Claude Belleil
Université de Nantes

Le langage UML 2.0
Diagramme de séquence d'objets

1 Introduction

L'objet est une unité formée d'un état, constitué des valeurs instantanées de ses attributs et d'un comportement. Les attributs sont des valeurs qui sont associées aux objets (ex : couleur est un attribut d'un objet voiture). Le comportement regroupe les compétences de l'objet. Ce comportement est décrit par ce que l'on appelle des opérations[1] déclenchées par des stimulations externes appelées messages. Etat, comportement, responsabilité et identité contribuent à mettre l'accent sur la nature intrinsèque d'un objet. Autrement dit, un objet joue un rôle dans un système. Il est assuré au moyen de ses attributs ou de ses opérations.
Le diagramme de classes est l'objectif final de toute démarche de modélisation « objet ». Celui-ci fera l'objet d'une étude détaillée prochainement. Pour le moment, il s'agit de comprendre le cheminement qui conduit des cas d'utilisation au diagramme de classes.
Le choix de présentation qui est fait ici est délibéré. Il consiste à montrer, et si possible à démontrer que la modélisation d'un système à objets s'effectue en observant la nature (statique) et le comportement (dynamique) des objets qui y participent. Comment faire ?
Comme nous l'avons déjà évoqué dans un précédent polycopié[2], un scénario correspond à une séquence d'événements « observables ». Ces événements mettent en œuvre des objets qui interagissent les uns avec les autres à partir d'un stimuli déclencher par un acteur sollicitant un service. Un scénario n'est donc pas une construction intellectuelle. Il s'agit de capter ce qui se passe « ici et maintenant ».

Mettre en évidence des comportements, semble plus en conformité avec le concept d'objet au sens dynamique, car c'est la dynamique qui est l'essence même d'un système.
Un cas d'utilisation est constitué d'un ensemble de scénarios. Un cas d'utilisation a un début et une fin clairement identifiés. Mais entre les deux, plusieurs parcours sont possibles. Ce sont ces parcours qui constituent les différents scénarios possibles.

Les cas d'utilisation se déterminent en observant et en précisant, acteur par acteur, les différents scénarios d'utilisation du système, étape par étape. Une famille de scénarios, regroupée suivant un

[1] Dans le formalisme UML, l'opération désigne la déclaration (la signature) et la méthode, l'implémentation
[2] Les cas d'utilisation

critère fonctionnel, forme un cas d'utilisation

> *« Un scénario est un chemin particulier au travers de la description abstraite et générale fournie par le cas d'utilisation. Les scénarios traversent le cas d'utilisation, en suivant le chemin nominal ainsi que tout chemin alternatif et exceptionnel. L'explosion combinatoire fait qu'il est bien souvent impossible de dérouler tous les scénarios*[3] *»*

Pour décrire un cas d'utilisation,

> *« il est (...) primordial de trouver le bon niveau d'abstraction, c'est-à-dire la quantité de détails à mentionner, et de faire la distinction entre un cas d'utilisation et un scénario. Il n'y a malheureusement pas de réponse toute faite, de sorte que l'appréciation du niveau d'abstraction repose grandement sur l'expérience. Les réponses apportées aux deux interrogations suivantes peuvent néanmoins servir de gabarit : Est-il possible d'exécuter une activité donnée indépendamment des autres, ou faut-il toujours l'enchaîner avec une autre activité ? Deux activités qui s'enchaînent toujours font probablement partie du même cas d'utilisation.*
> *Est-il judicieux de regrouper certaines activités en vue de les décrire, de les tester ou de les modifier ? Si oui, ces activités font sûrement partie du même cas d'utilisation. »*

2 Les diagrammes d'interactions

Ce sont les diagrammes d'interactions qui mettent en évidence le comportement des différents objets et permettent de délimiter leurs responsabilités. Le langage UML 2.0 en propose quatre :

Le diagramme de séquence
Le diagramme de communication
Le diagramme d'interaction composite
Le diagramme de timing

Il est préférable de détecter progressivement les objets d'un système au travers de différents scénarios. Ensuite seulement, on peut mettre en évidence que les scénarios sont des réalisations particulières de Cas d'Utilisation et que le regroupement conceptuel d'objets et leur abstraction vont permettre de définir une ébauche de diagramme de classes.
Il est de loin plus sécurisant de développer des scénarios et ensuite de regrouper les objets dans des diagrammes de classes partiels, qui vont se compléter les uns les autres, plutôt que de tenter d'obtenir le diagramme de classes directement et de façon plus ou moins intuitive.

La modélisation des scénarios correspond à une vision réaliste du système. Par ailleurs, les objets s'échangent des messages, pas les classes ! Autrement dit, d'un point de vue purement formel, il est naturel d'échanger des messages entre objets dans des diagrammes d'interactions, qui sont justement prévus pour cela.

Cette façon de procéder a pour avantage, d'une part de penser le système en terme d'objets (et non plus en terme de classes, encore une fois, le système contient des objets, pas des classes), d'autre part de s'en tenir aux objets réellement pertinents pour les besoins.
Quels sont les objets du système qui collaborent au scénario ? Quels messages échangent-ils ? Les diagrammes d'interactions permettent de dessiner les objets, donc de les visualiser. A partir de là, il est plus facile de se poser des questions telles que : quelle est la demande que tel objet envoie à tel autre pour participer au scénario ? Cela suppose une idée générale des responsabilités de chacun des objets. Rappelons que la responsabilité d'un objet rassemble à la fois :

Qui il est, c'est-à-dire ce qui permet de le distinguer de tous les autres objets, y compris de ceux qui lui sont semblables
Ce qu'il sait, c'est-à-dire l'ensemble de ses attributs instanciés, par exemple : couleur = « bleu »,

[3] Modélisation objet avec UML, Pierre-Alain Muller, Nathalie Gaertner, Éditions Eyrolles, 2e édition, mars 2000

poids = 275 g, etc…
Ce qu'il sait faire, c'est-à-dire l'ensemble de ses opérations, par exemple : Créer(), Détruire(), Décrocher(), Composer(numéro), etc….

Les diagrammes de classes représentent alors la synthèse - nécessaire – de ces différentes interactions[4].

Grâce à une approche itérative à partir des scénarios, dont le résultat est un ensemble de diagrammes d'interactions, l'analyse et la conception objet ne relèvent plus d'une approche guidée par l'intuition, avec tous les risques que cela comporte. Le système se construit progressivement, scénarios par scénarios et cas d'utilisation après cas d'utilisation. En tenant compte des responsabilités des différents objets qui doivent être à la fois limitées et cohérente entre elles on doit obtenir une conception qui aura su éviter les pièges les plus grossiers.
Les interactions constituent une clé du rapprochement informatique / systémique. Plus généralement, ce concept d'interaction, de "relation dynamique" est inhérent à toute science ou toute technique. Les interactions représentent le cœur du système, son essence même en terme de fonctionnement. Une utilisation régulière des diagrammes d'interaction est signe d'une véritable approche objet.

