UML
(Unified Modeling Language)
Introduction

Mohamed Samb
Ingénieur de Conception

mohamedsamb176@gmail.com

Prérequis

Ce cours s’adresse aux étudiants du secteur informatique. En effet, UML est un langage visuel permettant d’illustrer un projet logiciel et d’échanger sur le projet entre développeur et client.

Partie I : les bases en UML

- Définition
- Les différents types de diagrammes
- Démarche
- Le principe d’itération en UML

Partie I : les bases en UML - Concept

Comme n’importe quel projet, un projet informatique nécessite un phase d’analyse, suivi d’une étape de conception.

- Dans la phase d’analyse, on cherche d’abord à bien comprendre et à décrire de manière claire les besoins des utilisateurs. Communément, on l’appelle aussi phase « l’analyse des besoins ».
- Dans la phase de conception, on apporte plus de détails à la solution et on cherchera à clarifier des aspects techniques, tels que l’installation des différentes parties logicielles à installer sur du matériel.

Partie I : les bases en UML

I. Concept

I.1 Définition

UML, c'est l'acronyme anglais pour « **Unified Modeling Language** ». On le traduit par « Langage de modélisation unifié ».

La notation UML est un langage visuel constitué d'un ensemble de schémas, appelés des diagrammes, qui donnent chacun une vision différente du projet à traiter.

UML nous fournit donc des diagrammes pour représenter le logiciel à développer :

- son fonctionnement ;
- sa mise en route ;
- les actions susceptibles d'être effectuées par le logiciel ;
- Etc.

Réaliser ces diagrammes revient donc à modéliser les besoins du logiciel.

Partie I : les bases en UML - Concept

I.2 Pourquoi modéliser ?

Tout client ou utilisateur aimerait voir à quoi va ressembler le futur système qu'il utilisera. C'est dans ce sens que s'inscrit la modélisation.

Elle permet de montrer l'ensemble des éléments qui seront montés de toute pièces afin d'obtenir le système final.

Modéliser, c'est décrire de manière visuelle et graphique les besoins et, les solutions fonctionnelles et techniques de votre projet logiciel.

Partie I : les bases en UML - Concept

I.3 Les deux approches dans la gestion de projet

Dans la gestion de projet, nous pouvons citer deux approches permettant de définir les besoins :

- **La décomposition fonctionnelle** (ou l'approche procédurale)
- **L'approche objet** (sur laquelle est basée UML)

Partie I : les bases en UML - Concept

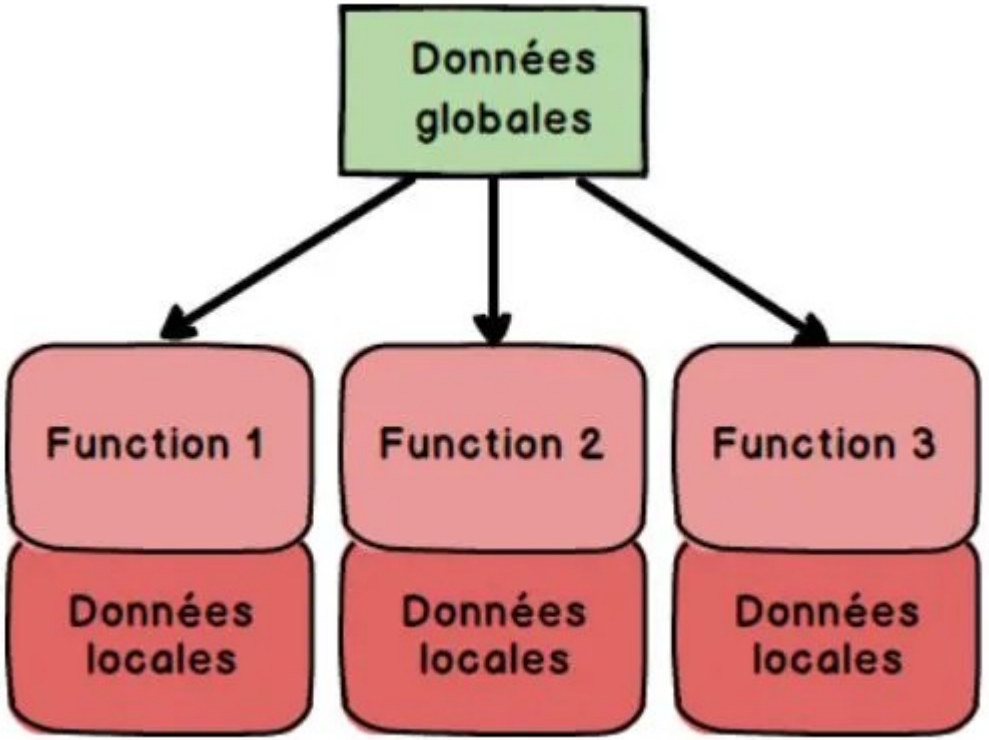
I.3.1 La décomposition fonctionnelle

Avant l’apparition de l’approche objet dans les années 80, une cette démarche était largement utilisée. Pendant longtemps, de nombreux logiciels étaient conçus par l’approche fonctionnelle descendante, appelée également la décomposition fonctionnelle.

L’approche par décomposition fonctionnelle considère que le logiciel est composé d'une hiérarchie de fonctions et de données.

Les fonctions fournissent les services désirés et les données représentent les informations manipulées.

Partie I : les bases en UML - Concept



Partie I : les bases en UML - Concept

Avantages

- Démarche est logique
- Cohérente
- Intuitive

Inconvénients

- Les fonctions sont alors interdépendantes
- Maintenance Difficile
- Evolution Difficile

I.3.2 L’approche Objet

L’approche objet est une démarche qui s’organise autour de 4 principes fondamentaux. C’est une démarche :

- itérative et incrémentale ;
- guidée par les besoins du client et des utilisateurs ;
- centrée sur l’architecture du logiciel ;
- qui décrit les actions et les informations dans une seule entité.

Partie I : les bases en UML - Concept

a) Itérative et incrémentale

La modélisation d’un logiciel se fera en plusieurs fois c’est-à-dire les diagrammes ne seront pas réalisés de façon complètement satisfaisante dès la première fois. Il nous faudra plusieurs versions d’un même diagramme pour obtenir la version finale. Chaque version est le fruit d’une itération (un nouveau passage sur les diagrammes). Une itération permet donc d’affiner la compréhension du futur logiciel.

b) Guidée par les besoins du client et des utilisateurs

Les besoins d’un client, dans un projet logiciel sont les exigences exprimées par le client au niveau des fonctionnalités, du rendu et des actions d’un logiciel. Les besoins des utilisateurs servent de fil rouge, tout au long du cycle de développement :

- lors de la phase d'analyse pour la clarification, l’affinage et la validation des besoins des utilisateurs ;
- lors de la phase de conception et de réalisation pour la vérification de la prise en compte des besoins des utilisateurs ;
- lors de la phase de test afin de garantir la satisfaction des besoins.

c) La démarche est également centrée sur l'architecture du logiciel

L’architecture logicielle décrit les choix stratégiques qui déterminent en grande partie les qualités du logiciel (adaptabilité, performances, fiabilité...). On peut citer l’architecture client serveur, en couche ou en niveaux.

d) L’approche objet décrit aussi bien les actions que les informations dans une même entité

Les différents diagrammes utilisés en UML donnent tous une vision particulière du logiciel à développer. Parmi ces diagrammes, il en est un qui représente particulièrement bien ce qui est à développer si on opte pour un développement objet. Il s’agit du ***diagramme de classes***. Le diagramme de classes donnera une vision assez claire des informations qui seront utilisées par le logiciel, mais également des fonctions (ou opérations) qui devront s’appuyer sur ces informations. L’approche objet mélange donc habillement l’analyse des informations et des actions au lieu de les analyser séparément.

Partie I : les bases en UML
II. Les types de diagrammes

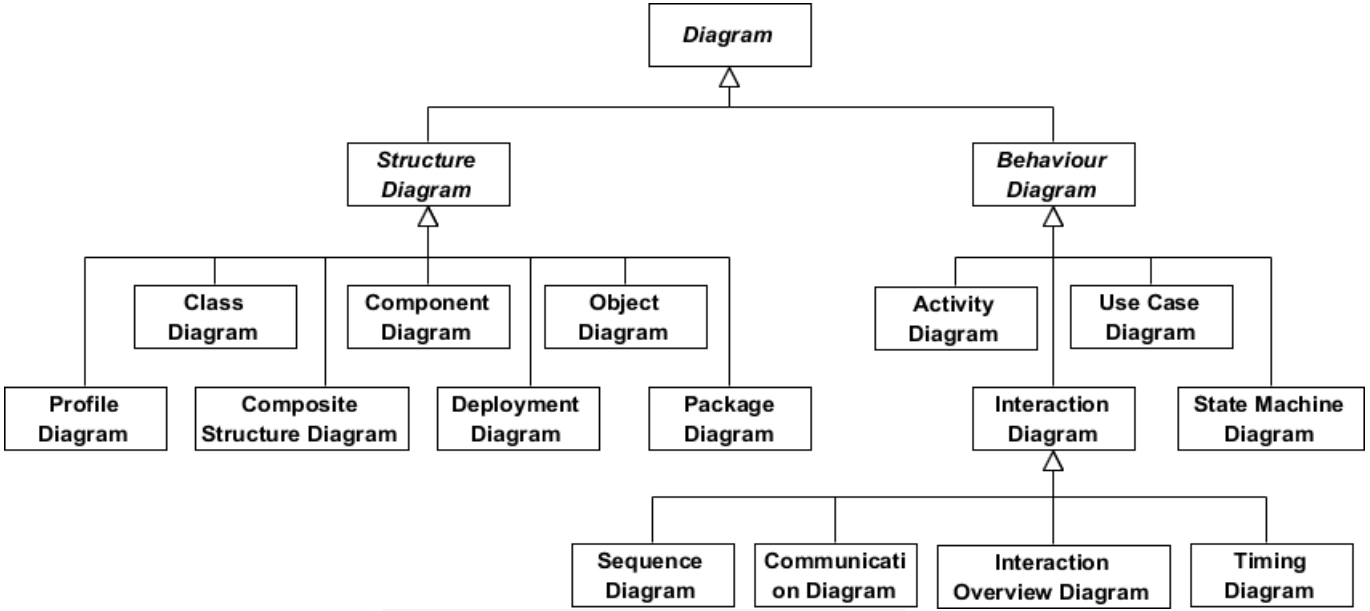
La réalisation d'une application informatique ou d'un ensemble d'applications est basée sur plusieurs diagrammes.
Le langage UML est constitué de diagrammes. À ce jour, il existe 13 diagrammes « officiels ». Ces diagrammes sont tous réalisés à partir du besoin des utilisateurs et peuvent être regroupés selon les deux aspects suivants :

- Les aspects fonctionnels : qui utilisera le logiciel et pour quoi faire ? Comment les actions devront-elles se dérouler ? Quelles informations seront utilisées pour cela ?
- Les aspects liés à l'architecture (statique et dynamique): quels seront les différents composants logiciels à utiliser (base de données, librairies, interfaces, etc.) ? Sur quel matériel chacun des composants sera installé ?

