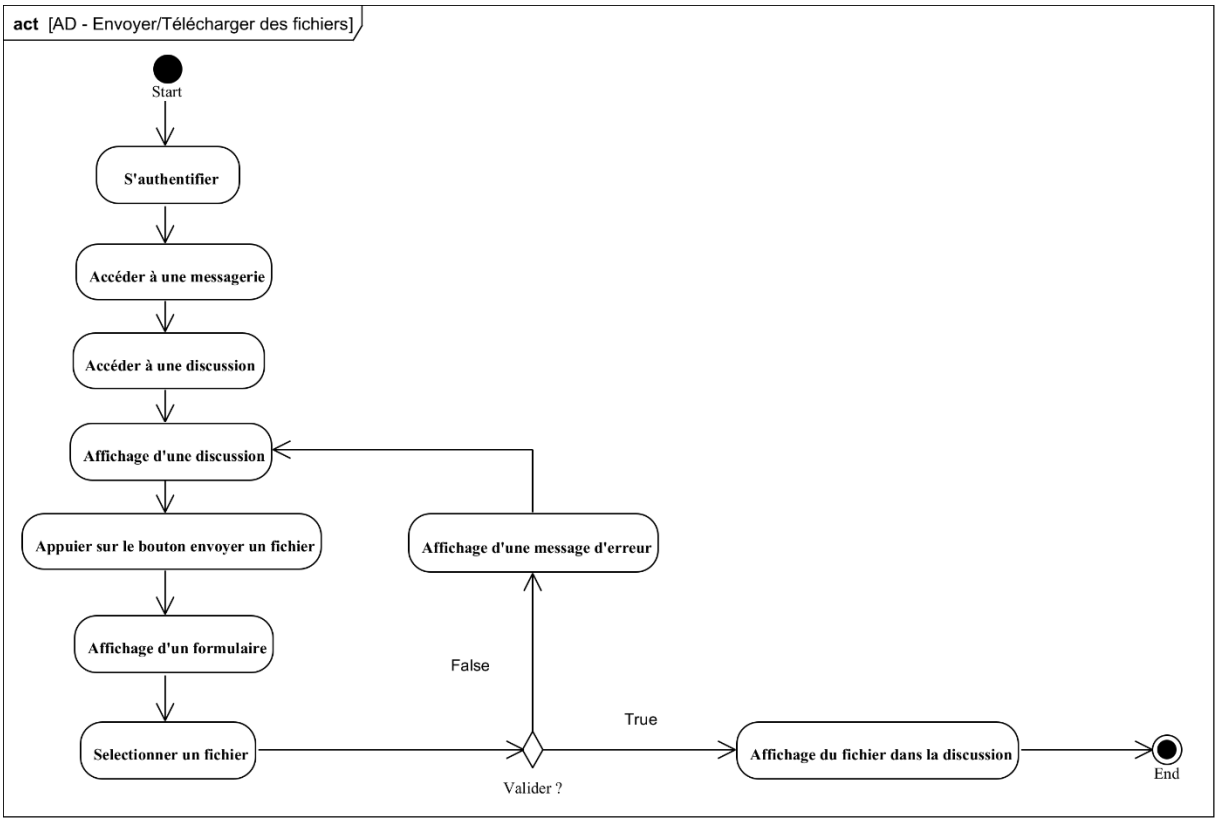

Figure 2.37 : *Diagramme d'activité de « Envoyer/Télécharger des fichiers »*

2.3.4.5 Diagramme d'activité de : « Créer une discussion »

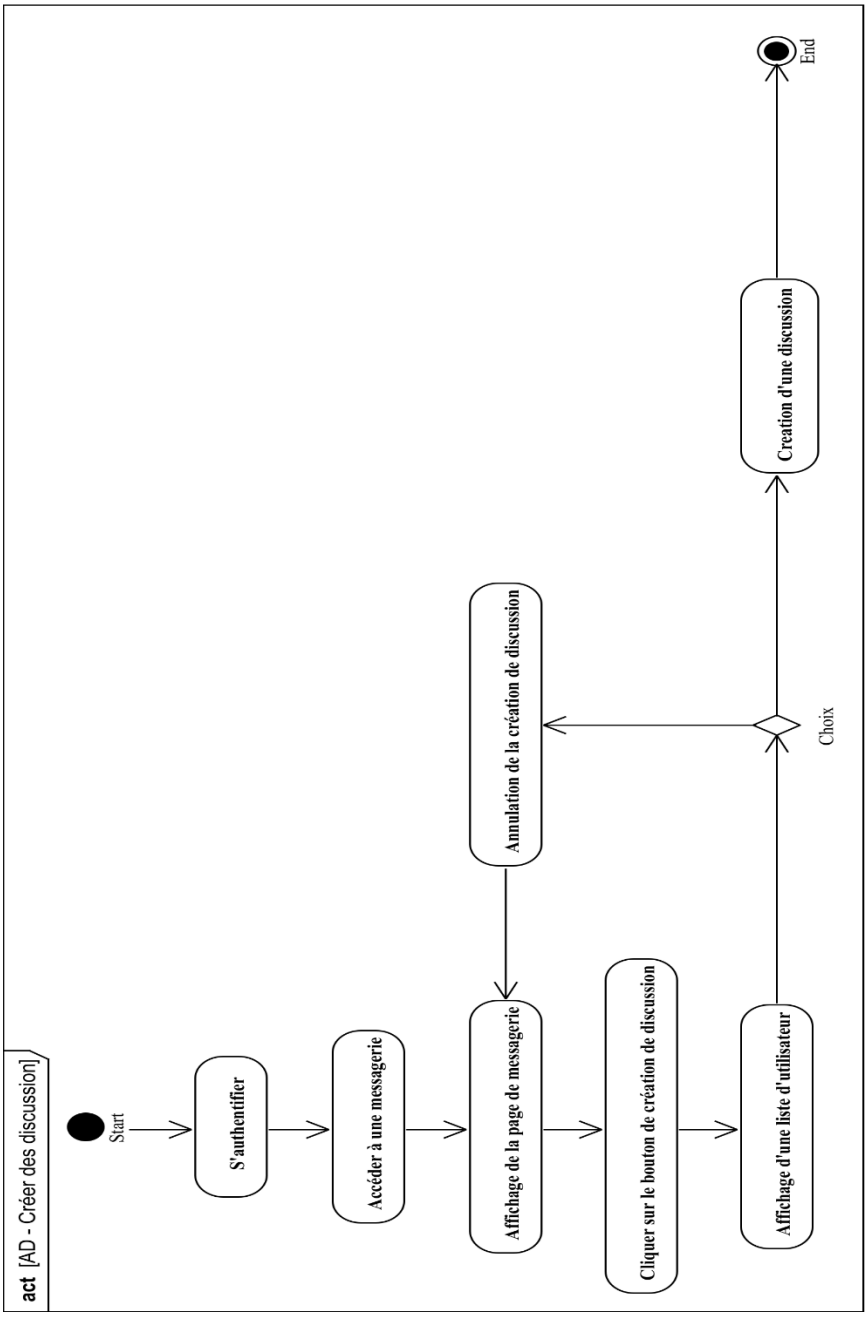

Figure 2.38 : *Diagramme d'activité de « Créer une discussion »*

2.3.4.6 Diagramme d'activité de : « Gérer les utilisateurs »

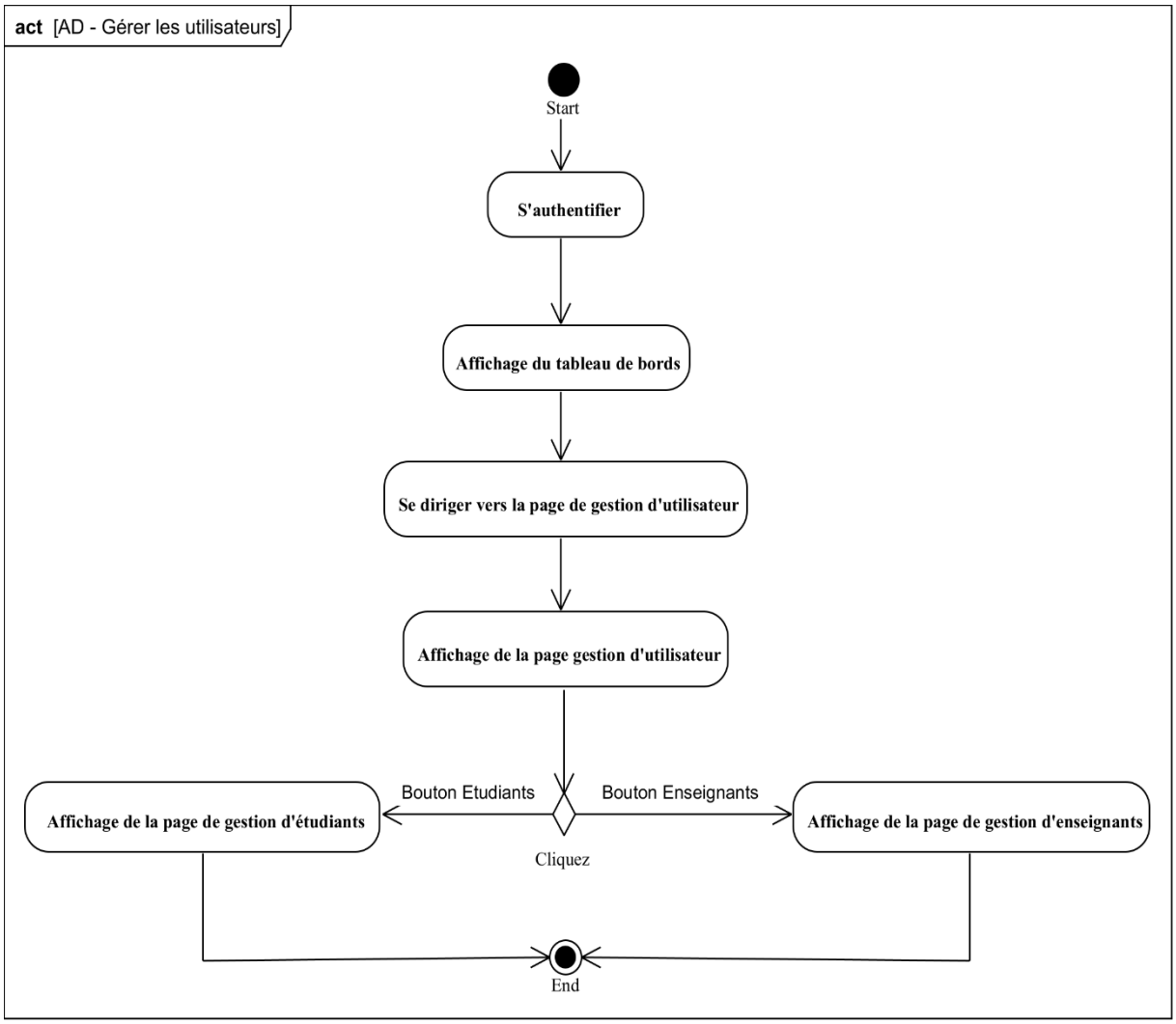