3 Le Diagramme de séquence d'objets

Un diagramme de séquence d'objets permet de décrire les interactions entre objets en insistant sur l'ordonnancement temporel des messages. Il fait partie des diagrammes d'interaction dans lesquels on retrouve systématiquement des objets, des liens et des messages. Le diagramme de séquence d'objets est utilisé pour modéliser des flux de contrôle ordonnés selon le temps. Il ne décrit toutefois pas le contexte des objets.
Les principales informations contenues dans un diagramme de séquence d'objets sont les messages échangés entre les lignes de vie, présentés dans un ordre chronologique. Ainsi, contrairement au diagramme de communication, le temps y est représenté explicitement par une dimension (la dimension verticale) et s'écoule de haut en bas. Ce diagramme, tel qu'il était défini en UML1.4 présentait un inconvénient majeur.
En effet, 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 de la conception. Avec la version UML2.0 ces diagrammes ont gagné en puissance de représentation grâce à de nouvelles constructions permettant de réduire significativement la quantité de diagramme à réaliser.

Les règles de cohérence se rapportant aux diagrammes de séquence peuvent être classées en plusieurs familles : l'une concerne les messages d'appel synchrone d'une opération, une autre les messages asynchrones, une autre les traces d'une interaction, et enfin les éléments contenus dans un diagramme de séquence.

3.1 Objet (UML 1.4) ou Ligne de vie (UML 2.0)

Dans un diagramme de séquence, un objet est représenté par une boîte contenant le nom de l'objet suivi du nom de la classe à laquelle il appartient. Une ligne verticale en pointillés représente la ligne de vie de l'objet.

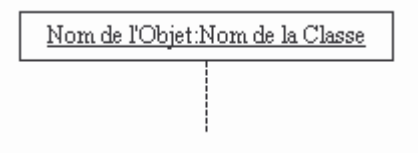

Figure 1 : Représentation d'un objet avec sa ligne de vie

[4] Ils feront l'objet d'un document spécifique

Figure 2 : Un objet sans classe, une classe sans objet

Avec UML 2.0, l'objet a pris le nom de « Ligne de vie». Une ligne de vie se représente par un rectangle, auquel est accroché une ligne verticale pointillée, contenant une étiquette dont la syntaxe est :

[<nom_du_rôle>] : [<Nom_du_type>]

Au moins un des deux noms doit être spécifié dans l'étiquette, les deux points (:) sont, quand à eux, obligatoires.

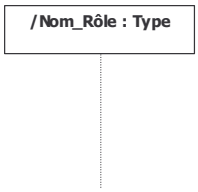

Figure 3: Représentation d'une ligne de vie

Un diagramme de séquence est défini par :

un axe vertical qui représente le temps. Par convention, le temps s'écoule de haut en bas et de gauche à droite. Dans la modélisation des systèmes en temps réel, l'axe du temps peut être gradué d'une façon très fine pour exprimer les contraintes temporelles.

un axe horizontal sur lequel sont représentés les objets. Un objet est indiqué par un rectangle, auquel on attache une ligne en pointillé qui représente sa ligne de vie. Cette ligne de vie sert de point de départ ou d'arrivée à des messages échangés entre objets.

Les messages sont représentés horizontalement par une ligne terminée par une flèche, orientée de l'émetteur du message vers le destinataire. La représentation d'un délai de propagation significatif dans la transmission d'un message est indiquée par le fait que la flèche correspondant au message est oblique.

Le diagramme de séquences ressemble au suivi d'événements de la méthode OMT. Chaque objet impliqué dans une collaboration est représenté avec un axe des temps, généralement orienté du haut vers le bas.

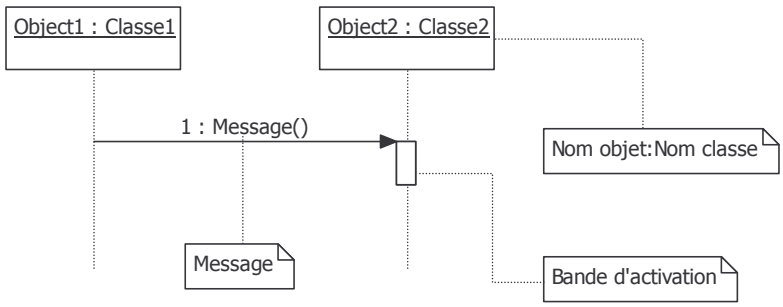

Figure 4: Exemple d'interaction entre deux objets

3.2 Bandes d'activation

Les périodes durant lesquelles l'objet est actif sont représentées sur l'axe des temps par un rectangle, et portent le nom de périodes d'activation. Attention, un objet peut être actif plusieurs fois au cours de son existence. Cette période correspond simplement à la durée pendant laquelle l'objet "travaille", pendant laquelle le code de cet objet est exécuté par le processeur si l'on est dans le cas d'un programme informatique.

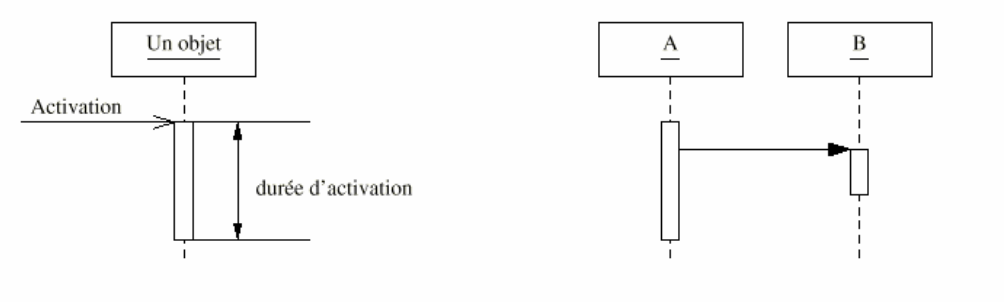

Figure 5 : Exemples de bandes d'activation

Un même message peut déclencher des opérations différentes selon l'objet qui le reçoit. C'est la notion de polymorphisme que nous approfondirons dans le polycopié sur les classes. Concernant la réponse à un message, un événement peut être associé à la réception d'un message. Si la réponse est instantanée il n'y a pas de message associé à la réponse. Il existe différents[5] types de messages.

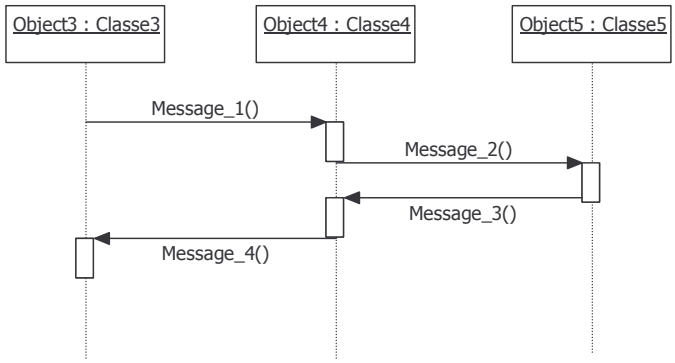

Figure 6: Des objets actifs plusieurs fois

Les objets s'échangent des messages. L'axe des temps donne l'information sur la chronologie, les délais, la simultanéité de l'envoi des messages. Les notations concernant les messages sont identiques à celles pratiquées dans le diagramme de collaboration.
Le diagramme de collaboration et le diagramme de séquence représentent les mêmes informations, mais considérées sous un angle de vue différent. Dans le premier cas on met l'accent sur la structure spatiale du modèle, dans le second cas on se focalise sur la dimension temporelle.