UML modélise donc le système logiciel suivant ces deux modes de représentation.

Partie I : les bases en UML
II. Les types de diagrammes

II.1 Vue d'ensemble



Partie I : les bases en UML
II. Les types de diagrammes

La première chose à remarquer à propos d'UML est qu'il existe de nombreux diagrammes (modèles) différents auxquels il faut s'habituer. La raison en est qu'il est possible d'examiner un système sous de nombreux points de vue différents. Un développement de logiciel impliquera de nombreuses parties prenantes.

Par exemple :

- Analystes
- Designers
- Codeurs
- Testeurs
- Assurance qualité
- Le client
- ...

Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de cas d’utilisation

Un cas d'utilisation est une liste d'actions ou d'étapes événementielles définissant généralement les interactions entre un rôle d'acteur et un système pour atteindre un objectif. Un cas d'utilisation est une technique utile pour identifier, clarifier et organiser les exigences du système. Un cas d'usage est constitué d'un ensemble de séquences possibles d'interactions entre systèmes et utilisateurs qui définit les fonctionnalités à mettre en œuvre et la résolution des éventuelles erreurs pouvant être rencontrées.

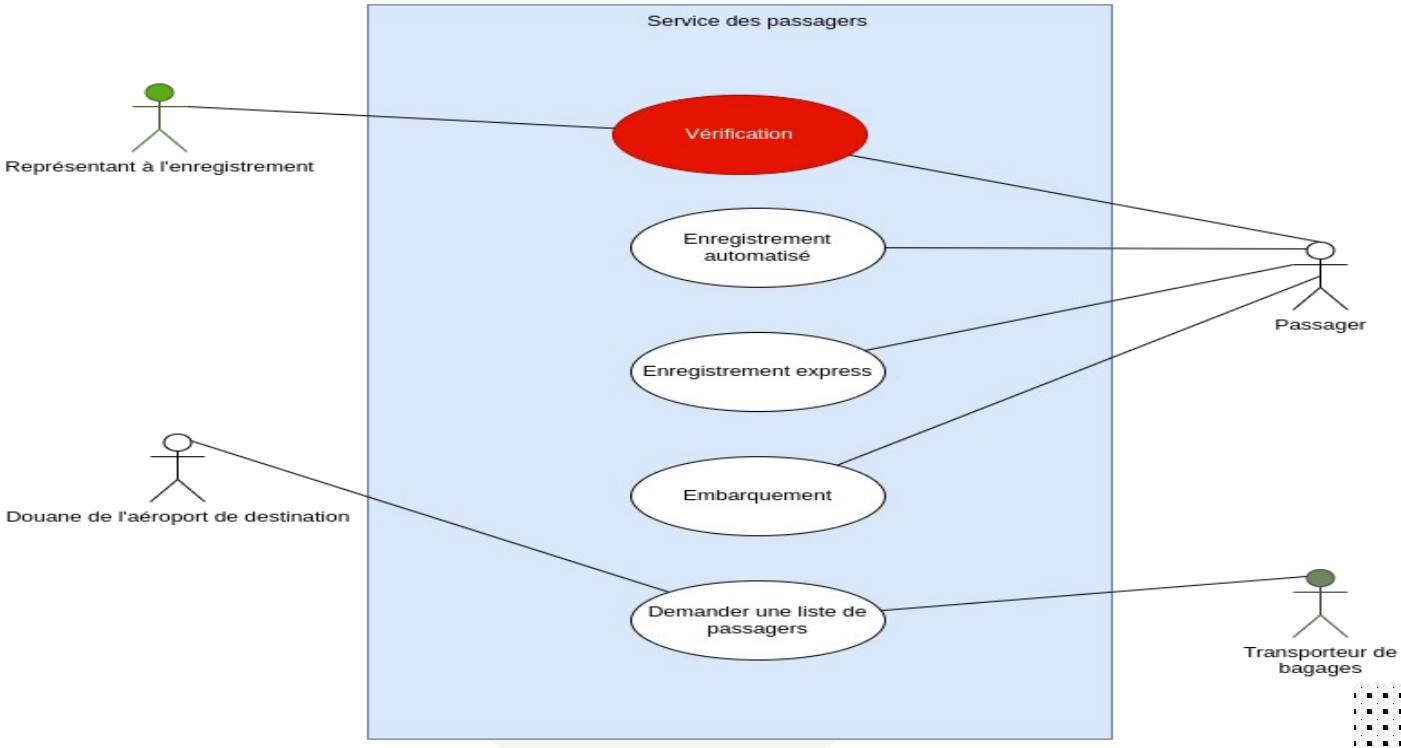
Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de cas d’utilisation (2)

Un cas d'utilisation (ou un ensemble de cas d'utilisation) présente les caractéristiques suivantes :

- Organise les exigences fonctionnelles
- Modélise les objectifs des interactions système/acteur (utilisateur)
- Décrit un flux principal d'événements (scénarios principaux) et éventuellement d'autres flux exceptionnels (alternatifs), également appelés chemins ou scénarios utilisateurs.

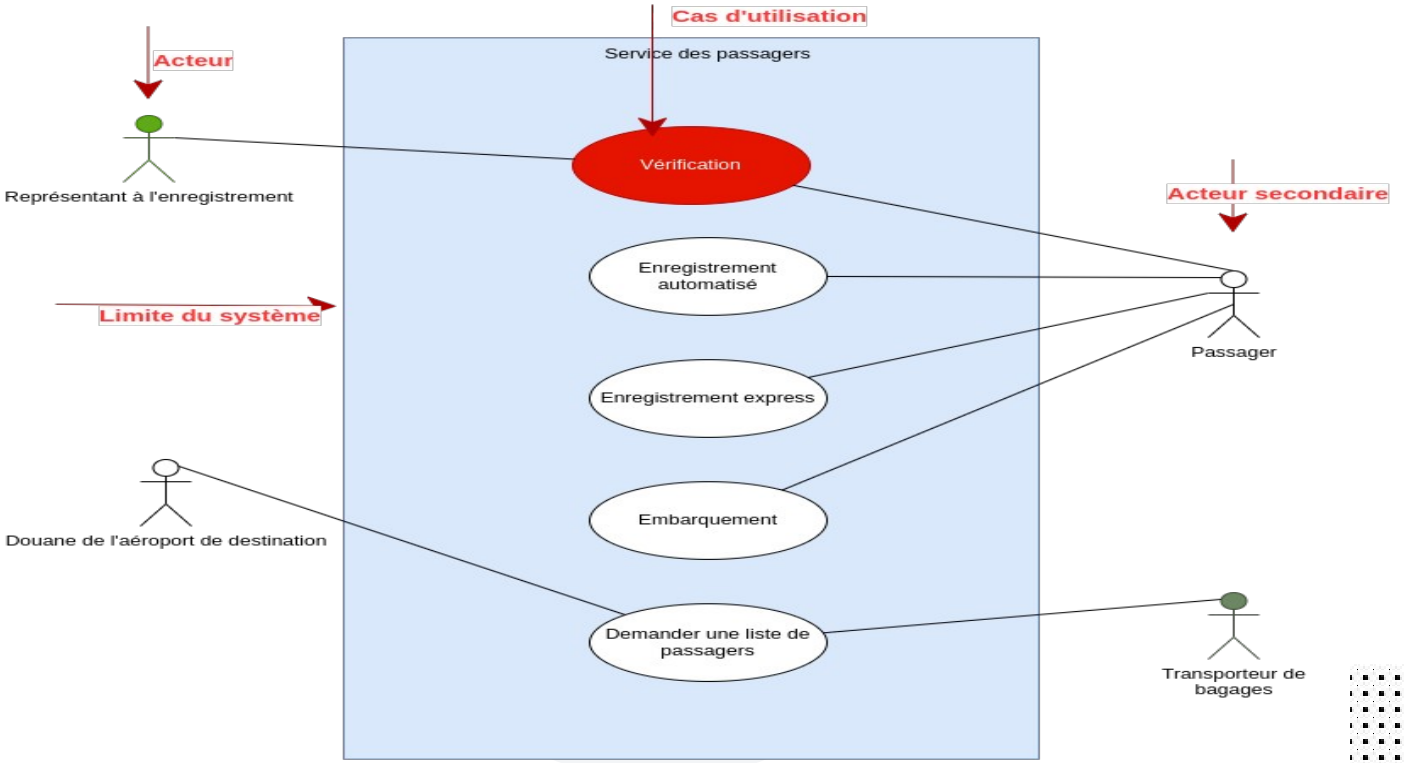
Partie I : les bases en UML
II. Les types de diagrammes
II.2 Diagramme de cas d'utilisation (3)



Partie I : les bases en UML

II. Les types de diagrammes

II.2 Diagramme de cas d'utilisation (4)



Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de cas d’utilisation (5)

- **Acteur**
  Les acteurs sont généralement des individus impliqués dans le système défini en fonction de leurs rôles. L'acteur peut être un humain ou un autre système externe.