Figure 2.39 : *Diagramme d'activité de « Gérer les utilisateurs »*

2.3.4.7 Diagramme d'activité de : « Gérer les classes »

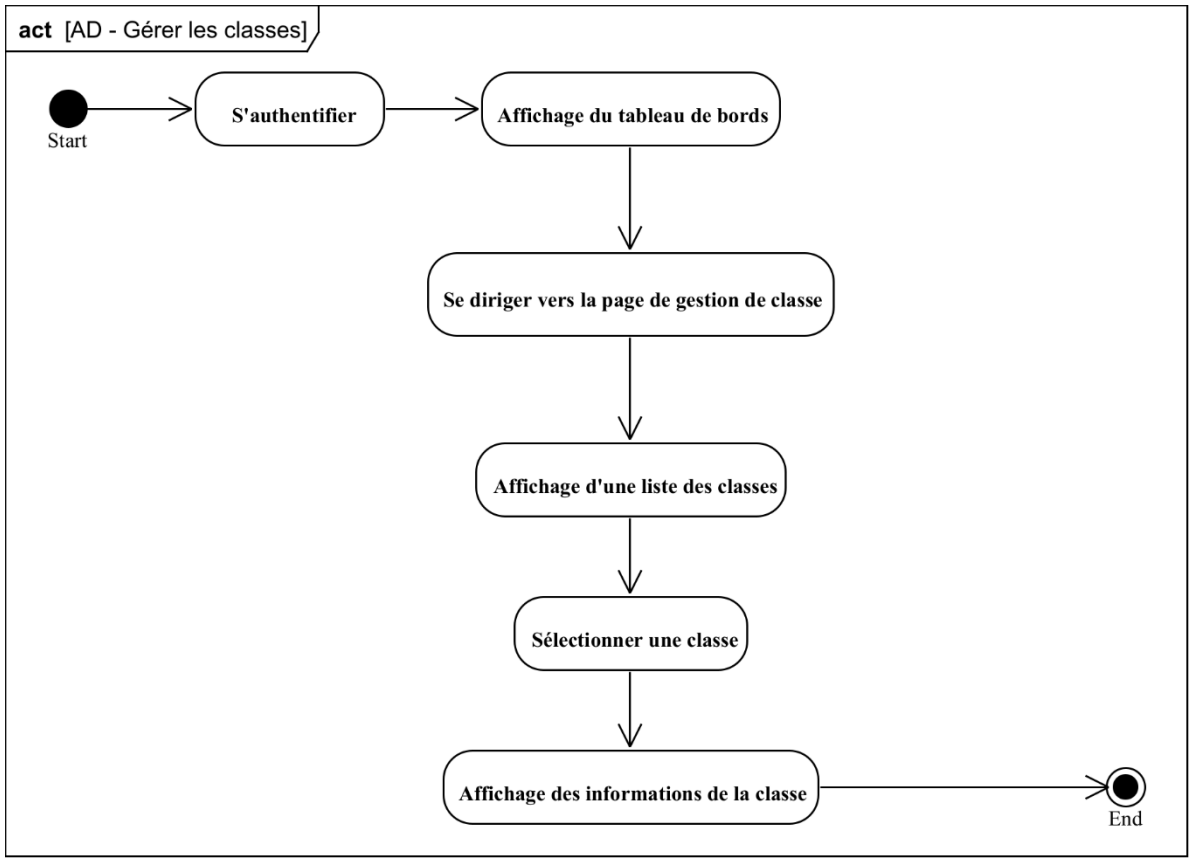

Figure 2.40 : *Diagramme d'activité de « Gérer les classes »*

2.3.5 Diagramme de classe

Le diagramme de classe est considéré comme le pivot de l'ensemble de la modélisation d'un système. Il permet de donner la représentation statique du système à développer, il est fondé sur le concept d'objet, le concept de classe comprenant les attributs, les opérations et les différents types d'association entre classes.

Ce diagramme est composé des éléments suivants :

- Classe : qui permet de décrire un groupe d'objets et comportant les attributs et les opérations ;
- Attribut : propriété élémentaire d'une classe ;
- Opération : c'est une fonction applicable aux objets d'une classes. Elles décrivent les comportements d'un objet ;
- Association : représente les liens existants entre les instances des classes ;
- Multiplicité : indique un domaine de valeur pour préciser le nombre d'instance d'une classe vis-à-vis d'une autre classe pour une association donnée ;

Le diagramme de classe de notre projet est représenté dans la figure suivante :

