Table des matières

Modélisation objet avec UML 2
Introduction 2
Modèle de système informatique : 2
Pourquoi UML pour la modélisation Objet? 3
Représentation dynamique du système 5
Le diagramme de cas d'utilisation : (Use case) 5
Définitions 5
Objectifs des cas d'utilisation 5
Représentation 6
Comment identifier les acteurs? 6
Comment identifier les cas d'utilisation? 6
Description textuelle des cas d'utilisation 7
Relation entre les cas d'utilisation 10
Modélisation statique 12
Introduction : 12
Définition :Le diagramme de classes 12
Liens entre classes ou association entre classes. 13
Notion de cardinalité ou multiplicité 14
Relation navigable: 15
Agrégation : 15
Généralisation, super-classe, sous-classe 17
Classe d'association 19
Notion de package 20
Modélisation dynamique : le diagramme de séquence 22
Définition et exemple 22
Les diagrammes de séquences selon UML 2.0 24
Opérateur alt (alternative) 24
Opérateur Loop 25
Opérateur Opt (option) 26
Opérateur Break 26
Opérateur Par (parallel) 27
Critical 27
Modélisation dynamique : le diagramme d'état de transition 28

Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

Modélisation objet avec UML

Introduction

Le développement des systèmes est une tâche d'une grande envergure et un investissement important pour toute entreprise. La modélisation des systèmes déjà existants ou d'un système à construire est une étape importante du cycle de développement des systèmes- La modélisation permet de visualiser, souvent d'une manière graphique, un système tel qu'il est ou comment nous souhaitons qu'il va être.
La modélisation d'une manière générale, aide à l'élaboration et à la structuration des idées et permet de faciliter la communication entre humains.

Un modèle est une abstraction de la réalité. Modéliser consiste à identifier les caractéristiques intéressantes ou pertinentes d'un système dans le but de pouvoir l'étudier du point de vue de ses caractéristiques.
Avant de construire une voiture, on conçoit des plans, des tests, des essais de moteurs… c'est la même chose avant de construire une maison, un plan est conçu, les tests par rapport à la résistance des matériaux sont réalisés etc…. Les météorologues utilisent des modèles pour prévoir la météo. On utilise les modèles dans tous les domaines scientifiques et de la réingénierie.
Un bon modèle doit posséder deux caractéristiques essentielles.

- Il doit faciliter la compréhension du phénomène (système) étudié, il réduit la complexité
- Il doit permettre de simuler le phénomène (système) étudié, il reproduit ses comportements.

Modèle de système informatique :

Un modèle d'un système informatique aide mieux à percevoir les relations et les interactions à l'intérieur de celui-ci. Il doit permettre de visualiser les conséquences de modifications apportées au système, il doit également permettre de visualiser les raisons du comportement du système par rapport à une situation donnée. C'est donc un guide pour construire un système fiable et stable. Le modèle doit également aider à documenter le système construit.
L'expérience du passé dans la construction de modèles suggère quatre principes de bases :

- Le choix du modèle initial a une grande influence sur la manière dont le modèle est attaqué et une solution ébauchée.

Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

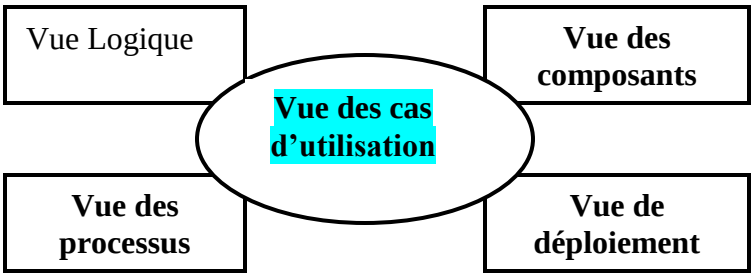
- Tout modèle peut être exprimé à divers niveaux de précision.
- Les meilleurs modèles sont ceux qui sont proches de la réalité.
- Aucun modèle ne peut prétendre résoudre à lui seul un problème complexe. Tout système complexe (non trivial) est approché à l'aide d'un petit nombre de modèles pratiquement indépendants.

Pourquoi UML pour la modélisation Objet?

UML opte pour **l'élaboration des modèles**, plutôt que pour une approche qui impose une barrière stricte entre analyse et conception. Les modèles d'analyse et de conception ne diffèrent que par leur niveau de détail, il n'y a pas de différence dans les concepts utilisés. UML n'introduit pas d'éléments de modélisation propres à une activité (analyse, conception...) ; le langage reste le même à tous les niveaux d'abstraction.
Cette approche simplificatrice facilite le passage entre les niveaux d'abstraction. L'élaboration encourage une approche non linéaire, les "retours en arrière" entre niveaux d'abstraction différents sont facilités et la traçabilité entre modèles de niveaux différents est assurée par l'unicité du langage.

- UML **favorise donc le prototypage**, et c'est là une de ses forces. En effet, modéliser une application n'est pas une activité linéaire. Il s'agit d'une tâche très complexe, qui nécessite une approche itérative, car il est plus efficace de construire et valider par étapes, ce qui est difficile à cerner et maîtriser.
- UML permet donc non seulement de représenter et de manipuler les concepts objet, il sous-entend une démarche d'analyse qui permet de concevoir une solution objet de manière itérative, grâce aux diagrammes, qui supportent l'abstraction.

Pour exprimer les diverses perspectives de l'architecture d'un système, les auteurs d'UML proposent **4+1 vues**, selon le schéma suivant :



Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

Cette approche propose que plusieurs perspectives concourent à l'expression de l'architecture d'un système et explique qu'il est nécessaire de garantir la séparation et l'indépendance des différentes perspectives; l'évolution de l'une des perspectives ne doit pas avoir impact (sinon limitée) sur les autres.

1. *La vue logique* : cette vue exprime la perspective abstraite de la solution en termes de classes, d'objets, de relations, de machine à états de transition etc.. de manière indépendante des langages de programmation et des environnements de développement utilisés pour les mettre en œuvre. Il s'agit d'exprimer le problème de façon abstraite. Les concepts utilisés incluent les concepts de la majorité des méthodes orientés objets existantes. Cette vue concerne *l'intégrité de conception*

2. *La vue des composants :* cette vue exprime la perspective physique de l'organisation du code en terme de modules, des composants et surtout des concepts du langage et de l'environnement d'implémentation. Cette perspective dépend du choix du langage utilisé. Dans cette perspective, le concepteur est surtout intéressé par des aspects de gestion de codes, d'ordre de compilation, de réutilisation.
   UML offre des concepts adaptés comme les modules, l'interface, les composants, les relations de dépendance ect… Malheureusement la plus part des concepteurs ignorent cette vue. Cette vue concerne *l'intégrité de gestion*

3. *La vue des processus* : cette vue exprime la perspective sur les activités concurrentes et parallèles (tâches et processus) du système. On y trouve les activités parallèles, et leur communication et synchronisation. Cette vue concerne *l'intégrité d'exécution*.