- **Cas d'utilisation**
  Un cas d'utilisation décrit comment les acteurs utilisent un système pour atteindre un objectif particulier. Les cas d'utilisation sont généralement initiés par un utilisateur pour atteindre des objectifs décrivant les activités et les variantes impliquées dans la réalisation de l'objectif.

Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de cas d’utilisation (6)

- **Relation**
  Les relations sont entre et parmi les acteurs et les cas d’usage.

- **Limite du système**
  La frontière du système définit le système d’intérêt par rapport au monde qui l’entoure.

Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de cas d'utilisation (7)
II.2.1 Avantages

- Les cas d'utilisation constituent une technique puissante pour l'élicitation et la documentation des exigences fonctionnelles de la boîte noire.

- Parce que les cas d'utilisation sont faciles à comprendre et constituent un excellent moyen de communiquer avec les clients et les utilisateurs car ils sont rédigés en langage naturel.

- Les cas d'utilisation peuvent aider à gérer la complexité des grands projets en divisant le problème en principales fonctionnalités utilisateur (c'est-à-dire les cas d'utilisation) et en spécifiant les applications du point de vue des utilisateurs.

Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de cas d'utilisation (8)

II.2.1 Avantages

- Un scénario de cas d'utilisation, souvent représenté par un diagramme de séquence, implique la collaboration de plusieurs objets et classes, les cas d'utilisation aident à identifier les messages (opérations et informations ou données requises - paramètres) qui collent les objets et les classes ensemble.

- Les cas d'utilisation fournissent une bonne base pour faire le lien entre la vérification des modèles de niveau supérieur (c'est-à-dire l'interaction entre les acteurs et un ensemble d'objets collaboratifs), et par la suite, pour la validation des exigences fonctionnelles (c'est-à-dire le plan de test en boîte blanche).

Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de cas d’utilisation (9)

II.2.1 Avantages

- L'approche basée sur les cas d'utilisation fournit des liens traçables pour le suivi du projet dans lequel les activités de développement clés telles que les cas d'utilisation sont mises en œuvre, testées et livrées, répondant aux buts et objectifs du point de vue de l'utilisateur.

Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de cas d’utilisation (10)

II.2.1 comment dessiner un diagramme de cas d'utilisation ?

Un modèle de cas d'utilisation peut être développé en suivant les étapes ci-dessous.

1. Identifiez les acteurs (rôle des utilisateurs) du système.
2. Pour chaque catégorie d'utilisateurs, identifiez tous les rôles joués par les utilisateurs pertinents pour le système.
3. Identifiez quels sont les utilisateurs qui ont besoin que le système soit exécuté pour atteindre ces objectifs.
4. Créez des cas d'utilisation pour chaque objectif.
5. Structurez les cas d’usage.
6. Hiérarchiser, examiner, estimer et valider les utilisateurs.

Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de cas d’utilisation (11)

II.2.1 comment dessiner un diagramme de cas d'utilisation ?

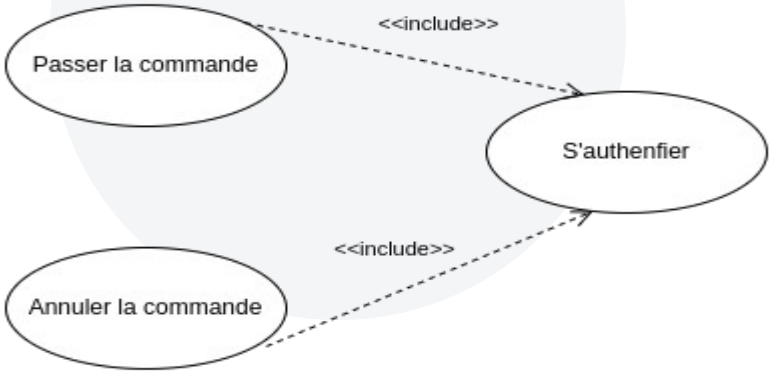
NB : pour rendre l'approche cas d'usage plus « Agile », ne détaillez pas tous les cas d'usage, mais priorisez-les dans votre backlog produit, vous devez affiner le cas d'usage à différents niveaux de détails en fonction de la phase de développement en juste-à-temps. et juste ce qu'il faut.

- **Backlog** : l’ensemble contenant toutes les fonctionnalités du système final en cours de développement (Scrum).

Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de cas d'utilisation (12)
II.2.1 Relations entres UCs

- Cas d'utilisation **<<include>>**
  Le moment d'utiliser la relation <<include>> est après avoir terminé la première description de tous vos principaux cas d'utilisation. Vous pouvez désormais examiner les cas d'utilisation et identifier les séquences courantes d'interaction utilisateur-système.



Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de cas d'utilisation (13)
II.2.1 Relations entres UCs

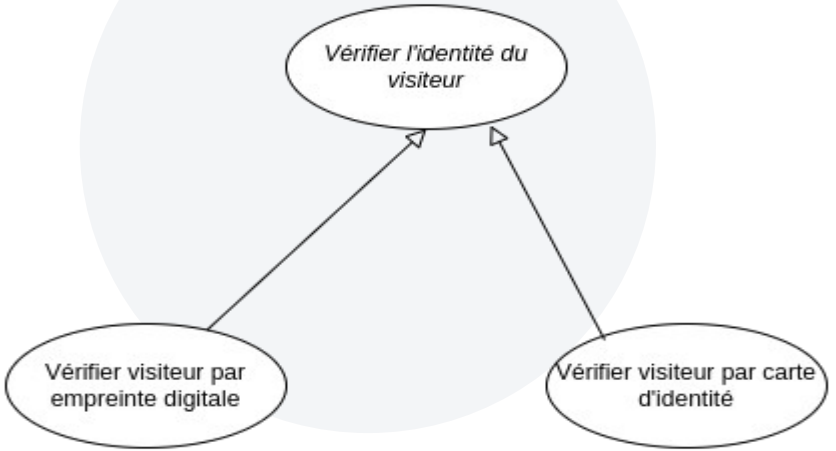
- Cas d'utilisation **<<extend>>**
  Un cas d'utilisation étendu est, en fait, une évolution alternative du cas d'utilisation de base. Le cas d'utilisation <<extend>> y parvient en insérant conceptuellement des séquences d'actions supplémentaires dans la séquence de cas d'utilisation de base.

Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de cas d'utilisation (14)

II.2.1 Relations entres UCs

- Cas d'utilisation **abstrait** et **généralisé**
  Le cas d'utilisation général est abstrait. Il ne peut pas être instancié car il contient des informations incomplètes. Le titre d'un cas d'utilisation abstrait est affiché en italique.



Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de cas d'utilisation (15)

II.2.1 Description d'un UC

La description textuelle d'un cas d'utilisation permet de décrire la chronologie des actions qui devront être réalisées par les acteurs et par le système lui-même. On parle d'ailleurs de scénarios.

La description d'un cas d'utilisation permet de :

- clarifier le déroulement de la fonctionnalité ;
- décrire la chronologie des actions qui devront être réalisées ;
- d'identifier les parties redondantes pour en déduire des cas d'utilisation plus précises qui seront utilisées par inclusion, extension ou généralisation/spécialisation. Et oui, dans ce cas nous réaliserons des itérations sur les diagrammes de cas d'utilisation ;
- d'indiquer d'éventuelles contraintes déjà connues et dont les développeurs vont devoir tenir compte lors de la réalisation du logiciel. Ces contraintes peuvent être de nature diverse.

Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de cas d'utilisation (16)

II.2.1 Description d'un UC

La description est faite l'aide d'une fiche descriptive pour chaque cas d'utilisation, de façon à raconter l'histoire du déroulement des différentes actions.
Cette fiche descriptive doit comporter 4 volets :

- L'identification
- La description des scénarios
- La fin et les post-conditions
- Les compléments

Partie I : les bases en UML
II. Les types de diagrammes
II.2 Diagramme de cas d'utilisation (17)
II.2.1 Description d'un UC

Identification		Nom du cas d'utilisation
Date		
Acteur(s)		
Préconditions		Activités qui doivent avoir lieu, ou toute condition qui doit être vraie, avant que le cas d'utilisation puisse être démarré
Scénario nominal	**Description**	Actions de l'utilisateur et réponses du système qui auront lieu lors de l'exécution du cas d'utilisation dans des conditions normales et attendues.
	Postconditions	État du système à la fin de l'exécution du cas d'utilisation avec un flux normal : nominal.
Scénarios alternatifs / erreurs		Principaux flux alternatifs ou exceptions pouvant survenir dans le flux de l'événement.
Compléments		Toutes les exigences non fonctionnelles : par exemple, fiabilité, sécurité, etc., performances, ergonomie

Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de package (1)

Le diagramme de package montre la disposition et l'organisation des éléments du modèle dans un projet de moyenne à grande échelle qui peut être utilisé pour montrer à la fois la structure et les dépendances entre les sous-systèmes ou les modules.

Les grands systèmes présentent des défis particuliers. Dessinez un modèle de classes d’un système d’une grande envergure rend ce dernier trop gros pour être compris. Il y a trop de liens entre les classes ou UCs pour comprendre. Une technique utile pour gérer cela est celle des packages UML.