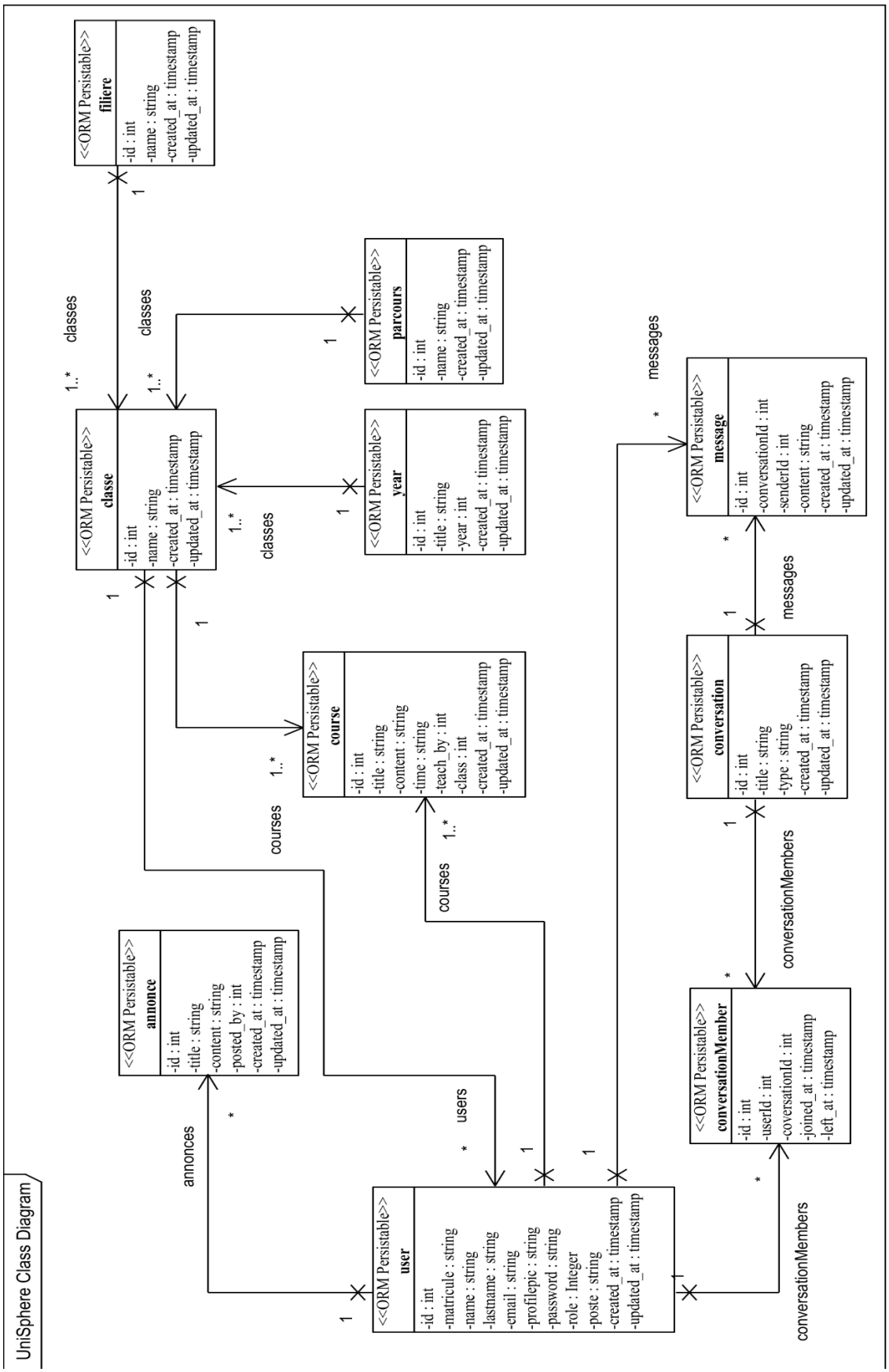

Figure 2.41 : *Diagramme de classe de UniSphere*

2.4 Conclusion

Comme nous avons vu dans ce chapitre, la phase de conception nous a permis de détailler les fonctionnalités importantes de notre plateforme. Ainsi dans le prochain chapitre nous allons voir la phase de réalisation de notre projet.