3.3 Notion de message

Un message est un mécanisme par lequel un objet communique avec un autre. Un message est supposé provoquer l'exécution d'une opération par l'objet destinataire. Il faut distinguer le message et

[5] Ces différents types ne sont pas tous implémentés dans les AGL UML

l'opération. Le message peut être assimilé à un appel, un stimulus extérieur qui provoque l'exécution de l'opération.
Uml distingue **l'opération** (équivalent de la signature d'une procédure ou d'une fonction) de la **méthode** qui correspond au code qui sera exécuté. Par un glissement du sens lié à la pratique des certains langages de programmation (Java), dans de nombreux ouvrages ou sites Internet, le terme de méthode est utilisé pour faire référence à l'opération Un message définit une communication particulière entre des lignes de vie. Plusieurs types de messages existent, les plus commun sont :

– l'envoi d'un signal ;
– l'invocation d'une opération ;
– la création ou la destruction d'une instance.

3.4 Les différents types de messages

3.4.1 Message de type «send » (envoi)

Une interruption ou un évènement sont de bons exemples de signaux. Ils n'attendent pas de réponse et ne bloquent pas l'émetteur qui ne sait pas si le message arrivera à destination, le cas échéant quand il arrivera et s'il serra traité par le destinataire. Un signal est, par définition, un message asynchrone.

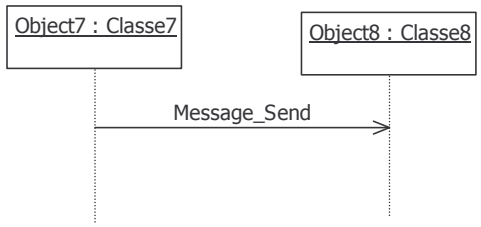

Figure 7: Message de type "Send" (envoi simple)

3.4.2 Message synchrone et asynchrone

Asynchrone

Graphiquement, un message asynchrone se représente par une flèche en traits pleins et à l'extrémité ouverte partant de la ligne de vie d'un objet expéditeur et allant vers celle de l'objet cible
L'invocation d'une opération est le type de message le plus utilisé en programmation objet. L'invocation peut être asynchrone ou synchrone. Dans la pratique, la plupart des invocations sont synchrones, l'émetteur reste alors bloqué le temps que dure l'invocation de l'opération.

```
================================================================================
«Message asynchrone[6]»

Règles de cohérence extraites du document de l'OMG « UML2SuperStructure[7]»

• Aucun message de retour ne doit être la conséquence d'un message
asynchrone. [Nouvelle règle]
• Un message d'envoi de signal asynchrone ne doit déclencher aucune
occurrence d'exécution. [Nouvelle règle]
================================================================================
```

[6] Les paragraphes avec ce style (Courier New 8) sont des traductions d'extraits du document officiel de définition du langage UML 2 dont vous trouverez la référence web ci-dessous.
[7] http://www.omg.org/technology/documents/formal/uml.htm

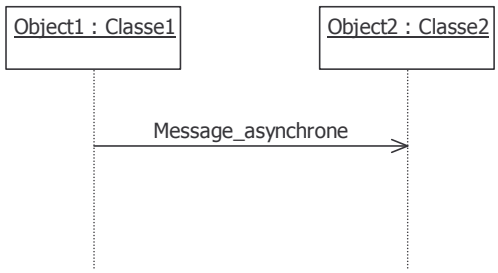

Figure 8: Message asynchrone

3.4.3 Synchrone

Graphiquement, un message synchrone se représente par une flèche en traits pleins et à l'extrémité pleine partant de la ligne de vie d'un objet expéditeur et allant vers celle de l'objet cible. Ce message peut être suivi d'une réponse qui se représente par une flèche en pointillé. Un appel d'opération synchrone se fait via l'envoi d'un message synchrone étiqueté par le nom d'une opération.

```
================================================================================
«Message synchrone»

Règles de cohérence extraites du document de l'OMG « UML2SuperStructure»

• Dans le cas d'un appel synchrone d'une opération, un message déclenche une
occurrence d'exécution sur l'objet cible. [Nouvelle règle]
• La fin d'une occurrence d'exécution doit engendrer un message de retour.
[Nouvelle règle]
• Le message de retour à la fin d'une occurrence d'exécution doit avoir pour
source la ligne de vie qui supporte l'occurrence d'exécution et pour cible
la ligne de vie qui est responsable de l'envoi du message déclencheur de
l'occurrence d'exécution.
[Nouvelle règle]
• Un message de retour est toujours le résultat de la fin d'une occurrence
d'exécution. [Nouvelle règle]
• Un appel synchrone d'opération se fait entre objets qui appartiennent à la
même unité d'exécution et qui ne doivent donc pas être des objets actifs
[Nouvelle règle]
================================================================================
```

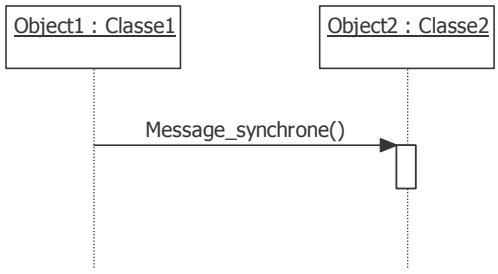

Figure 9: Message synchrone

3.4.4 Les messages d'appel d'opération

C'est le type message le plus souvent utilisé. Selon les AGL[8], il est référencé comme :

L'invocation d'une opération (Objecteering)
Un type « CALL » (Magic Draw, StarUML)

[8] Atelier de Génie Logiciel

Une “ActionTypeCall” (Visual Paradigm)

Dans chacun des ces cas, le typage du message en tant qu’appel d’une opération n’est réalisable qu’à partir du moment où l’objet considéré a préalablement été déclaré comme appartenant à une classe[9].

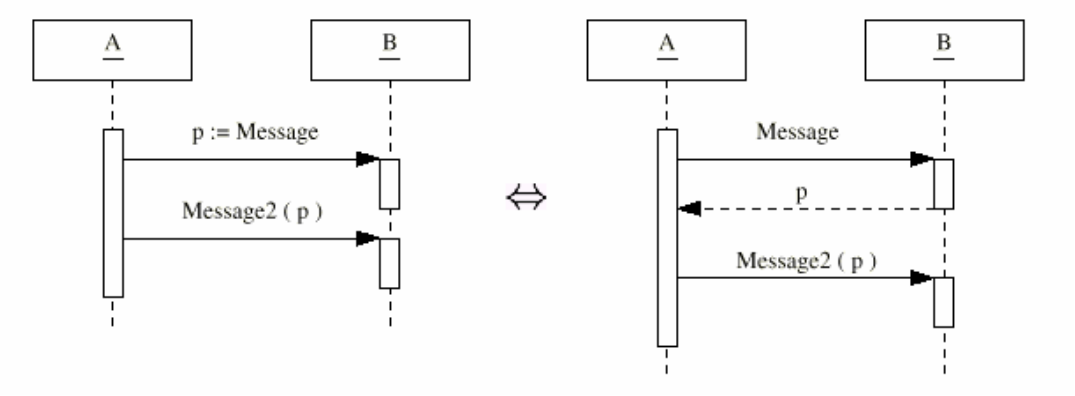

Figure 10 : Différents formalismes des messages d'appel d'opérations

3.4.5 Messages avec paramètre(s)

Les messages d’appel d’opération peuvent transporter des valeurs au travers de paramètres. On distingue :

- les paramètres d’appel qui peuvent être nombreux et qui sont placés dans les parenthèses ouvrantes et fermantes de l’opération. Dans l’exemple ci-dessus : Message2(p)