Partie I : les bases en UML
II. Les types de diagrammes

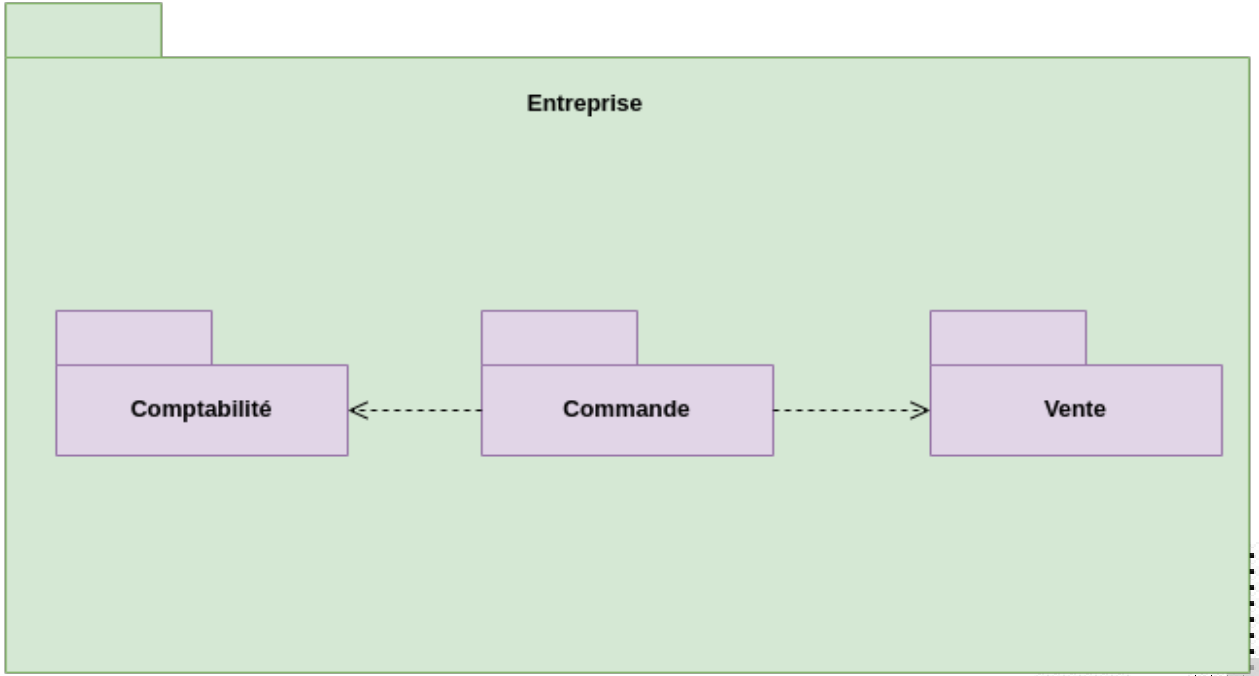
II.2 Diagramme de package (2)

Un package dans le langage de modélisation unifié permet :

- Pour regrouper des éléments
- Pour fournir un espace de noms (namespace) pour les éléments groupés
- Un package peut contenir d'autres packages, permettant ainsi une organisation hiérarchique des packages.
- Les éléments UML peuvent être regroupés en packages.

Partie I : les bases en UML
II. Les types de diagrammes
II.2 Diagramme de package (3)

L'illustration ci-dessous montre un exemple de diagramme de package utilisé pour représenter la composition d'une entreprise.



Partie I : les bases en UML
II. Les types de diagrammes
II.2 Diagramme de package (4)
II.2.1 Notations
Les diagrammes de packages sont utilisés pour structurer les systèmes de haut niveau. Les packages sont utilisés pour organiser un grand système contenant des diagrammes, des documents et d’autres livrables clés. En d’autres termes, les packages peuvent également être utilisés dans le cadre d’autres diagrammes.

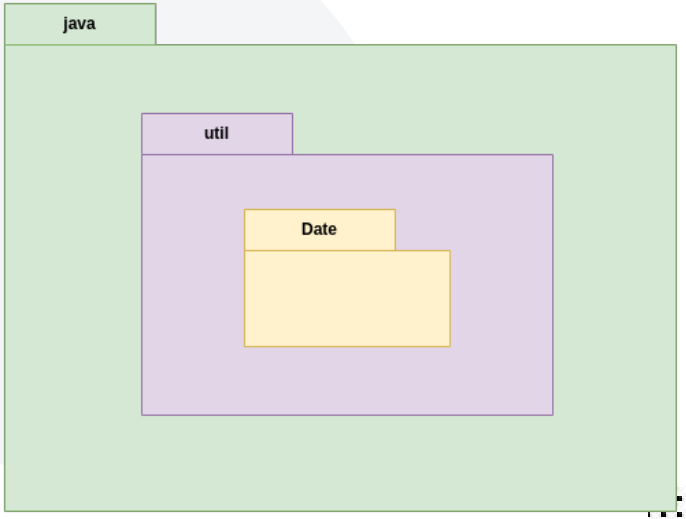
Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de package (5)

II.2.1 Notations

- **Packages imbriqués et hiérarchiques**

Un package peut être représenté comme une structure hiérarchique avec des packages imbriqués. Les modules atomiques pour les packages imbriqués sont généralement des diagrammes de classes.



Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de package (6)

II.2.1 Notations

- **Règles**

1. Le nom des packages doit être unique au sein d'un système. Cependant, il est permis que les classes de différents packages portent le même nom. Par exemple, Package::Produit & Vente::Produit sont autorisés.
2. Les utilisateurs doivent éviter d'utiliser le nom du package fourni par le langage de programmation. Par exemple, Java fournit Date sous forme de package. Ainsi, les programmeurs doivent construire un package nommé Date.
3. Les packages peuvent inclure des diagrammes entiers, le nom des composants uniquement ou aucun composant du tout.

Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de package (6)

II.2.1 Notations

- **Dépendance**

Il existe deux sous-types impliqués dans la dépendance. Ce sont <<access>> et <<import>>. Bien qu'il existe deux stéréotypes, les utilisateurs peuvent utiliser leur propre stéréotype pour représenter le type de dépendance entre deux packages.

<<import>> : un package importe les fonctionnalités d'un autre package.

Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de package (6)

II.2.1 Notations

- **Dépendance**

<<access>> : un package nécessite l'aide des fonctions d'un autre package.

Partie I : les bases en UML
II. Les types de diagrammes

II.2 Diagramme de package (6)
II.2.2 Quand utiliser le diagramme de package ?

L'UML ne traite pas les diagrammes de packages comme une technique distincte. Il est souvent utile de les combiner en regroupant d'autres éléments du modèle dans différents packages sur le même diagramme. Les diagrammes de packages peuvent être utiles de plusieurs manières, telles que :

- Pour créer une vue d'ensemble d'un grand ensemble d'éléments de modèle
- Pour organiser un grand modèle
- Pour regrouper des éléments liés
- Pour séparer les espaces de noms

Partie I : les bases en UML
II. Les types de diagrammes
II.3 Diagramme de séquence

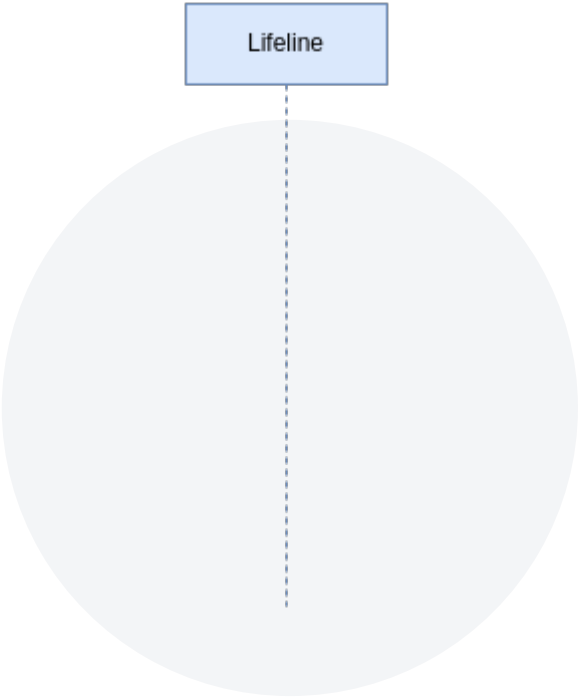
Un diagramme de séquence décrit une interaction entre un ensemble d'objets ayant participé à une collaboration (ou à un scénario), classés dans un ordre chronologique ; il montre les objets participant à l'interaction par leurs « lignes de vie » et les messages qu'ils s'envoient les uns aux autres.

Partie I : les bases en UML
II. Les types de diagrammes

II.3 Diagramme de séquence

II.3.2 Notations (1)

- Une **ligne de vie** représente un participant individuel à l'interaction.



Partie I : les bases en UML
II. Les types de diagrammes

II.3 Diagramme de séquence

II.3.2 Notations (2)

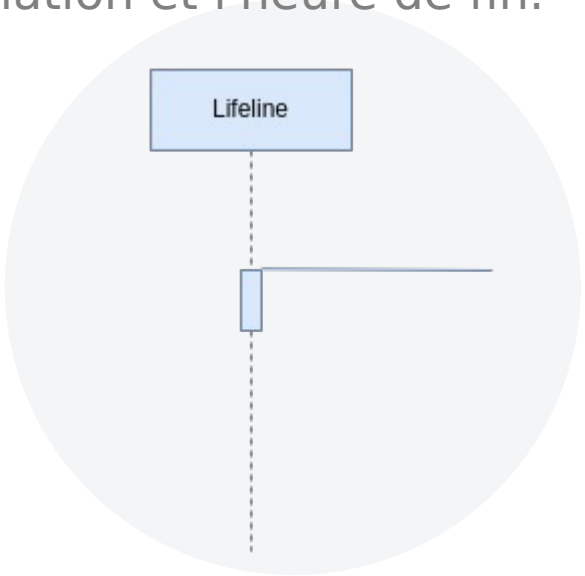
- Un **acteur** est un type de rôle joué par une entité qui interagit avec le sujet (par exemple, en échangeant des signaux et des données). Un acteur peut également être externe au sujet (c'est-à-dire dans le sens où une instance d'un acteur ne fait pas partie de l'instance de son sujet correspondant). Ils représentent généralement les rôles joués par des utilisateurs humains, du matériel externe ou d'autres sujets.