4. L*a vue de déploiement* : elle exprime la répartition du système à travers un réseau de calculateurs et de nœuds logiques de traitement. Cette vue est utile pour décrire la répartition du système réparti, elle concerne *l'intégrité de performance*

5. *La vue des cas d'utilisation* : cette vue guide et justifie les autres en ce sens que la modélisation fondée sur les scénarios (cas d'utilisation) constitue ce que l'on fait de mieux aujourd'hui. Cette approche constitue l'unique moyen de guider la modélisation, de trouver le bon modèle,

Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

UML propose 13 diagrammes pour modéliser un système. Selon la vue que l'on veut décrire, statique ou dynamique, ces diagrammes sont :

Représentation statique du système :

- Le diagramme de cas d'utilisation
- Le diagramme Objets
- Le diagramme de classes
- Le diagramme des composants
- Le diagramme de déploiement
- Diagramme de packages
- Diagramme de structure composite

Représentation dynamique du système

- Le diagramme de collaboration
- Le diagramme de séquences
- Le diagramme d'état de transition
- Le diagramme d'activités
- Diagramme de communication
-

Le diagramme de cas d'utilisation : (Use case)

Définitions

Les cas d'utilisation décrivent le comportement du système du point de vue de l'utilisateur. Ils permettent de définir les limites du système et les relations entre le système et son environnement. Un cas d'utilisation est une manière spécifique d'utiliser le système. C'est l'image d'une fonctionnalité en réponse à la stimulation d'un acteur externe

Objectifs des cas d'utilisation

- Permettent de structurer les besoins des utilisateurs et les objectifs correspondants d'un système.
- Ils centrent l'expression des exigences du système sur ses utilisateurs Ils se limitent aux préoccupations "réelles" des utilisateurs ; ils ne présentent pas de solutions d'implémentation et ne forment pas un inventaire fonctionnel du système.
- Ils identifient les utilisateurs du système et leur interaction avec celui-ci

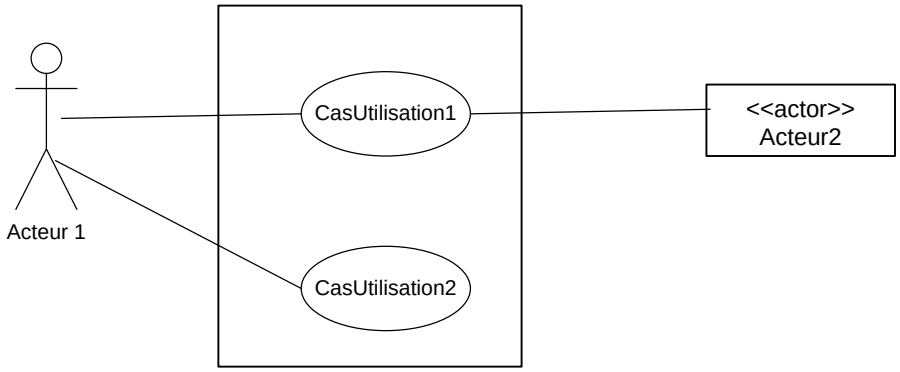
Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

Un cas d'utilisation est un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable pour un acteur particulier du système. Il permet de décrire ce que le futur système devra faire sans spécifier comment il le fera.

Acteur : entité externe qui agit sur le système. Un acteur peut consulter et /ou modifier l'état du système en émettant et/ou en recevant des messages susceptibles d'être porteurs de données.

En réponse à l'action d'un acteur, le système fournit un service qui correspond à son besoin. Un acteur peut être une personne (utilisateur humain) ou les autres systèmes connexes qui interagissent directement avec le système.

Représentation



Les noms des cas d'utilisation doit être un verbe à l'infinitif suivi d'un complément du point de vue de l'utilisateur

Comment identifier les acteurs?

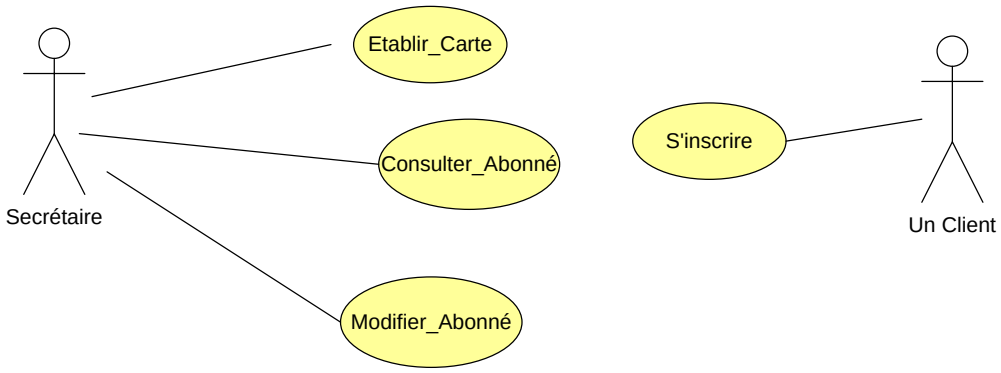
- Les utilisateurs humains (opérateur, administrateur, l'utilisateur, le formateur …)
- Les autres systèmes connexes qui interagissent directement avec le système.

Comment identifier les cas d'utilisation?

- Chaque cas d'utilisation doit décrire les exigences fonctionnelles du système.
- Chaque cas d'utilisation correspond à une fonction métier du système (besoins des utilisateurs et possibilités du système).
- Il convient donc de rechercher pour chaque acteur
    - Les différentes intentions métier avec lesquelles il utilise le système
    - Déterminer les services fonctionnels attendus du système.

  Exemple

Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html



Description textuelle des cas d'utilisation

Les cas d'utilisation ont besoin d'être décrits soit textuellement, soit en utilisant un autre diagramme. Deux parties sont importantes lors de la description d'un cas d'utilisation.

1) Le sommaire d'identification
    a. Le titre
    b. Résumé
    c. Acteurs
    d. Date de création
    e. Version
    f. Date de mise à jour
    g. Responsable
2) Le scénario nominal.
    a. Préconditions
    b. Scenario nominal (enchaînement des opérations dans le cas où le cas d'utilisation se déroule normalement.)
    c. Scénario alternatif
    d. Enchaînement des erreurs
    e. postconditions

Exemple. Caisse enregistreuse

Le déroulement normal des opérations d'une caisse enregistreuse (point de vente) est le suivant :

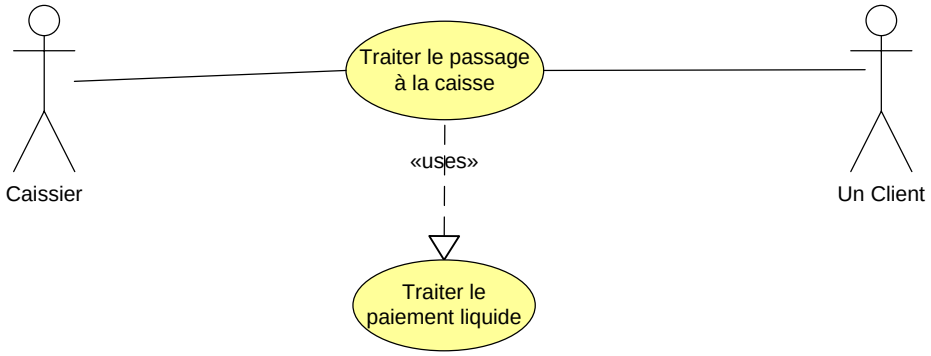
- Un client arrive à la caisse avec des articles à payer

Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

- Le caissier enregistre le n° d'identification de chaque article, ainsi que la quantité si elle est supérieure à 1
- La caisse affiche le prix de chaque article et son libellé.
- Lorsque tous les achats sont enregistrés, le caissier signale la fin de la vente.
- La caisse affiche le total des achats.
- Le client choisit son mode de paiement.
    1. Liquide: le caissier encaisse l'argent reçu, la caisse indique la monnaie à rendre au client.
    2. Carte de débit
    3. **Carte de crédit: un terminal bancaire fait partie de la caisse. Il transmet une demande d'autorisation en fonction du type de la carte.**
- La caisse enregistre la vente et imprime un ticket.
- Le caissier donne le ticket de caisse au client.