- le paramètre résultat pour lequel il existe deux formalismes de représentation, dans l’exemple de la figure ci-dessous :

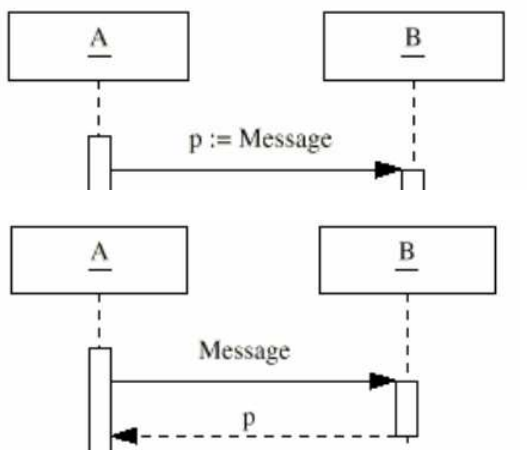

Figure 11 : Les deux formalismes de représentation du paramètre de retour

3.4.6 Les messages de création et de destruction

Ils correspondent aux deux opérations de base auxquelles ont accès tous les objets. Mais si tous les objets d’un système peuvent êtres créés et détruits[10], ils ne se positionnent pas de la même façon vis-à-vis de ces deux opérations fondamentales.

[9] Cette nécessité sera mise en évidence lors de l’étude du diagramme de classes
[10] Plus généralement historisés

C'est, en général, ce qui permet de distinguer les objets réputés « permanents » des objets « temporaires ».
Bien sûr, à l'échelle de l'histoire de l'univers, tous les objets peuvent être considérés comme temporaires ! Dès lors, comment distinguer, les uns des autres ?
Ici, comme dans d'autres circonstances, ce sont les cas d'utilisation qui nous apporteront une réponse.

Par exemple, dans le contexte d'un échange téléphonique entre deux personnes, les téléphones, l'ensemble des objets composants le Réseau Téléphonique sont perçus par les utilisateurs (les acteurs) comme étant des **objets permanents**. En revanche, l'appel téléphonique lui-même, si il est matérialisé par un objet, celui-ci aura été créé à l'occasion de cet échange téléphonique et il sera détruit dès que l'un des deux interlocuteurs aura raccroché. Au regard du cas d'utilisation cet objet aura été perçu comme **temporaire**.

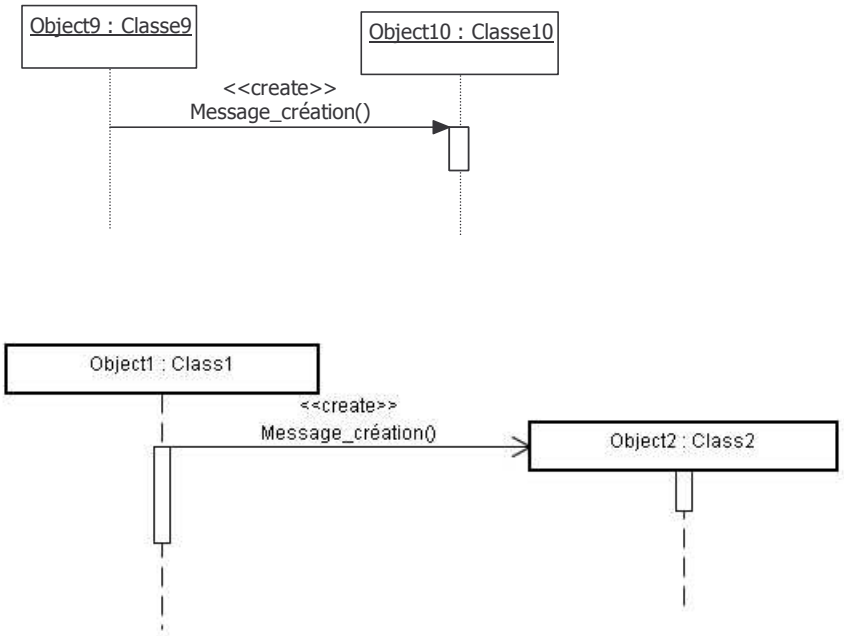

Figure 12: Message de création d'objet (deux formalismes)

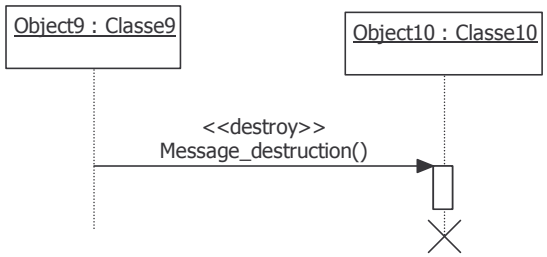

Figure 13: Message de destruction d'objet

3.4.7 Les messages réflexifs

Ils représentent des messages qu'un objet s'envoie à lui-même. En général, cela est représentatif que l'on se trouve confronté à un objet composite. Dès lors deux positions peuvent s'envisager :

- la nature de l'étude conduit à ignorer la composition exacte de cet objet. C'est un choix délibéré de granularité qui amène l'analyste a laissé à cet objet le statut de « boîte noire », l'étude du système se concentrant sur d'autres aspects.
- à l'inverse, les messages réflexifs mettent en évidence l'existence d'objets composants que l'on n'a pas mis en évidence et qui apparaissent comme indispensables à l'étude. Ces

messages mettent alors en relief une mauvaise granularité de l'analyse qui devra donc être corrigée.

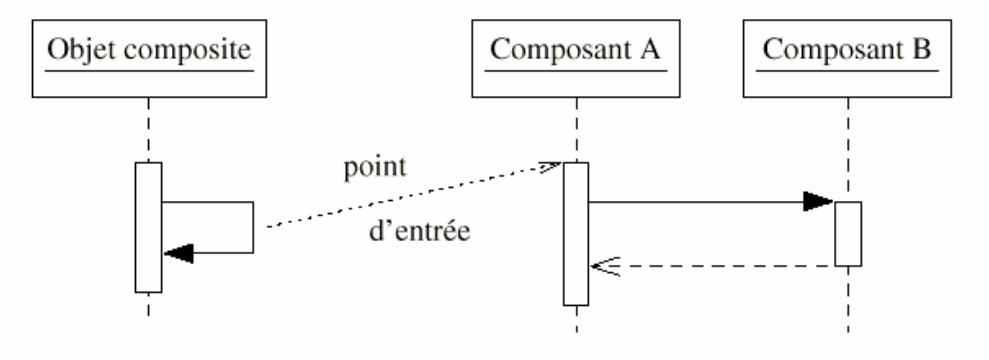

Figure 14 : Un message réflexif est représentatif de l'activité d'un objet composite

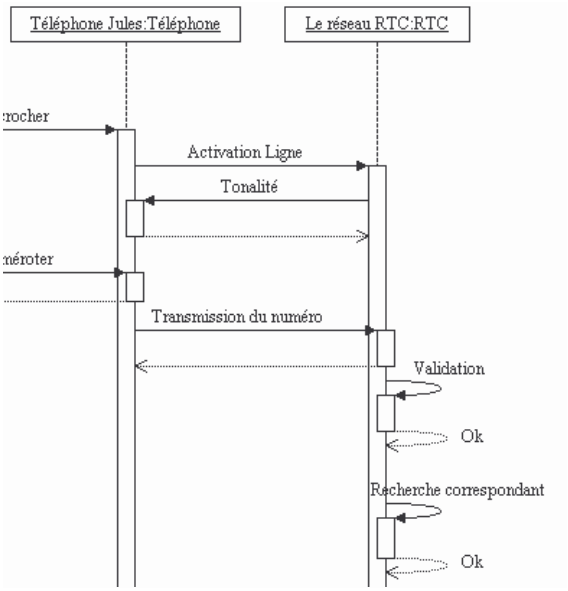

Figure 15 : Exemple de messages réflexifs sur l'objet composite « Le réseau »