Partie I : les bases en UML
II. Les types de diagrammes

II.3 Diagramme de séquence

II.3.2 Notations (3)

- Une **activation** (représentée par un rectangle fin sur une ligne de vie) représente la période pendant laquelle un élément effectue une opération. Le haut et le bas du rectangle sont respectivement alignés avec l'heure d'initiation et l'heure de fin.

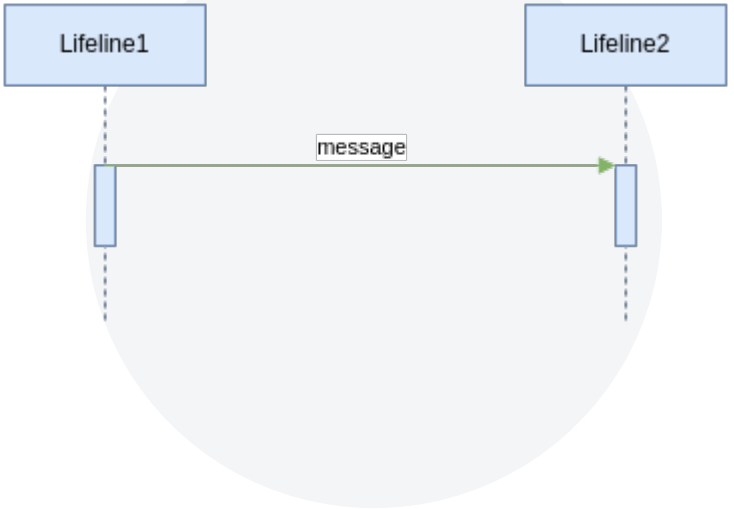


Partie I : les bases en UML
II. Les types de diagrammes

II.3 Diagramme de séquence

II.3.2 Notations (4)

- Un **message d'appel** définit une communication particulière entre les lignes de vie d'une interaction, qui représente une invocation du fonctionnement de la ligne de vie cible.

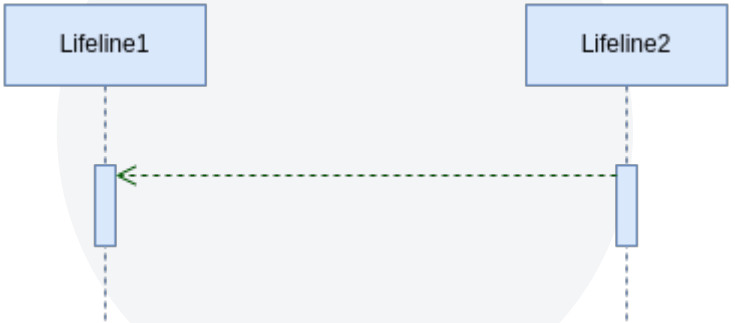


Partie I : les bases en UML
II. Les types de diagrammes

II.3 Diagramme de séquence

II.3.2 Notations (5)

- Un **message de retour** définit une communication particulière entre les lignes de vie d'une interaction, qui représente le retour d'informations à l'appelant d'un ancien message correspondant.

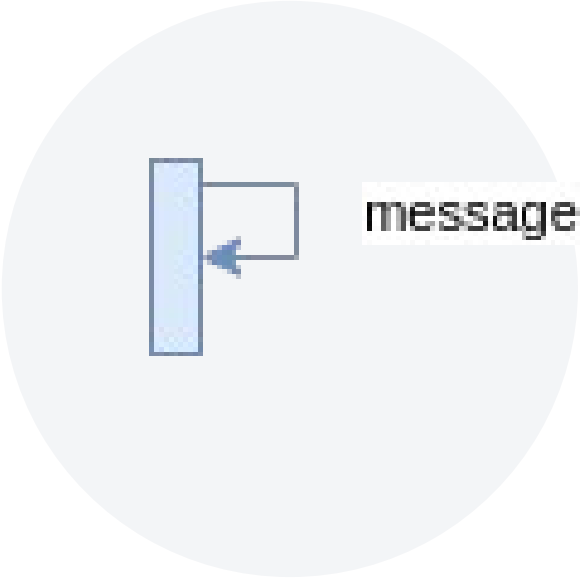


Partie I : les bases en UML
II. Les types de diagrammes

II.3 Diagramme de séquence

II.3.2 Notations (6)

- Un **auto-message** définit une communication particulière entre les lignes de vie d'une interaction, qui représente l'invocation du message de la même ligne de vie.



II.3 Diagramme de séquence

II.3.2 Notations (8)

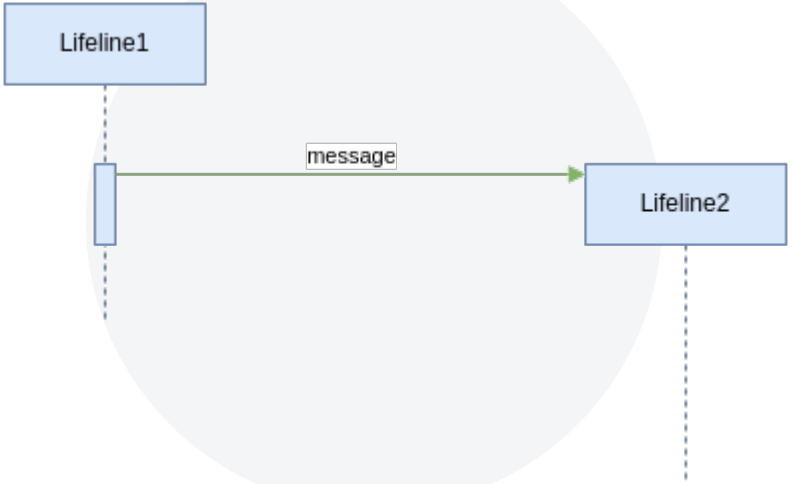
- Un **message récursif** définit une communication particulière entre les lignes de vie d'une interaction, qui représente l'invocation du message de la même ligne de vie. Sa cible pointe vers une activation en plus de l'activation à partir de laquelle le message a été invoqué.

Partie I : les bases en UML
II. Les types de diagrammes

II.3 Diagramme de séquence

II.3.2 Notations (9)

- Un **message de création** définit une communication particulière entre les lignes de vie d'une interaction, qui représente l'instanciation de la ligne de vie (cible).



Partie I : les bases en UML
II. Les types de diagrammes

II.3 Diagramme de séquence

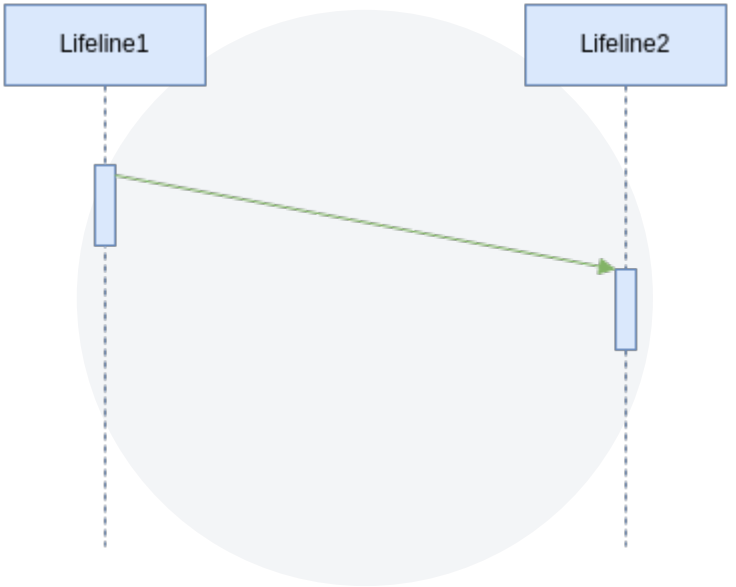
II.3.2 Notations (10)

- Un **message de destruction** définit une communication particulière entre les lignes de vie d'une interaction, qui représente la demande de destruction du cycle de vie de la ligne de vie cible.

II.3 Diagramme de séquence

II.3.2 Notations (11)

- Un **message de durée** définit une communication particulière entre les lignes de vie d'une interaction, qui montre la distance entre deux instants temporels pour une invocation de message.

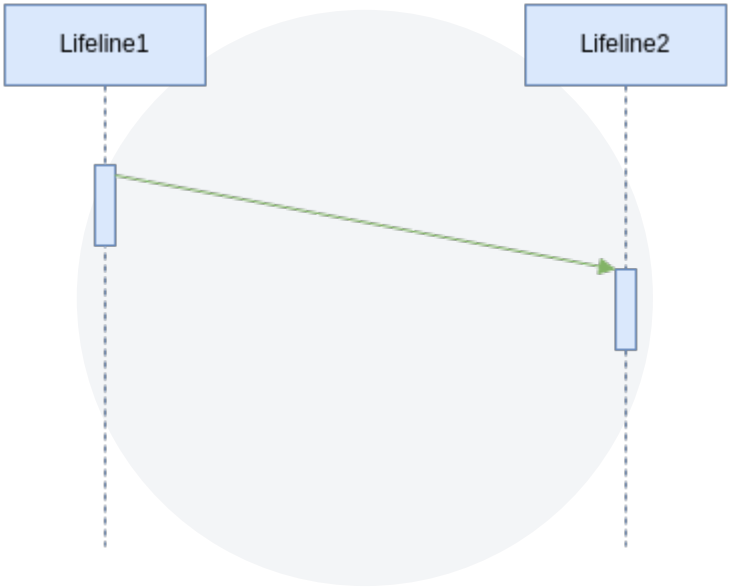


Partie I : les bases en UML
II. Les types de diagrammes

II.3 Diagramme de séquence

II.3.2 Notations (12)

- Un **message de durée** définit une communication particulière entre les lignes de vie d'une interaction, qui montre la distance entre deux instants temporels pour une invocation de message.