Lorsque le paiement est terminé, la caisse transmet les informations sur le nombre d'articles vendus au système de gestion des stocks.
Tous les matins, le responsable du magasin initialise les caisses pour la journée.

Voici le diagramme des cas d'utilisation (non complété) de la caisse enregistreuse.



Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

Description des cas d'utilisation

Titre: Traiter le passage à la caisse
Brève description: Un client arrive à la caisse avec des articles qu'il souhaite acheter.
Acteurs: Caissier (principal), Le client (secondaire)

Préconditions:
- Le terminal de point de vente est ouvert
- Un caissier y est connecté
- La base de données des produits est disponible

Postcondition.
La vente est enregistrée dans le terminal de vente.

Enchaînement des opérations (scénario nominal ou déroulement normal des opérations)

Acteurs	Système
1. Le cas d'utilisation débute lorsqu'un client arrive à la caisse avec des articles à acheter.	
2. Le caissier enregistre chaque article. Le caissier indique également la quantité pour chaque article s'il y a lieu.	3. le terminal de vente valide le numéro d'identification de chaque article. Le terminal de vente affiche la description et le prix de chaque article.
4. après avoir enregistré tous les articles, le caissier indique que la vente est terminée.	5. Le terminal de vente calcule et affiche le montant de la vente.
6. le caissier annonce le prix au client.	
7. Le client choisi de payer en liquide. Exécuter le cas «paiement en espèces	
	8. Le terminal de vente enregistre la vente et imprime un ticket.
9. le caissier remet le ticket de caisse	
10. Le client s'en va	

Scénarios alternatif

A1:Numéro d'identification du produit inconnu.

L'enchaînement de A1 démarre au point 3 du scénario nominal

le terminal de vente indique que le numéro du produit est inconnu. L'article ne peut être prix en vente. Le scénario reprend au point 2 s'il y a d'autres produit ou au point 4 s'il n'yen a pas

A2: le client demande l'annulation de l'article (article trop cher)

L'enchaînement de A2 démarre au point 2du scénario nominal

Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

2- le caissier demande l'annulation de l'article
3- le terminal de vente supprime l'article de la vente.

Le scénario nominal reprend au point 2, s'il y a d'autres articles ou au point 4 sinon.

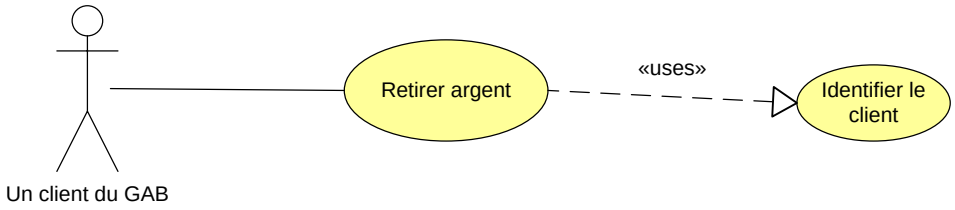
Questions : décrire le cas d'utilisation : Traiter le paiement liquide. Continuer le diagramme des cas d'utilisation de la caisse enregistreuse.

Relation entre les cas d'utilisation

Relation d'inclusion:«include» ou «utilise» ou «use»

Un cas d'utilisation de base (Emprunter un livre) incorpore explicitement un autre cas d'utilisation (s'identifier) de **façon obligatoire** à un endroit précis de son enchaînement. Le but est d'éviter de décrire plusieurs fois le même cas d'utilisation

Exemple:

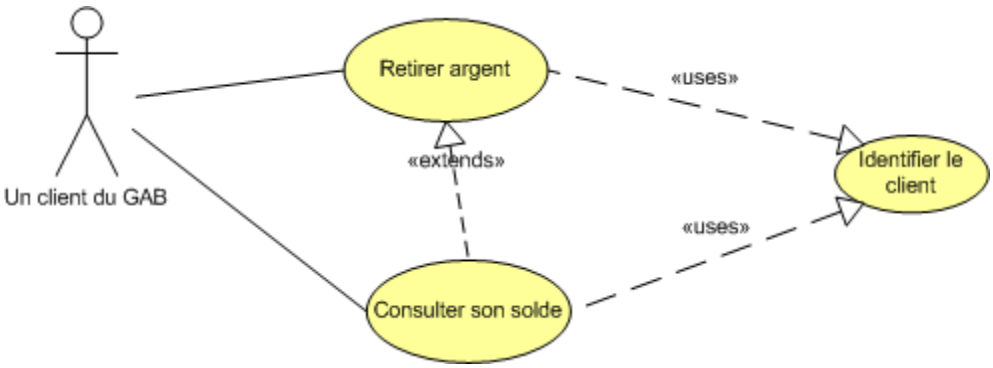


Relation d'extension: B est la suite logique de A

Un cas d'utilisation A (consulter la liste des abonnés) peut étendre son action, pas obligatoire, sur un autre cas d'utilisation B (imprimer la liste des abonnés). **Les deux cas peuvent s'exécuter de manière indépendante.**

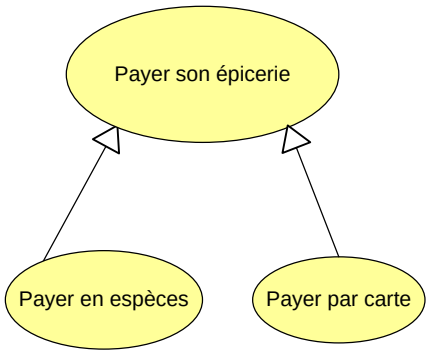
Exemple:

Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html



Les cas d'utilisation *Retirer argent* et *Consulter son solde* sont deux cas indépenadants, en ce sens qu'un client du GAB peut consulter son solde sans retirer de largent ou qu'il peut retirer de l'argent sans effectuer d'interrogation de solde.
Cependant le cas *Consulter son solde* peut etendre son action au cas *Retirer de l'argent*. Un client après avoir consulter son solde peut décider de retirer de l'argent.
Quelque soit le cas d'utilisation à exécuter (*Retirer argent* ou *Consulter son solde*), ils doivent faire appel (utilise) au d'utilisation *Identifier le client* (carte débit et nip). On ne peut retirer de l'argent si nous n'avons pas de carte bancaire valide et un nip valide

La relation de généralisation. Un cas A est une généralisation d'un cas B si B est un cas particulier de B
Cette relation de généralisation/spécialisation est présente dans la plupart des diagrammes UML et se traduit par le concept d'héritage dans les langages orientés objet.
Exemple :



Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

Modélisation statique

Introduction :

Le diagramme de classes est sans doute le diagramme le plus important à représenter pour les méthodes d'analyse orientées objet. C'est le point central de tout développement orienté objet.

On peut voir le diagramme de classes à différents niveau de développement. En analyse il permet de décrire la structure des entités manipulées par les utilisateurs. En conception, il permet de représenter un code orienté objet.