3.4.8 Les messages conditionnels (Uml 1.4)

Les envois de messages peuvent être conditionnels. L'évaluation d'un prédicat déclenche l'activation d'une opération sur un objet ou d'une autre opération sur un autre.
Les messages conditionnels doivent être utilisés avec prudence. En effet, un diagramme de séquence d'objets sert à illustrer un scénario. Or il est assez rare qu'un scénario présente des cheminements alternatifs. En revanche, une famille de scénarios, pourra comporter des alternatives. Mais en pareil cas, il faut être certain que l'on ne glisse pas insensiblement d'une représentation des objets du système vers une représentation plus globale du comportement des classes !
Avec Uml 2.0, ce formalisme a été abandonné au profit d'une représentation des alternatives ou des structures de choix multiples par les « fragment d'interaction combinés [11]»

[11] Voir plus loin dans le document

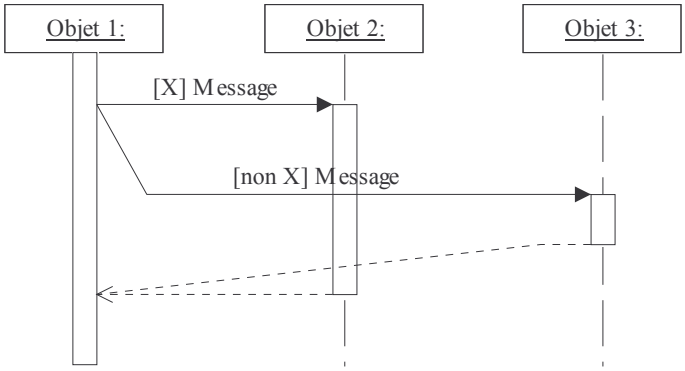

Figure 16 : Un envoi de message conditionnel

3.4.9 Les messages récursifs

On peut représenter des messages récursifs, en dédoublant la bande d'activation de l'objet concerné. Les structures de contrôles telles que les boucles ou les conditions peuvent aussi être représentées. Par exemple, pour représenter de manière graphique une exécution conditionnelle d'un message, on peut documenter un diagramme de séquence avec du pseudo-code et représenter des bandes d'activation conditionnelles.

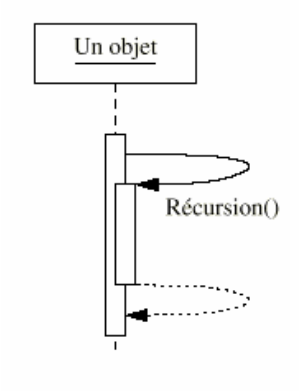

Figure 17 : Exemple de message récursif

4 Fragments et fragments d'interaction combinés

4.1 Fragments

Un fragment permet de définir un bloc d'interactions entre objets pouvant être utilisé dans le cadre d'un autre diagramme de séquence d'objets ou d'un diagramme de structure composite. Il est référencé dans le cartouche figurant en haut à gauche, par « sd » (sequence diagram) et le nom correspondant à sa sémantique. Théoriquement (voir les capacités des différents AGL), il peut apparaître dans un diagramme de séquence de niveau supérieur pour figurer comme une sous séquence d'interaction.

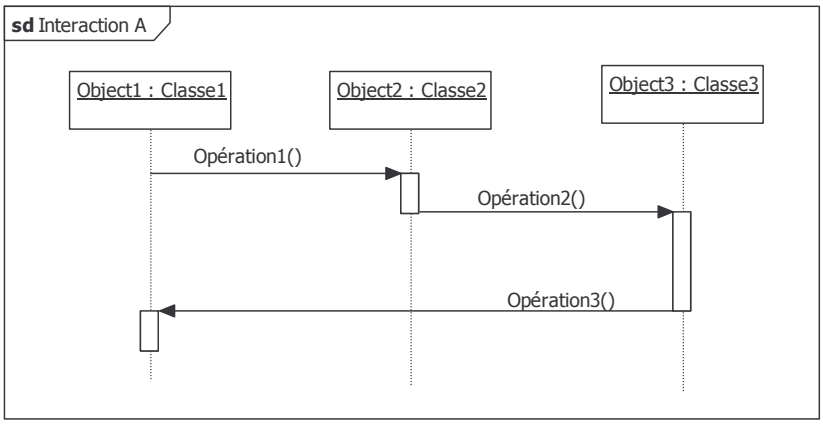

Figure 18: Un fragment d'interaction

Il est possible de faire référence à une interaction (on appelle cela une utilisation d'interaction) dans la définition d'une autre interaction. Comme pour toute référence modulaire, cela permet la réutilisation d'une définition dans de nombreux contextes différents.
Lorsqu'une utilisation d'interaction s'exécute, elle produit le même effet que l'exécution d'une interaction référencée avec la substitution des arguments fournie dans le cadre de l'utilisation de l'interaction.

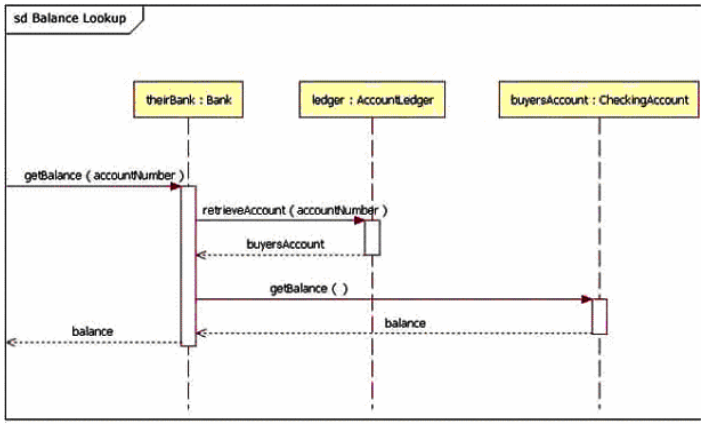

Figure 19: Fragment "contrôle de la balance d'un compte" (balance lookup)

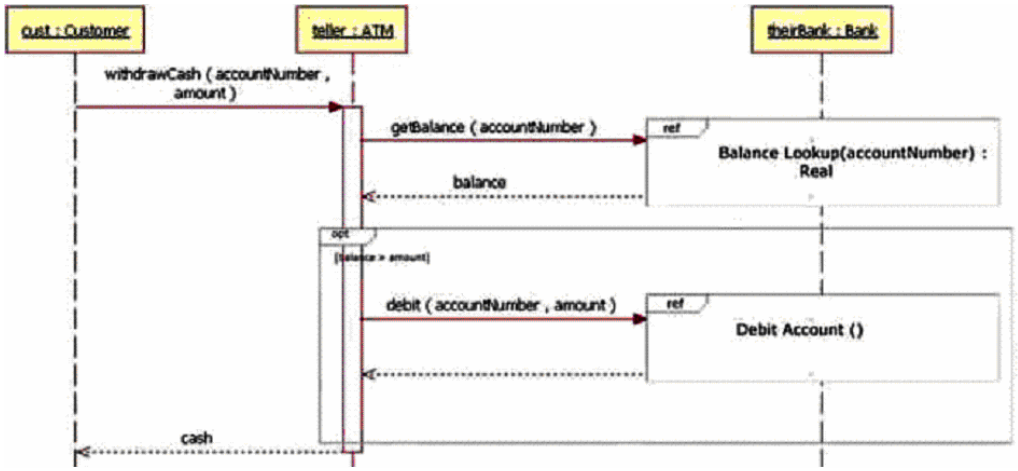

Figure 20: Référence au fragment "balance lookup" dans un autre diagramme de séquence