Partie I : les bases en UML
II. Les types de diagrammes

II.3 Diagramme de séquence

II.3.2 Notations (13)

- Une **note** (**commentaire**) donne la possibilité de joindre diverses remarques à des éléments. Un commentaire n'a aucune force sémantique, mais peut contenir des informations utiles à un modélisateur.

Partie I : les bases en UML
II. Les types de diagrammes

II.3 Diagramme de séquence
II.3.2 Notations (14)

- Dans un diagramme de séquence UML, les **fragments combinés** vous permettent d'afficher des boucles, des branches et d'autres alternatives. Un fragment combiné se compose d'un ou plusieurs opérandes d'interaction, et chacun d'eux contient un ou plusieurs messages, utilisations d'interaction ou fragments combinés.

- Un fragment de séquence est représenté par une boîte appelée fragment combiné, qui renferme une partie des interactions dans un diagramme de séquence. L'opérateur fragment (dans le cornet supérieur gauche) indique le type de fragment. Les types de fragments incluent **ref, assert, loop, break, alt, opt et neg, ref, sd**.

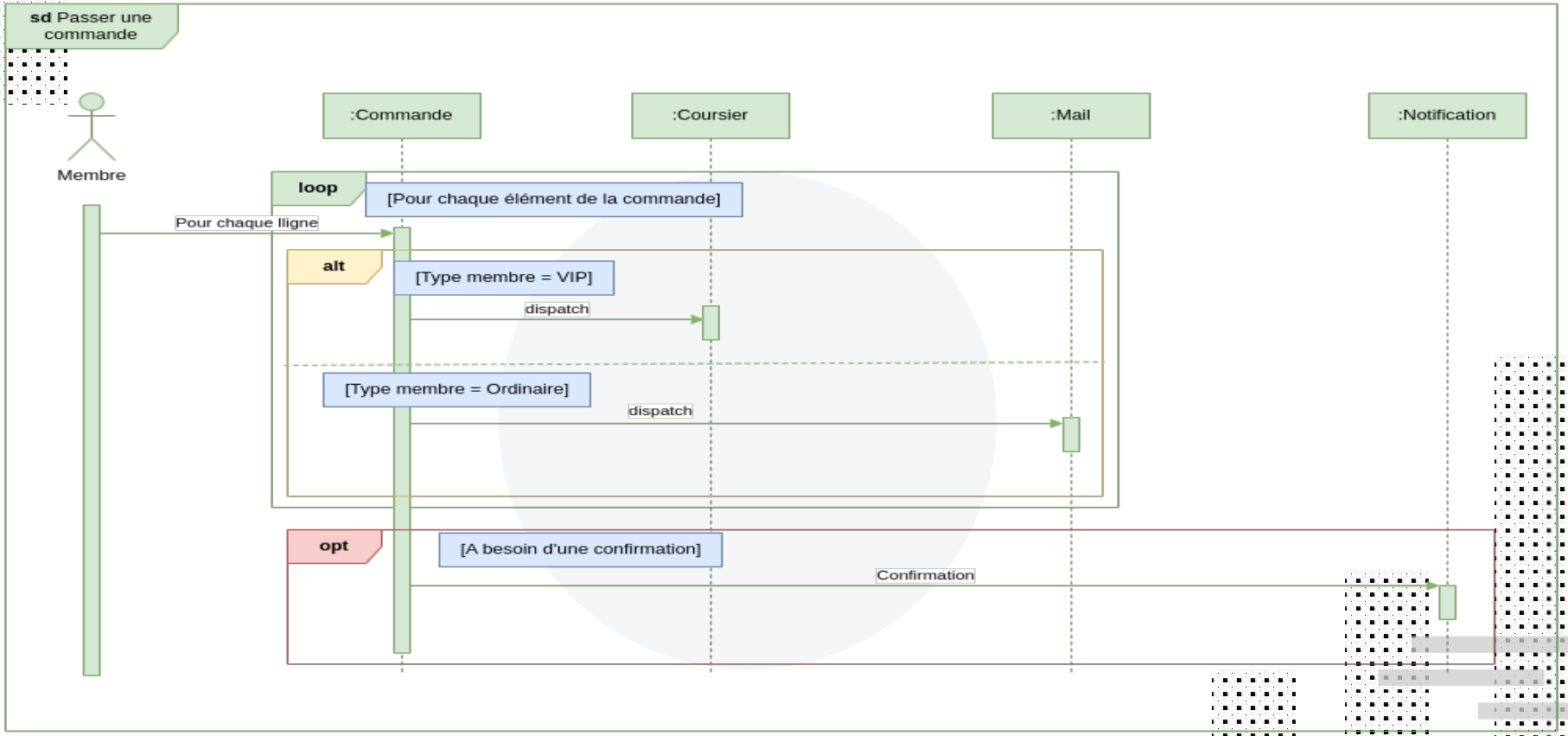
Partie I : les bases en UML
II. Les types de diagrammes
II.3 Diagramme de séquence
II.3.2 Notations (15)

Opérateur	Signification
alt	Fragments multiples alternatifs : seul celui dont la condition est vraie s'exécutera.
opt	**Optionel**: le fragment s'exécute uniquement si la condition fournie est vraie. Équivalent à un alt avec une seule trace.
par	**Parallèle** : chaque fragment est exécuté en parallèle.
loop	**Boucle** : le fragment peut s'exécuter plusieurs fois, et la garde indique la base de l'itération.
critical	**Région critique** : le fragment ne peut avoir qu'un seul thread qui l'exécute à la fois.
neg	**Négatif** : le fragment montre une interaction non valide.
ref	**Référence** : fait référence à une interaction définie sur un autre diagramme. Le cadre est dessiné pour couvrir les lignes de vie impliquées dans l'interaction. Vous pouvez définir des paramètres et une valeur de retour.
sd	**Diagramme de séquence** : utilisé pour entourer un diagramme de séquence entier.

Partie I : les bases en UML
II. Les types de diagrammes
II.3 Diagramme de séquence
II.3.2 Notations (16)

- Exemple : passer une commande
  Un membre d'un navire qui souhaite passer une commande en ligne. L'article commandé sera envoyé au membre soit par coursier, soit par courrier ordinaire selon son statut de membre (VIP, Adhésion Ordinaire). En option, la boutique enverra au membre une notification de confirmation si le membre a opté pour l'option de notification lors de la commande.

Partie I : les bases en UML
II. Les types de diagrammes
II.3 Diagramme de séquence
II.3.2 Notations (17)



Partie I : les bases en UML
II. Les types de diagrammes
II.3 Diagramme de séquence
II.3.3. Quand dessiner un diagramme de séquence ?

- Modéliser une interaction de haut niveau entre les objets actifs dans un système
- Modéliser l'interaction entre les instances d'objets au sein d'une collaboration qui réalise un cas d'utilisation
- Modéliser l'interaction entre les objets au sein d'une collaboration qui réalise une opération
- Modélisez soit des interactions génériques (montrant tous les chemins possibles à travers l'interaction), soit des instances spécifiques d'une interaction (montrant un seul chemin à travers l'interaction)

Partie I : les bases en UML
II. Les types de diagrammes
II.3 Diagramme de séquence
II.3.3. Comment dessiner un diagramme de séquence ? (1)

- Identifier un ensemble d'objets qui participeront à la collaboration générale (ou au scénario de cas d'utilisation)
  a) Si vous dérivez le diagramme de séquence sur la base d'un scénario d'un cas d'utilisation, sélectionnez d'abord les scénarios normaux.
  b) Vous devez connaître le ou les principaux acteurs qui activent le cas d'utilisation.

- Considérez le premier point du scénario (ou si vous l'obtenez du premier point du flux d'événements d'un cas d'utilisation)

Partie I : les bases en UML
II. Les types de diagrammes
II.3 Diagramme de séquence
II.3.3. Comment dessiner un diagramme de séquence ? (2)

- Considérez ce que le système doit faire pour répondre à l'acteur, lorsque l'acteur envoie le message au système
  a) Qu'est-ce que le système doit gérer avant la réponse du message de retour du système ?
  b) Par exemple. Un client a inséré une carte ATM dans la machine, le système affichera le « code PIN d'entrée » dans le scénario normal, n'est-ce pas ?
  c) Devinez, que sera géré à l'intérieur de l'ADM par un ensemble d'objets à « l'arrière » du système ? Quelque chose comme lire et vérifier la carte ATM (lecteur de carte), lire les informations de carte du titulaire de la carte (par la banque) et demander le code PIN, ou renvoyer "type de carte invalide, insérer une autre carte", et etc.
  d) De cette façon, vous identifierez les objets et les opérations candidats de l'application cible pour ce scénario particulier et vous pourrez également utiliser ces informations comme base pour dériver le diagramme de classes de manière incrémentielle.

II.3 Diagramme de séquence

II.3.3. Comment dessiner un diagramme de séquence ? (3)

- Répétez chacun des points du scénario (ou déroulement de l'événement) et jusqu'à ce que vous ayez terminé tous les points du scénario.

Partie I : les bases en UML
II. Les types de diagrammes
II.4 Diagramme d’activité
II.4.1. Définition

Les diagrammes d'activités décrivent comment les activités sont coordonnées pour fournir un service qui peut se situer à différents niveaux d'abstraction. En règle générale, un événement doit être réalisé par certaines opérations, en particulier lorsque l'opération est destinée à réaliser un certain nombre de choses différentes qui nécessitent une coordination, ou comment les événements d'un cas d'utilisation unique sont liés les uns aux autres, en particulier dans les cas d'utilisation où les activités peuvent se chevaucher et nécessiter une coordination. Il convient également à la modélisation de la façon dont un ensemble de cas d'utilisation se coordonne pour représenter les flux de travail métier.