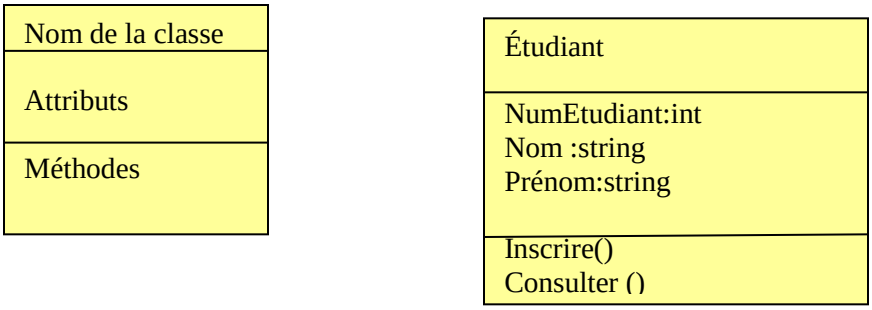
Définition :Le diagramme de classes

Un diagramme de classes est une collection d'éléments de modélisation statique qui montre la structure d'un modèle. Une classe représente la description d'un ensemble d'objets possédant les mêmes caratéristiques.

Un diagramme de classes fait abstraction des aspects dynamiques et temporels du système

Représentation d'une classe

Une classe est représentée par un rectangle séparé en trois parties:



Un attribut ou une méthode peut être de type :

Protégé, il est précédé du symbole #, visible aux sous classes de la classe

Privé, il est précédé du symbole –, visible à la classe seule

Public, il est précédé du symbole +, visible à tous les clients de la classe

Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

Liens entre classes ou association entre classes.

L'association est la relation la plus évidente qui existe entre classes, elle exprime une connexion sémantique bidirectionnelle entre deux classes.

Une association est une relation entre deux classes qui décrit les connexions structurelles entre leurs instances. Une association indique donc qu'il peut y avoir des liens entre des instances des classes associées.

Exemple1, association simple :

Comment représenter une personne qui travaille dans une seule compagnie? Ou encore dans une compagnie, nous avons plusieurs employés?

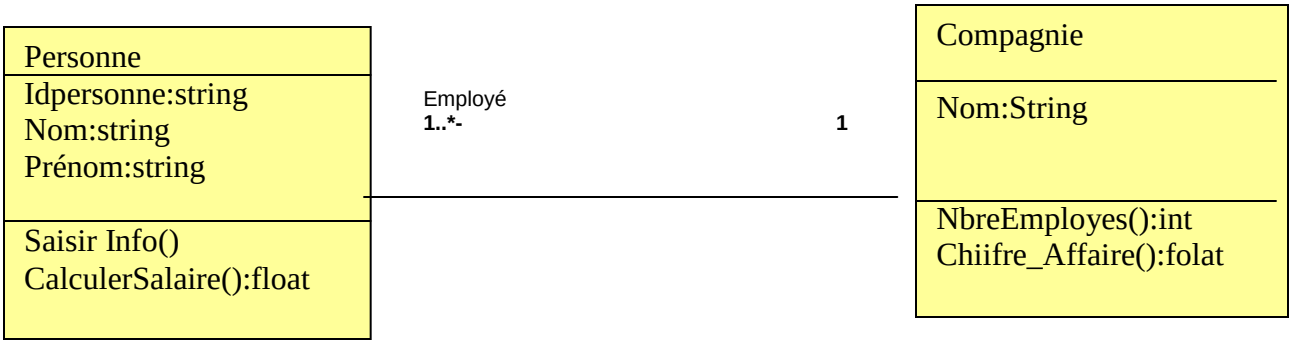


Figure 1: Diagramme de classes (représentation 1)

Dans ce diagramme, le **nom de l'association** est **Employé**, le nom du **rôle** de la classe **Personne** pour l'association est: Employé**.**

Le diagramme se lit comme suit :

Une personne travaille pour une seule entreprise. Dans une entreprise il y 'a une à plusieurs personnes:

Figure 2: Diagramme de classes: Représentation2

Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

Remarque:

Entre la représentation de la figure 1 et la représentation de la figure 2, pour la représentation des associations entre les classes UML opte pour la représentation de la figure1

Notion de cardinalité ou multiplicité

Une cardinalité est le nombre de fois minimum et maximum qu'une instance d'une classe participe à l'association. Ou encore c'est le nombre d'instances de l'association pour une instance de la classe. On l'appelle également la multiplicité

Nous distinguons :

1	un et un seul
0..1	Zéro ou un
m..n	De m à n (entier)
*	Plusieurs
0..*	De zéro à plusieurs
1..*	d'un à plusieurs

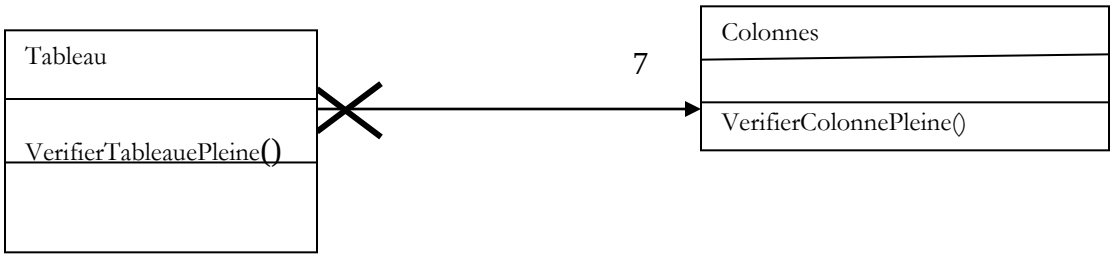
Question 1 : Comment représenter qu'une pièce a une seule forme?

Réponse :

N*otion de rôle* : un rôle spécifie la fonction d'une classe pour une association donnée

Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

Relation navigable:

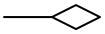


On représente graphiquement la navigabilité par une flèche du côté de la terminaison navigable et on empêche la navigabilité par une croix du côté de la terminaison non navigable

Par exemple, sur la figure la terminaison du côté de la classe *Tableau* n'est pas navigable : cela signifie que les instances de la classe *Colonnes* ne stockent pas de liste d'objets du type *Tableau*. Inversement, la terminaison du côté de la classe *Colonnes* est navigable : chaque objet Tableau contient 7 colonnes.

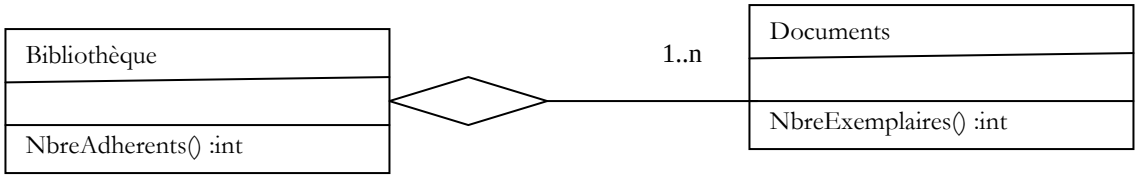
Par défaut, une association est navigable des deux côtés.

Agrégation :

Est un type d'association, elle est symbolisé par 

Elle définit la relation «partie de ».