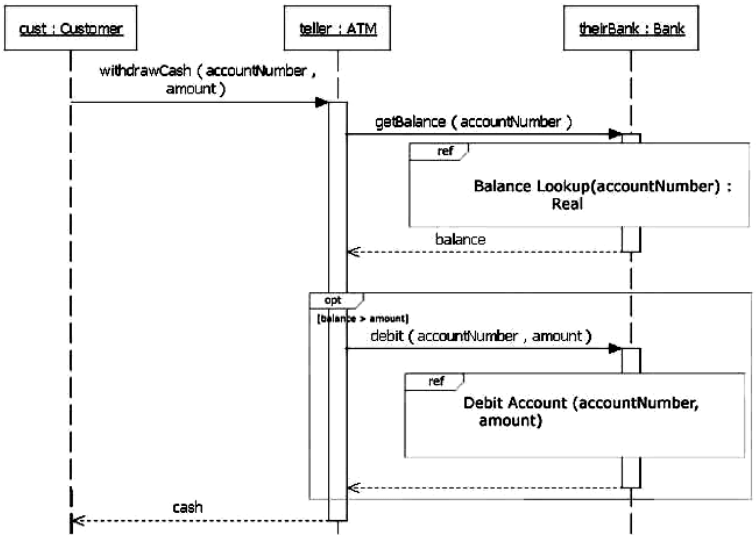

Figure 21: Diagramme de séquence utilisant la référence à deux fragments (ref)

4.2 Fragments d'interactions combinés

Un fragment combiné représente des articulations d'interactions présentées dans le cadre d'un opérateur. Il est défini par un opérateur et des opérandes. L'opérateur conditionne la signification du fragment combiné. Il existe 12 d'opérateurs définis dans la notation UML 2.0. Les fragments combinés permettent de décrire des diagrammes de séquence de manière compacte. Les fragments combinés peuvent faire intervenir l'ensemble des entités participant au scénario ou juste un sous-ensemble.
Un fragment combiné se représente de la même façon qu'une interaction. Il est représenté un rectangle dont le coin supérieur gauche contient un cartouche. Dans le cartouche figure le type de la combinaison, appelé opérateur d'interaction. Les opérandes d'un opérateur d'interaction sont séparés par une ligne pointillée. Les conditions de choix des opérandes sont données par des expressions booléennes entre crochets ([ ]).

La liste suivante regroupe les opérateurs d'interaction par fonctions :

– les opérateurs de choix et de boucle : **alternative**, **option**, **break** et **loop** ;
– les opérateurs contrôlant l'envoi en parallèle de messages : **parallel** et **critical region** ;
– les opérateurs contrôlant l'envoi de messages : **ignore**, **consider**, **assertion** et **negative** ;
– les opérateurs fixant l'ordre d'envoi des messages : **weak sequencing**, **strict sequencing**.

Quelques exemples de ces interactions dans sont illustrés ci-dessous.

4.2.1 Opérateurs **alt** et **opt**

L'opérateur **alternative**, ou **alt**, est un opérateur conditionnel possédant plusieurs opérandes. C'est un peu l'équivalent d'une exécution à choix multiple (condition switch en C++). Chaque opérande détient une condition de garde. L'absence de condition de garde implique une condition vraie (true). La condition `else` est vraie si aucune autre condition n'est vraie. Exactement un opérande dont la condition est vraie est exécuté. Si plusieurs opérandes prennent la valeur vraie, le choix est non déterministe.
L'opérateur **option**, ou **opt**, comporte un opérande et une condition de garde associée. Le sous fragment s'exécute si la condition de garde est vraie et ne s'exécute pas dans le cas contraire.

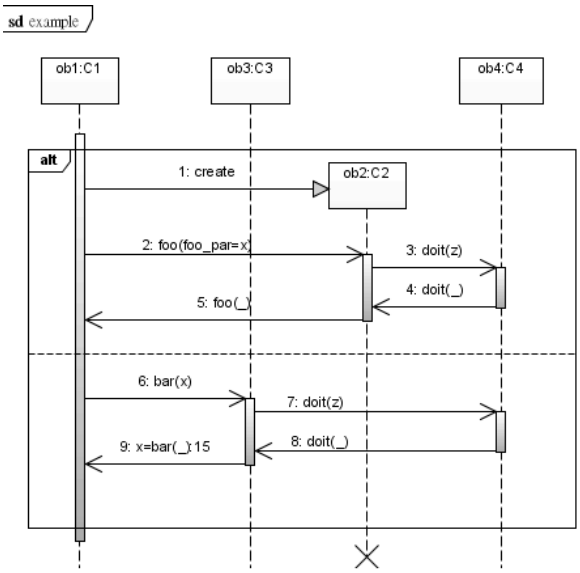

Figure 22: Exemple d'alternative

4.2.2 Opérateur **loop**

Un fragment combiné de type **loop** (cf. figure 7.12) possède un sous-fragment et spécifie un compte minimum et maximum (boucle) ainsi qu'une condition de garde. La syntaxe de la boucle est la suivante :

```
loop[ '('<mini> [ ',' <maxit> ] ')' ]
```

La condition de garde est placée entre crochets sur la ligne de vie. La boucle est répétée au moins `mini` fois avant qu'une éventuelle condition de garde booléenne ne soit testée. Tant que la condition est vraie, la boucle continue, au plus `maxi` fois.

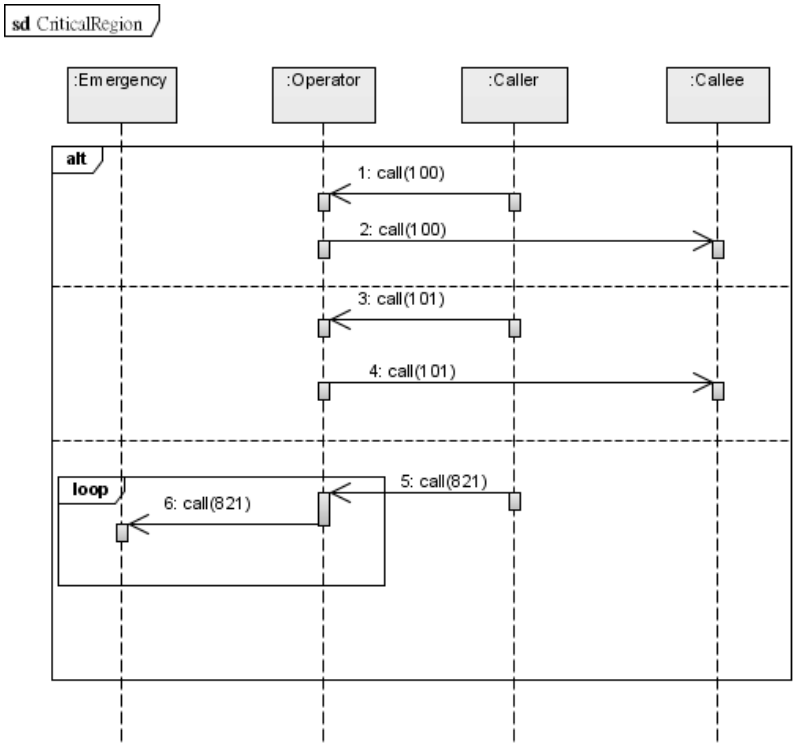

Figure 23: Emboîtement d'une boucle dans une des branches d'une alternative

4.2.3 Opérateur **par**

Un fragments combiné de type **parallel**, ou **par**, possède au moins deux sous-fragments exécutés simultanément. La concurrence est logique et n'est pas nécessairement physique : les exécutions concurrentes peuvent s'entrelacer sur un même chemin d'exécution dans la pratique.