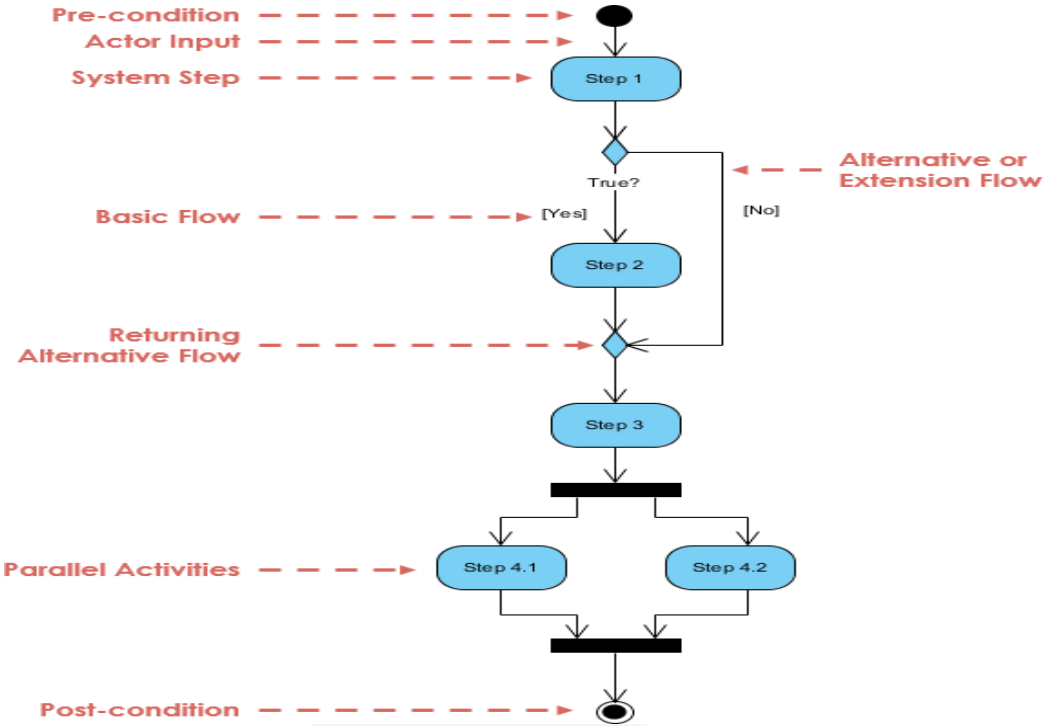
Partie I : les bases en UML
II. Les types de diagrammes

II.4 Diagramme d’activité

II.4.2. Mode d’emploi

- Identifier les cas d'utilisation candidats, grâce à l'examen des flux de travail métier
- Identifier les conditions préalables et postérieures (le contexte) pour les cas d'utilisation
- Modéliser les workflows entre/au sein des cas d'utilisation
- Modéliser des flux de travail complexes dans les opérations sur les objets
- Modéliser en détail des activités complexes dans un diagramme d'activité de haut niveau

Partie I : les bases en UML
II. Les types de diagrammes
II.4 Diagramme d'activité
II.4.2. Exemples : (1)

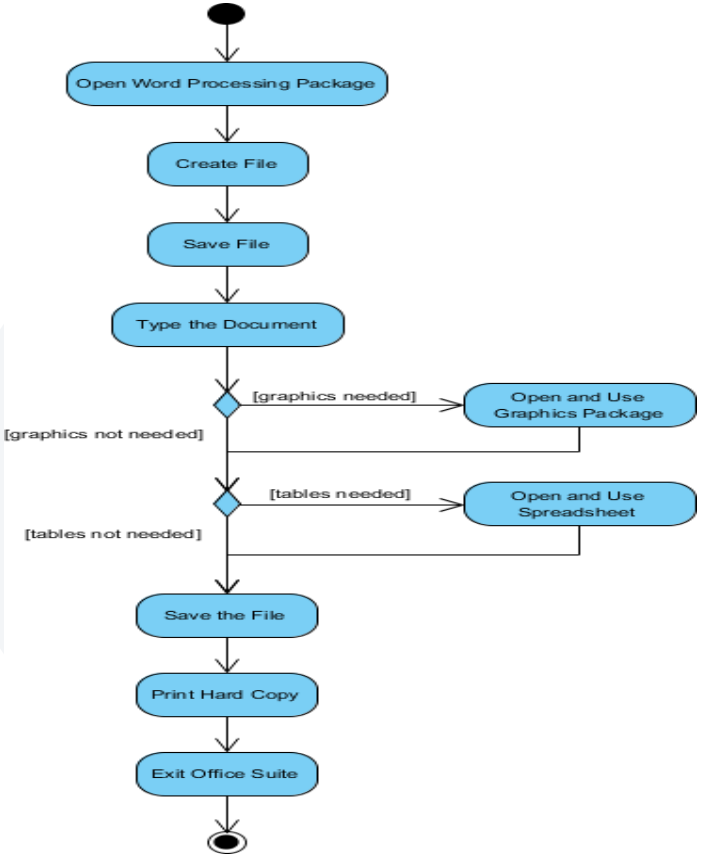


Partie I : les bases en UML
II. Les types de diagrammes
II.4 Diagramme d’activité
II.4.2. Exemples : modélisation d'un traitement de texte (2)

L'exemple de diagramme d'activités ci-dessous décrit le flux de travail d'un traitement Word pour créer un document à travers les étapes suivantes :

- Ouvrez le package de traitement de texte.
- Créez un fichier.
- Enregistrez le fichier sous un nom unique dans son répertoire.
- Tapez le document.
- Si des graphiques sont nécessaires, ouvrez le package graphique, créez les graphiques et collez-les dans le document.
- Si une feuille de calcul est nécessaire, ouvrez le package de feuille de calcul, créez la feuille de calcul et collez-la dans le document.
- Enregistrez le fichier.
- Imprimez une copie papier du document.
- Quittez le package de traitement de texte.

Partie I : les bases en UML
II. Les types de diagrammes
II.4 Diagramme d'activité
II.4.2. Exemples : modélisation d'un traitement de texte (2)



Partie I : les bases en UML
II. Les types de diagrammes
II.4 Diagramme d’activité
II.4.2. Exemples : passer une commande (3)

Compte tenu de la description du problème lié au workflow de traitement d'une commande, modélisons la description en représentation visuelle à l'aide d'un diagramme d'activité :

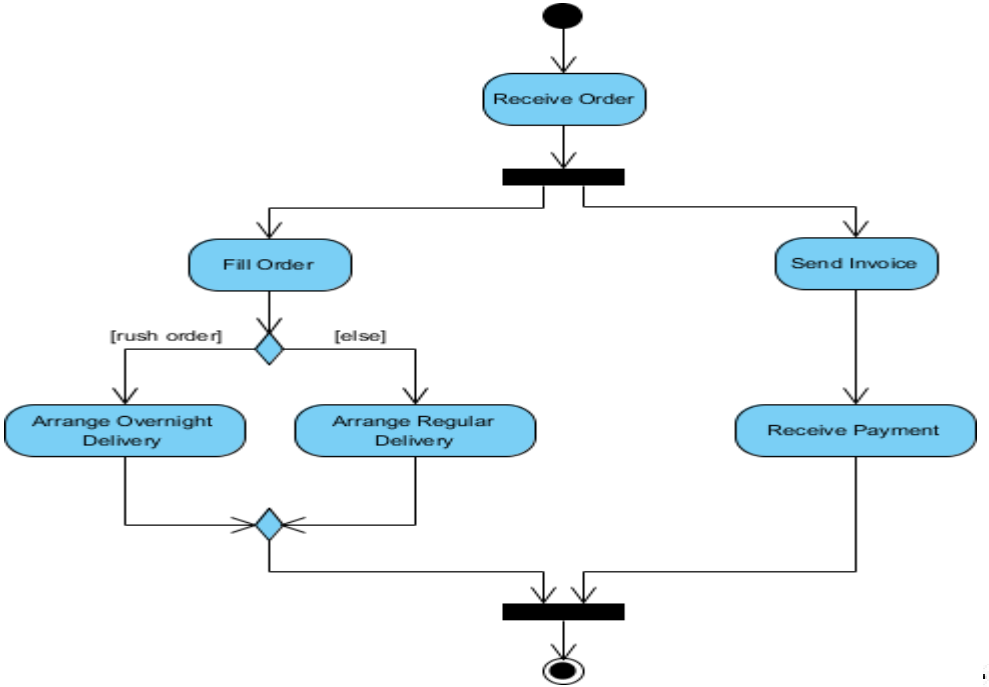
- Une fois la commande reçue, les activités se divisent en deux ensembles d'activités parallèles. Un côté remplit et envoie la commande tandis que l’autre gère la facturation.

- Du côté de la commande remplie, le mode de livraison est décidé de manière conditionnelle. En fonction de la condition, l'activité Livraison de nuit ou l'activité Livraison régulière est exécutée.

- Finalement les activités parallèles se combinent pour clôturer la commande.

Partie I : les bases en UML
II. Les types de diagrammes
II.4 Diagramme d'activité
II.4.2. Exemples : passer une commande (3)

L'exemple de diagramme d'activité ci-dessous visualise le flux sous forme graphique.