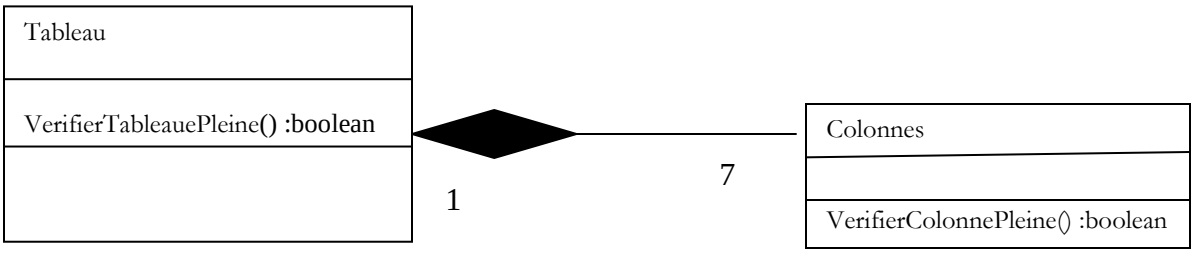
Exemple2 : un document fait partie d'une bibliothèque



Une **composition** est une agrégation forte, elle est représentée par le symbole 

Exemple3 :un tableau est composé de colonnes

Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html



La composition, également appelée agrégation composite, décrit une contenance structurelle entre instances. Ainsi, la destruction de l'objet composite implique la destruction de ses composants

Dans les relations de composition ou agrégation, nous parlons d'objet composite et d'objet composant. Ainsi une instance de la classe **Tableau est dit objet composite** et une instance de la classe **colonne est dit objet composant**

Différences entre une agrégation et une composition :

Pour la relation de composition :

- La destruction de l'objet composite implique la destruction de ses composants.
- La multiplicité du côté composite ne doit pas être supérieure à 1 (*i.e.* 1 ou 0..1).
- La composition est une agrégation non partagée.

Question 2 : comment représenter «un livre est composé de chapitres»?

- **Réponse**

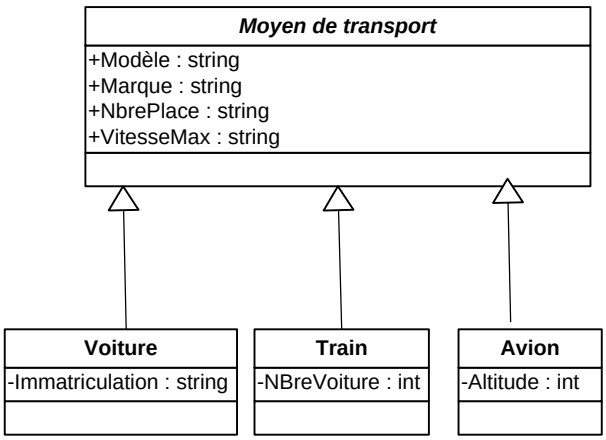
Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

Généralisation, super-classe, sous-classe

La généralisation : définit une relation de classification entre une classe plus générale (super-classe0 et une classe plus spécifique(sous-classe). Il s'agit de prendre des classes existantes (déjà) mise en évidence et de créer de nouvelles classes qui regroupent leurs parties communes, il faut aller du plus spécifiques au plus général, c'est une démarche ascendante.
La spécialisation : il s'agit de sélectionner des classes existantes (déjà) identifiées et d'en dériver de nouvelles classes plus spécialisées en spécifiant simplement les différences. Il s'agit d'une démarche descendante

Une classe abstraite est une classe qui ne s'instancie pas directement, mais qui représente une simple abstraction afin de factoriser les propriétés communes des sous-classes. Elle se note en italique.

Exemple4 : Comment représenter qu'un moyen de transport peut être un avion, un train ou une voiture?



La classe *Moyen de transport* est une classe abstraite.

Voici quelques propriétés de la relation d'héritage :

- La sou- classe possède toutes les caractéristiques des ses super-classes , mais elle ne peut accéder aux caractéristiques privées de cette dernière.
- Toutes les associations de la super-classe s'appliquent aux sous-classes dérivées.

Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

- Une instance d'une classe peut être utilisée partout où une instance de sa super-classe est attendue. Par exemple, toute opération acceptant un objet d'une classe *Moyen de transport* doit accepter un objet de la classe avion.
- Une classe peut avoir plusieurs parents, on parle alors d'héritage multiple

En UML, la relation d'héritage n'est pas propre aux classes. Elle s'applique à d'autres éléments du langage comme, les acteurs ou les cas d'utilisation

Question3 : Comment représenter la situation suivante :

- Des documents sont soit des journaux soit des volumes ou des BD (bandes dessinées)
- Les volumes sont soit des dictionnaires soit des livres
- Les documents ont un titre et un numéro.
- Les volumes ont en plus un auteur, les BD ont en plus le nom du dessinateur, les journaux ont une périodicité.

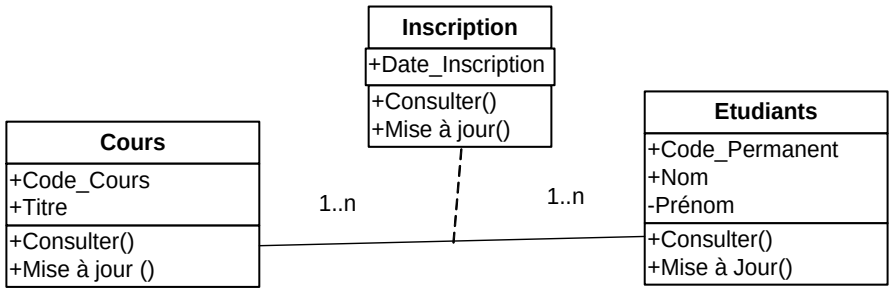
Réponse

Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

Classe d'association

Il s'agit d'une classe qui réalise la navigation entre les instances d'autres classes. Elle sert connecter les classes entre elles

Exemple 5



Parfois, une association doit posséder des propriétés. Par exemple, l'association *Inscription* entre un élève et un cours possède comme propriété la date d'inscription. Cette propriété n'appartient ni aux cours, ni aux étudiants Il s'agit donc bien de propriétés de l'association *Inscription*

Les associations ne pouvant posséder de propriété, il faut introduire un nouveau concept pour modéliser cette situation : celui de *classe-association*.

Une classe-association possède les caractéristiques des associations et des classes : elle se connecte à deux ou plusieurs classes et possède également des attributs et des opérations.

Une classe-association est caractérisée par un trait discontinu entre la classe et l'association qu'elle représente

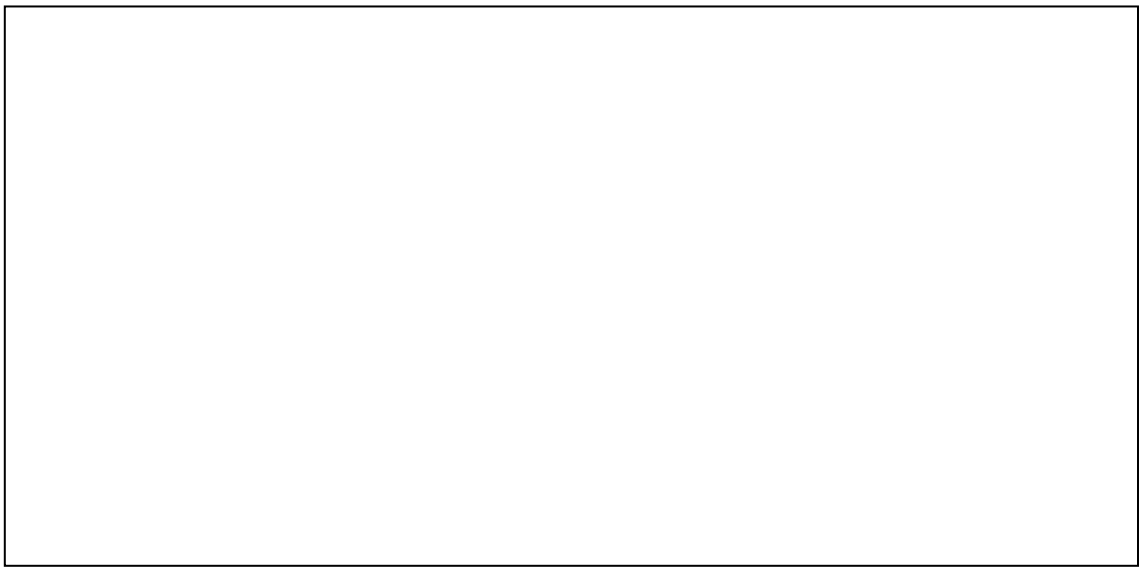
Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

Question4

Reprendre l'énoncé de la question précédente et représenter la situation suivante :

- Seuls les livres sont empruntables.
- Un adhérent peut emprunter plusieurs livres. On gardera la date de prêt et la date prévue pour le retour.