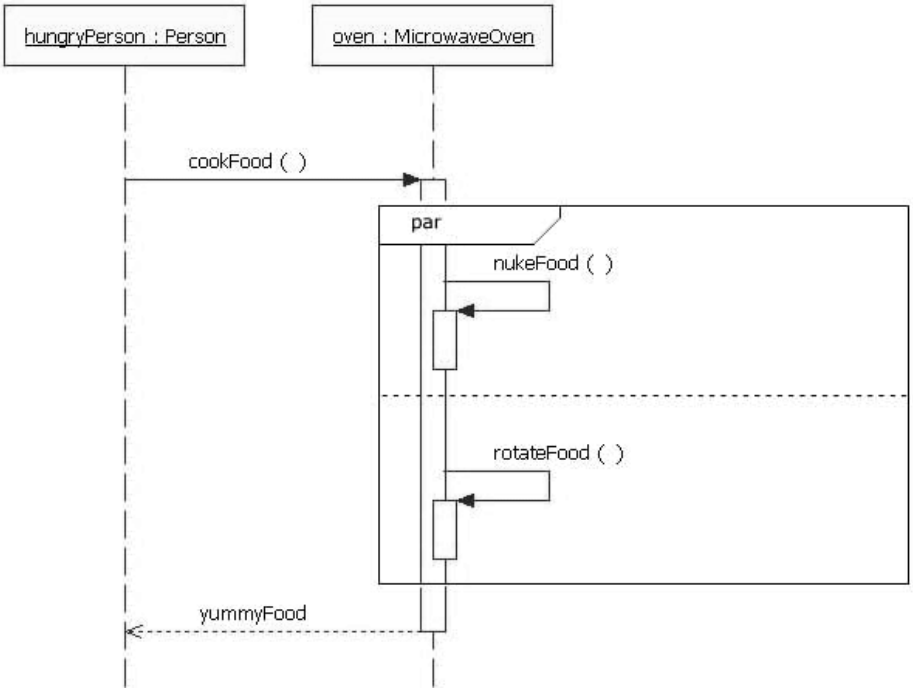

Figure 24: Un fragment combiné d'interactions parallèles

4.2.4 Opérateur **stric**

Un fragments combiné de type **strict sequencing**, ou **strict**, possède au moins deux sous-fragments. Ceux-ci s'exécutent selon leur ordre d'apparition au sein du fragment combiné. Ce fragment combiné est utile surtout lorsque deux parties d'un diagramme n'ont pas de ligne de vie en commun.

4.3 Interactions entre acteurs et objets

Comme nous l'avons mis en évidence avec les cas d'utilisation, les acteurs d'un système sont des entités externes à ce système qui interagissent avec lui. Quand on dit « qui interagissent », cela signifie d'une part :
qu'ils produisent des évènements comme, introduire une carte dans un lecteur, saisir une date de naissance, cliquer sur un bouton, introduire une pièce de monnaie… etc.
d'autre part, qu'ils reçoivent des informations de la part du système comme, un message affiché sur un écran de contrôle, le rafraîchissement de pages Html dans un navigateur, le signal sonore d'un lecteur de carte magnétique… etc.
Les acteurs sont donc extérieur au système et dialoguent avec lui. On peut considérer qu'ils sont d'une part, de simples émetteurs de stimulis sur le système, d'autre part des observateurs attentifs aux réactions du système dont ils attendent en général des consignes claires d'utilisation.

Dès lors, le système (les objets le constituant) ne peut pas dialoguer directement avec l'acteur en lui envoyant des messages du type « appel d'opération ». L'acteur étant extérieur au système, il n'est muni d'aucune opération. En revanche il doit posséder les compétences lui permettant d'utiliser le système au mieux de ses intérêts.

Or, dans les ouvrages Uml, il n'est pas rare de trouver des diagrammes de séquence d'objets tels que celui qui est présenté ci-dessous. On constate que l'objet « distributeur de billet » envoie directement des message de type « appel d'opération » à l'acteur Djamel. La plupart du temps, il s'agit d'un raccourci illustrant à un très haut niveau de granularité les échanges entre l'acteur et le système.

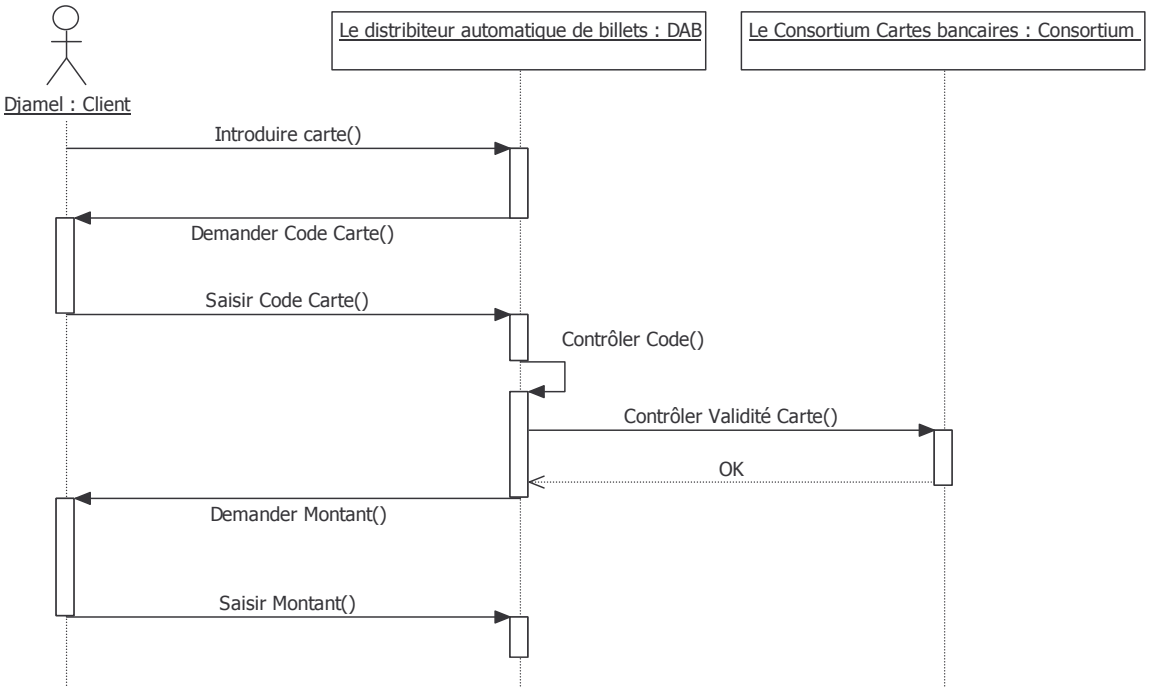

Figure 25: Interaction entre un acteur (client) et un objet (DAB) sans interface

Si on veut réellement représenter ces interactions, on préfèrera une modélisation faisant apparaître une catégorie spéciale de classes : les classes d'analyse parmi lesquelles on trouve le stéréotype « boundary » qui est utilisé pour figurer tous les éléments du Distributeur Automatique de Billets permettant de réaliser le dialogue entre l'acteur et le système.
Par exemple dans le diagramme suivant, la classe « boundary » Interface Utilisateur représente l'ensemble des objets : écran, lecteur de carte, pavé de touches, distributeur de tickets, distributeur de billets.