Partie I : les bases en UML
II. Les types de diagrammes

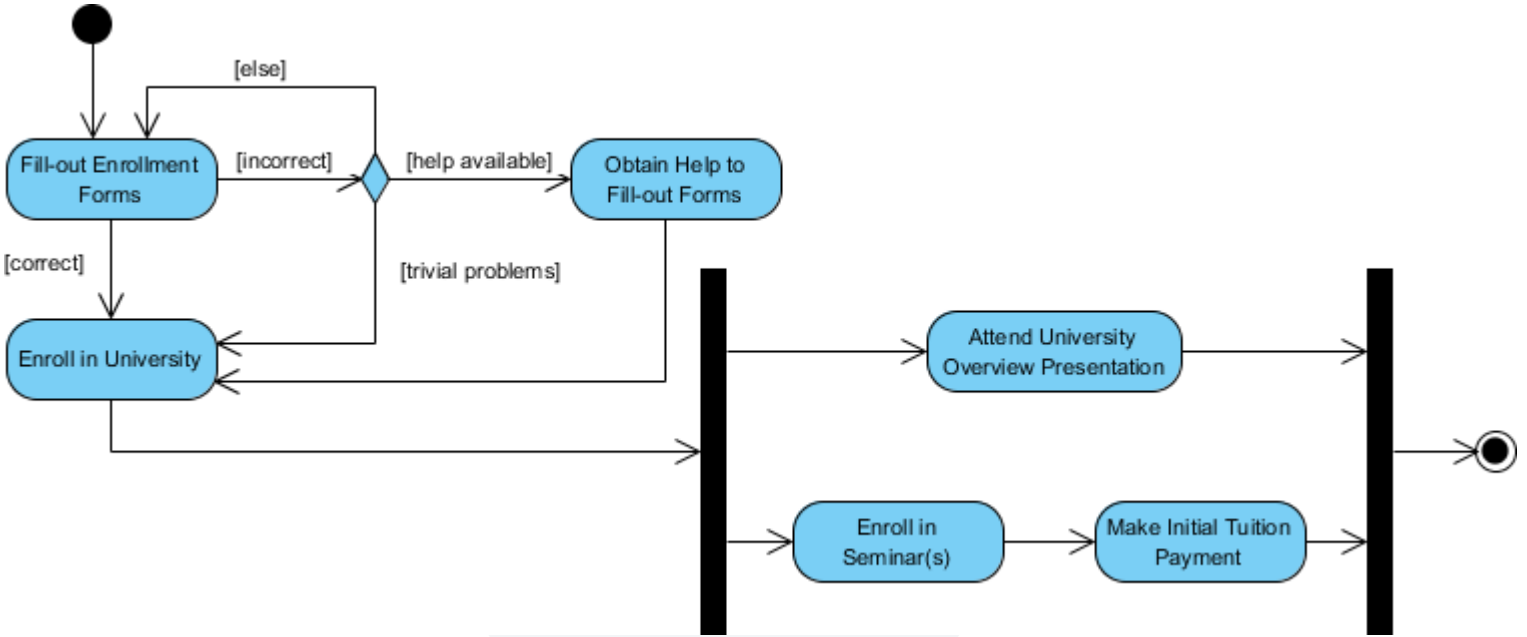
II.4 Diagramme d’activité

II.4.2. Exemples : inscription des étudiants (4)

Cet exemple de diagramme d'activités UML décrit un processus d'inscription d'étudiants dans une université comme suit :

- Un candidat souhaite s'inscrire à l'université.
- Le candidat remet une copie dûment remplie du formulaire d’inscription.
- L’agent chargé des inscriptions inspecte les formulaires.
- Il détermine que les formulaires ont été correctement remplis.
- Il informe l'étudiant d'assister à la présentation générale de l'université.
- Il aide l'étudiant à s'inscrire aux séminaires
- Il demande à l'étudiant de payer les frais de scolarité initiaux.

Partie I : les bases en UML
II. Les types de diagrammes
II.4 Diagramme d'activité
II.4.3. Exemples : inscription des étudiants (4)

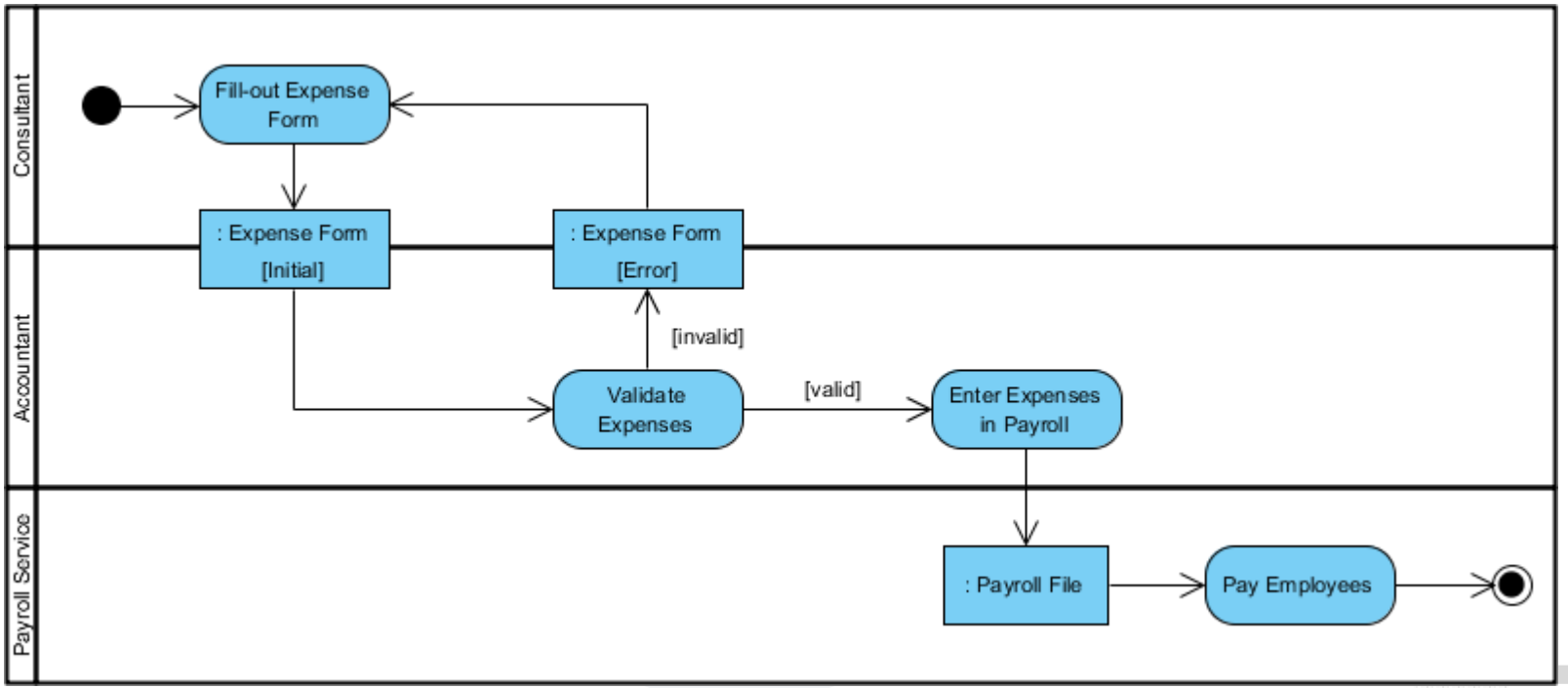


II.4 Diagramme d’activité

II.4.3. Diagramme d’activité en couloirs (swimline)

Un couloir est un moyen de regrouper les activités effectuées par le même acteur sur un diagramme d'activités ou un diagramme d'activités ou de regrouper les activités dans un seul thread. Voici un exemple de diagramme d'activités en couloirs pour modéliser la soumission des dépenses de personnel :

Partie I : les bases en UML
II. Les types de diagrammes
II.4 Diagramme d'activité
II.4.3. Diagramme d'activité en tiroirs (2)

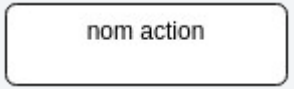


Partie I : les bases en UML
II. Les types de diagrammes

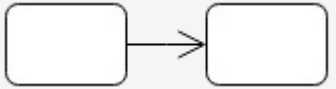
II.4 Diagramme d'activité
II.4.3. Notations (1)

- Activité

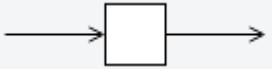
- Action



- Flux de contrôle



- Flux d'objets



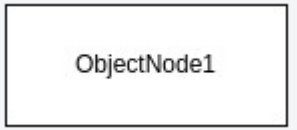
Partie I : les bases en UML
II. Les types de diagrammes

II.4 Diagramme d'activité
II.4.3. Notations (2)

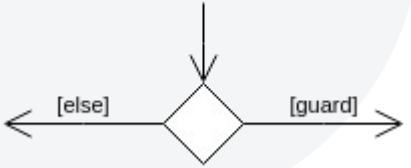
- Noeud initial

- Nœud final d'activité

- Noeud Objet



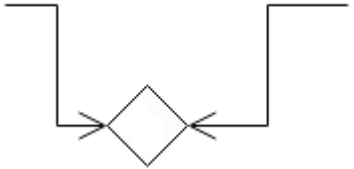
- Noeud de décision



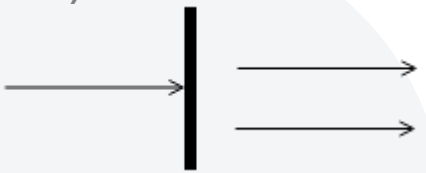
Partie I : les bases en UML
II. Les types de diagrammes

II.4 Diagramme d'activité
II.4.3. Notations (3)

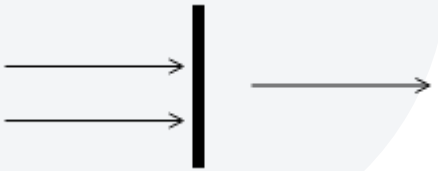
- Noeud de fusion



- Noeud de bifurcation (fork)



- Noeud de jonction (join)



- Couloir et cloison

Partie I : les bases en UML
II. Les types de diagrammes
II.4 Diagramme d’activité
II.4.3. Notations (4)

- Couloir et cloison