Réponse



Notion de package

Le package est un mécanisme regroupant plusieurs éléments d'UML (peut regrouper des classes, des cas d'utilisation, des interfaces….).

Le diagramme de package sert à :

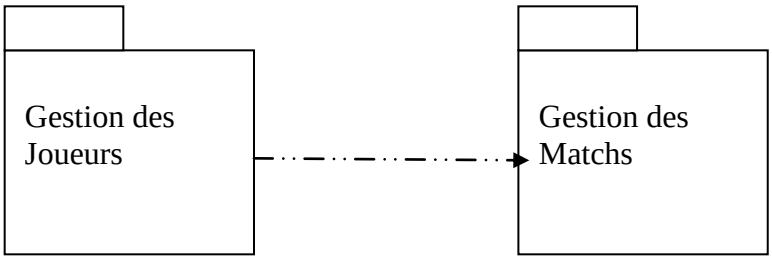
- Avoir une vision globale des différents sous systèmes du système à l'étude.
- Représenter l'architecture globale du système
- Aider à organiser du code (java ou C#) .
- modulariser les diagrammes (UML) les plus complexes.

Le découpage d'un modèle (de classes ou de cas d'utilisation) est une activité délicate. Il faudra regrouper les classes d'un point de vue sémantique c'est-à-dire :

Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

- Les classes d'un même package doivent rendre des services de même nature aux utilisateurs.
- Minimiser les dépendances entre les packages.
- Les dépendances entre packages doivent refléter des relations internes au système.
- Il ne doit pas y avoir de dépendance cyclique entre des packages

Exemple 6:



Chaque package doit être décrit par son diagramme de classes.

Question5

Reprendre l'énoncé des questions3 et 4, puis donner le des packages

Réponse

Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

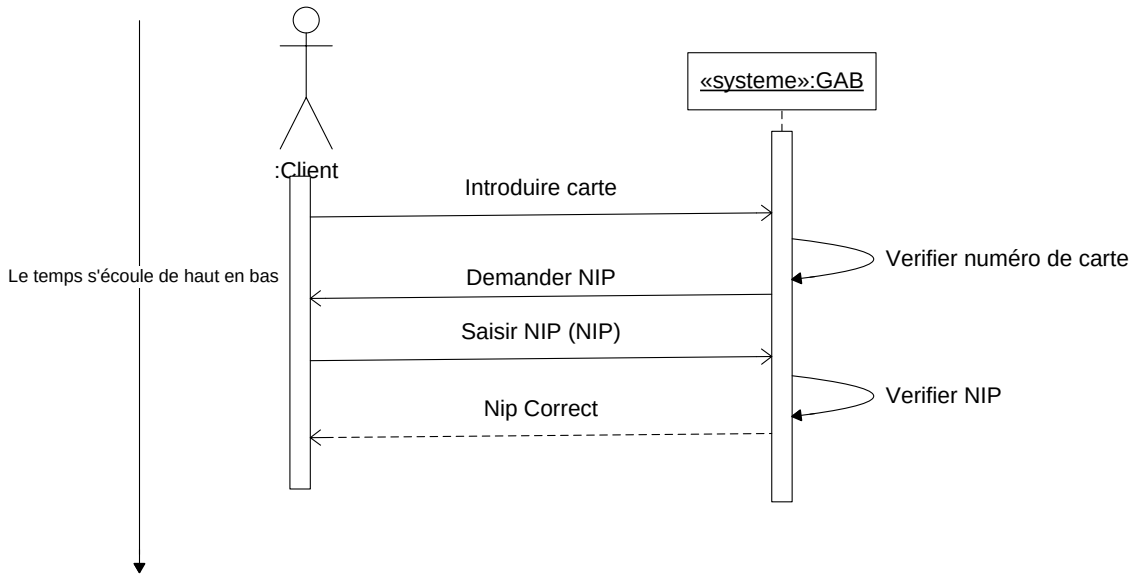
Modélisation dynamique : le diagramme de séquence

Définition et exemple

Un diagramme de séquence montre les interactions entre **les objets**, arrangés en séquence dans le temps. En particulier ils montrent les objets participants dans l'interaction par leur ligne de vie et les messages qu'ils s échangent **ordonnancés dans le temps**. Il ne montre pas les associations entre les objets.
L'ordre d'envoi des messages est déterminé par sa position sur l'axe vertical (l'axe du temps) du diagramme. Le temps s'écoule de haut en bas.

Les diagrammes de séquences permettent entre autre de faire une description graphique d'un cas d'utilisation

Exemple :

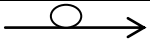
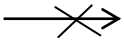


Sur un diagramme de séquence, il est possible de représenter de manière explicite les différentes périodes d'activité d'un objet au moyen d'une bande rectangulaire superposée à la ligne de vie de l'objet.
Un objet peut être activé plusieurs fois au cours de son existence. Un objet non actif n'est pas un objet détruit.

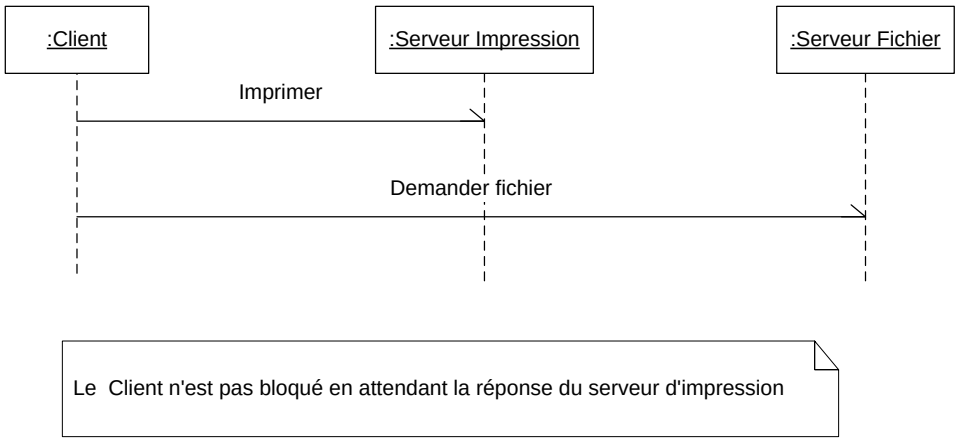
Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