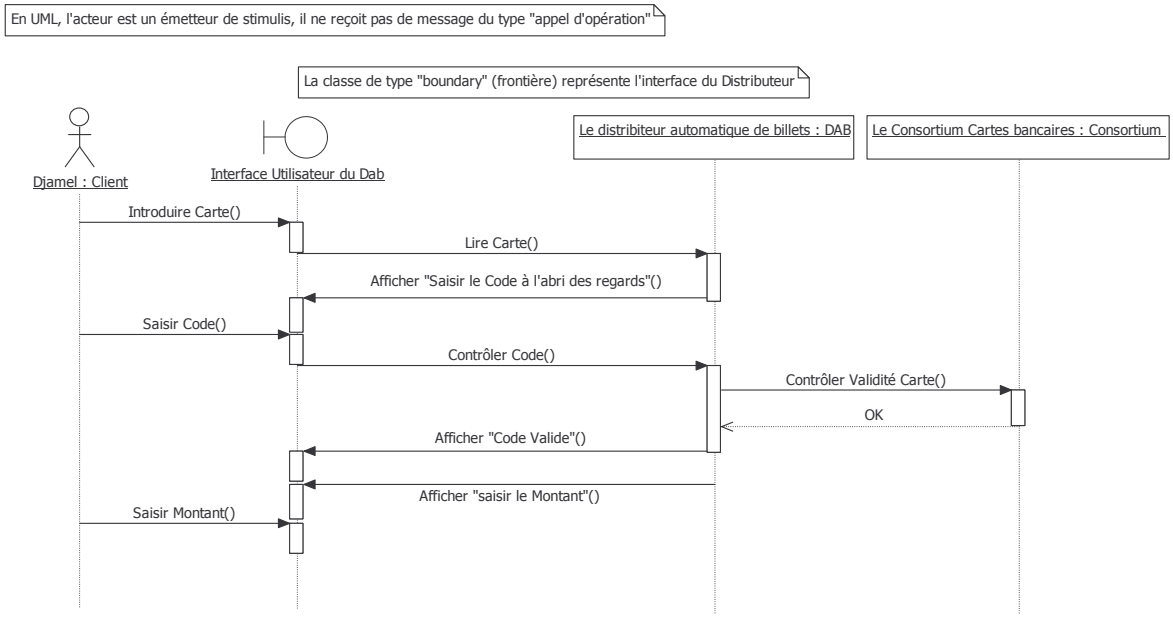

Figure 26: Interaction entre un acteur (client) et un objet (DAB) avec une interface « boundary »

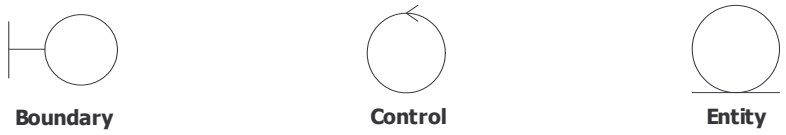

Figure 27: Les trois représentations des classes d'analyse

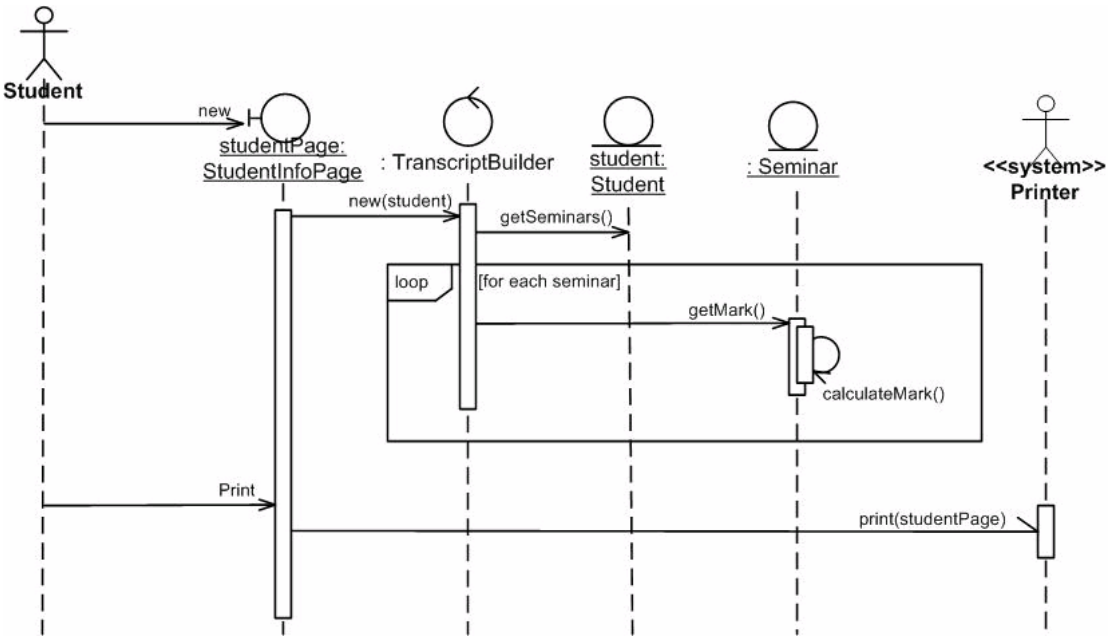

Figure 28: Des exemples de classes d'analyse

5 Conclusion

Les diagrammes de séquence d'objets permettront de valider que les acteurs du système, les objets, les classes auxquelles ils appartiennent, les associations, les opérations et certains des attributs ont bien été identifiés. Ils constituent par ailleurs une spécification utile pour le codage d'un algorithme ou la conception d'un automate.

Avec les diagrammes de séquences on cherche à mettre en valeur les passages de messages (déclenchant des événements) entre acteurs et objets (ou entre objets et objets) en les ordonnant dans le temps.
Un diagramme de séquences est un moyen semi-formel de capturer le comportement de tous les objets et acteurs impliqués dans un scénario d'un cas d'utilisation.

Le standard UML2.0 a apporté une refonte majeure aux diagrammes de séquence. Il est maintenant possible de décrire et spécifier les exigences sous forme d'un ensemble de diagrammes de séquence de base et par la suite de les composer en utilisant des opérateurs pour obtenir des scénarios plus complexes. Aussi, les diagrammes de séquence dans UML2.0 sont maintenant considérés comme des collections partiellement ordonnées d'événements (au lieu des collections ordonnées de messages dans UML1.4).cela permet d'introduire la concurrence et l'asynchronisme et ouvre ces diagrammes à la définition de comportements plus complexes que ne le permettait la version 1.4.

Index du texte :

Index des figures