Ligne de vie d'un objet	Activation d'un objet
	

Type de Messages

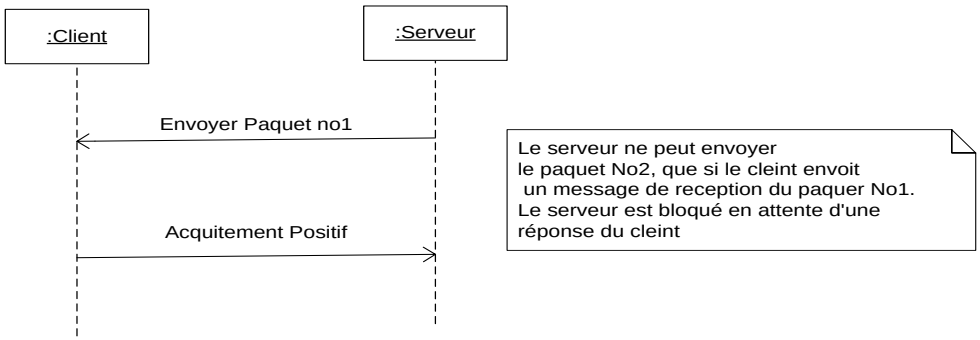
Type de message	Symbole	Explication
Message simple		Message dont il n'y a aucune spécification particulière
Message asynchrone		L'emetteur n'est pas bloquée en attente d'une réponse de la part du récepteur.
Message minuté		L'émetteur du message est bloqué pour un temps t
Message synchrone		L'emetteur du message est bloqué en attente d'une réponse de la part du récepteur.
Message retour		

Exemple de message asynchrone



Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

Exemple de message synchrone



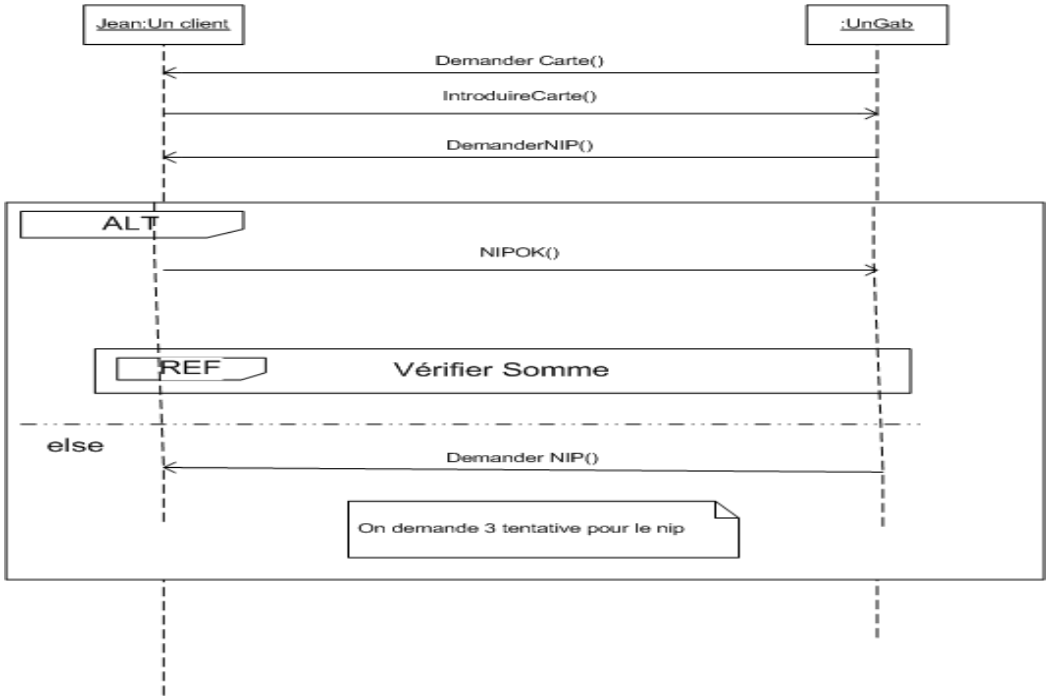
Les diagrammes de séquences selon UML 2.0

Les diagrammes de séquence tels que définis en UML1.x souffraient cependant d'un gros inconvénient. La quantité de diagrammes à réaliser pouvait atteindre un nombre conséquent dès lors que l'on souhaitait décrire avec un peu de détail les différentes branches comportementales d'une fonctionnalité. Le plus coûteux étant de remettre à jour ces diagrammes lors d'un changement au niveau des exigences ou bien du design.
UML2 a donné plus de puissance a ses diagrammes de séquence en introduisant un certain nombre d'opérateurs:

Opérateur alt (alternative)

Désigne un choix, une alternative. Il représente deux comportements possibles : c'est en quelque sorte l'équivalent du **SI...ALORS...SINON** : donc, une seule des deux branches sera réalisée dans un scénario donné. La condition d'exécution d'une des deux branches (l'équivalent du SI) peut être explicite ou implicite.
L'utilisation de l'**opérateur else** permet d'indiquer que la branche est exécutée si la condition du alt est fausse.

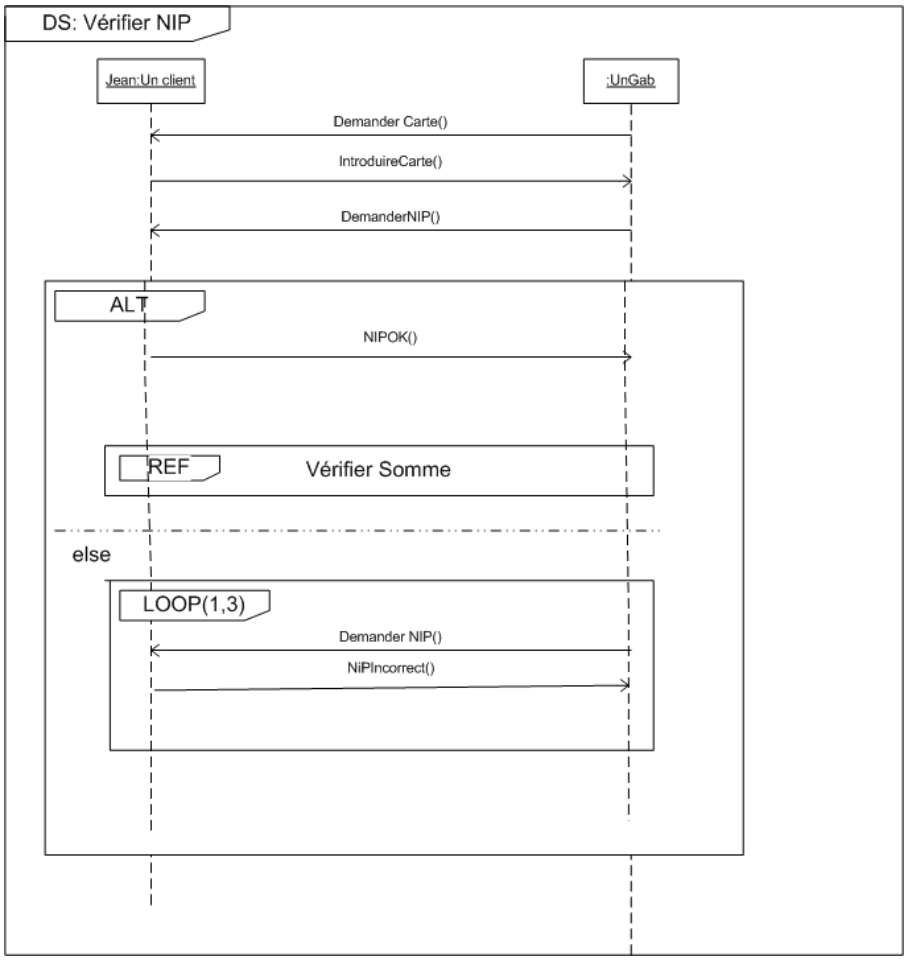
Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html



Opérateur Loop

Cet opérateur est utilisé pour décrire un ensemble d'interaction qui s'exécute **en boucle**. En général, une contrainte appelée **garde** indique le nombre de répétitions (minimum et maximum) ou bien une condition booléenne à respecter.

Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html



Opérateur Opt (option)

Désigne un fragment combiné optionnel comme son nom l'indique : c'est à dire qu'il représente un comportement qui peut se produire... ou pas. Un fragment optionnel est équivalent à un fragment "alt" qui ne posséderait pas d'opérande else (qui n'aurait qu'une seule branche). Un fragment optionnel est donc une sorte de **SI...ALORS**

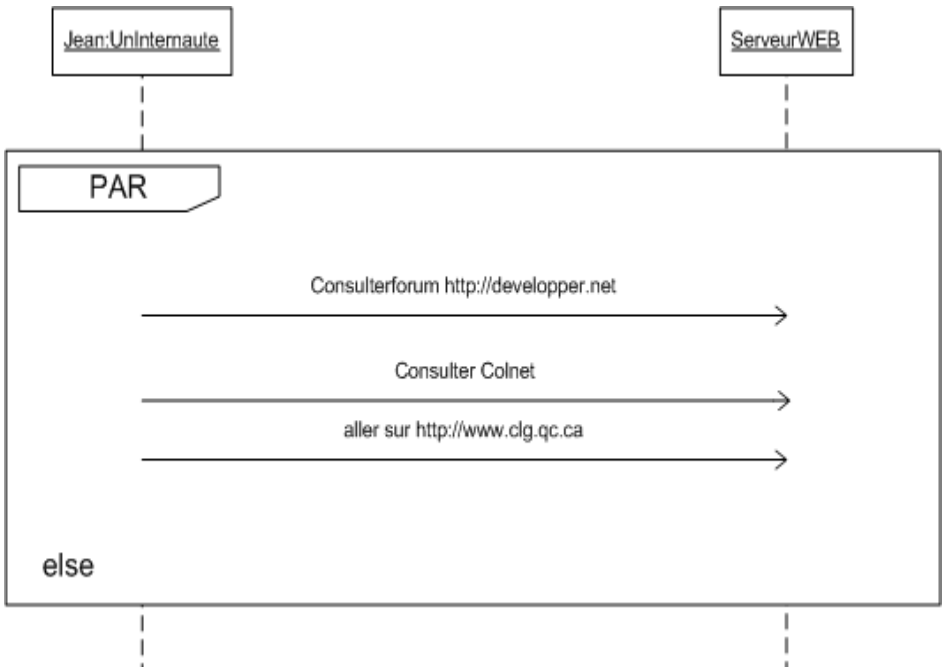
Opérateur Break

Il est utilisé dans les fragments combinés qui représentent des scenarii d'**exception** en quelque sorte. Les interactions de ce fragment seront exécutées à la place des interactions décrites en dessous. Il y a donc une notion d'interruption du flot "normal" des interactions

Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

Opérateur Par (parallel)

Il est utilisé pour représenter des interactions ayant lieu en **parallèle**. Les interactions des différents opérandes (les deux branches de notre opérateur ci-dessous) peuvent donc se mélanger, s'intercaler, dans la mesure où l'ordre imposé dans chaque opérande est respecté.



Critical

Désigne une **section critique**. Une section critique permet d'indiquer que les interactions décrites dans cet opérateur ne peuvent pas être interrompues par d'autres interactions décrites dans le diagramme. On dit que l'opérateur impose un traitement atomique des interactions qu'il contient.

Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

Modélisation dynamique : le diagramme d'état de transition

UML a repris le concept de machine à état finis qui consiste à s'intéresser au cycle de vie d'une instance générique d'une classe particulière au fil de ses interactions avec le reste du monde.
Un diagramme d'état de transition peut être vu comme un ordinogramme pour un objet bien précis du système étudié. Il peut représenter un automate d'états finis.

Un état : représente une situation durant la vie d'un objet pendant laquelle :

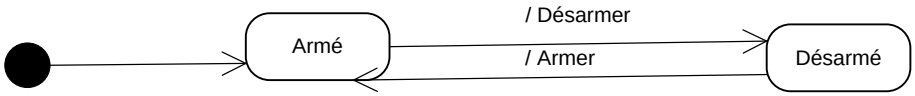
- Il satisfait une certaine condition
- Il exécute une certaine activité
- Ou bien il attend un certain événement

Etat initial et état final :: L'état initial d'un diagramme correspond à la création de l'instance. L'état final correspond à la destruction de l'instance.

Un événement : en UML un événement peut être la réception d'un message, l'appel d'une opération, le passage du temps (mot clé : after) le changement dans la satisfaction d'une condition..
Un événement peut porter des paramètres qui représentent le flot d'information entre objets.

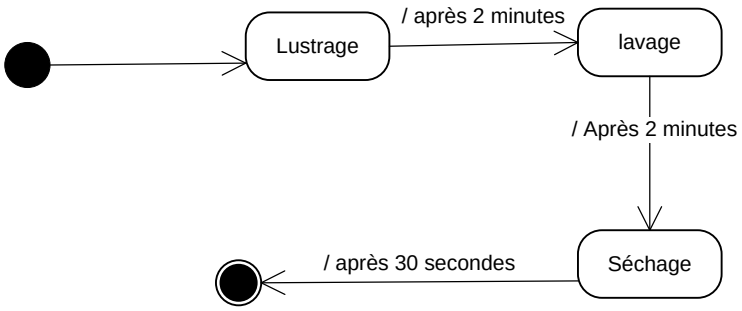
Pour passer d'un état à un autre (une transition), il faudrait un événement.
Une condition ou (condition de garde) est une expression booléenne qui doit être vraie lorsque l'événement arrive pour que la transition soit déclenchée.

Exemple 1:
Diagramme d'état de transition d'un système d'alarme.



Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html

Exemple 2 :

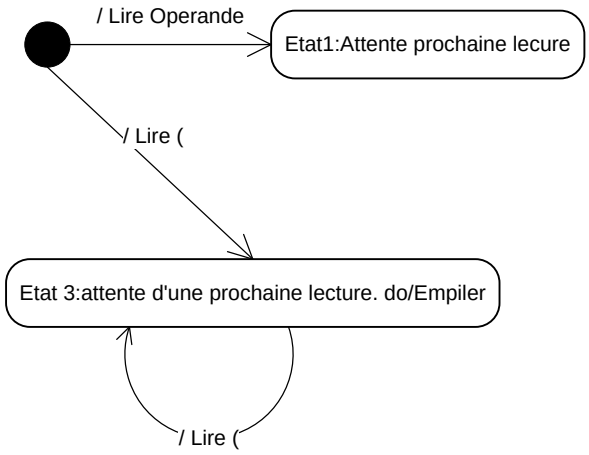


Action dans un état :

Une transition peut spécifier un comportement réalisé par l'objet lorsque la transition est effectuée. Do/action permet d'exécuter une action dans un état :

Exemple

Début du diagramme d'état pour les expressions arithmétiques et logiques.



Sources UML 2 par la pratique Étude de cas et exercices corrigés. Pascal Roques Editions Eyrolles
Le processus unifié de développement logiciel Par : I. Jacobson, G. Booch et J. Rumbaugh, 2000
http://uml.free.fr/cours/i-p20.html