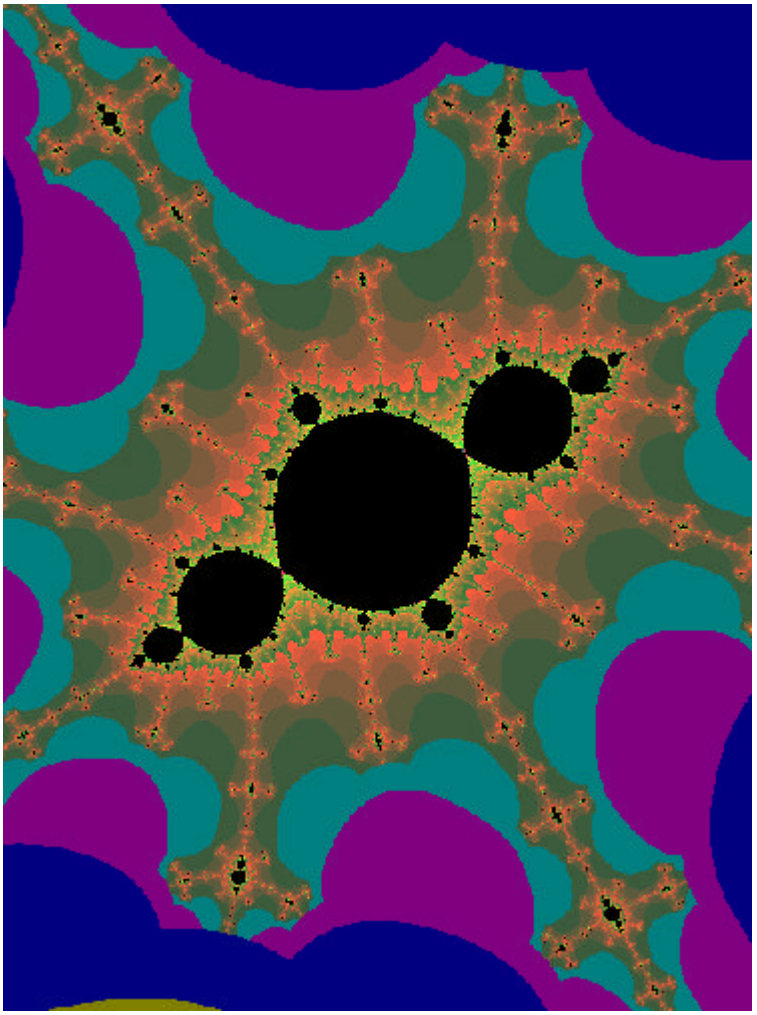
CHAPITRE 13

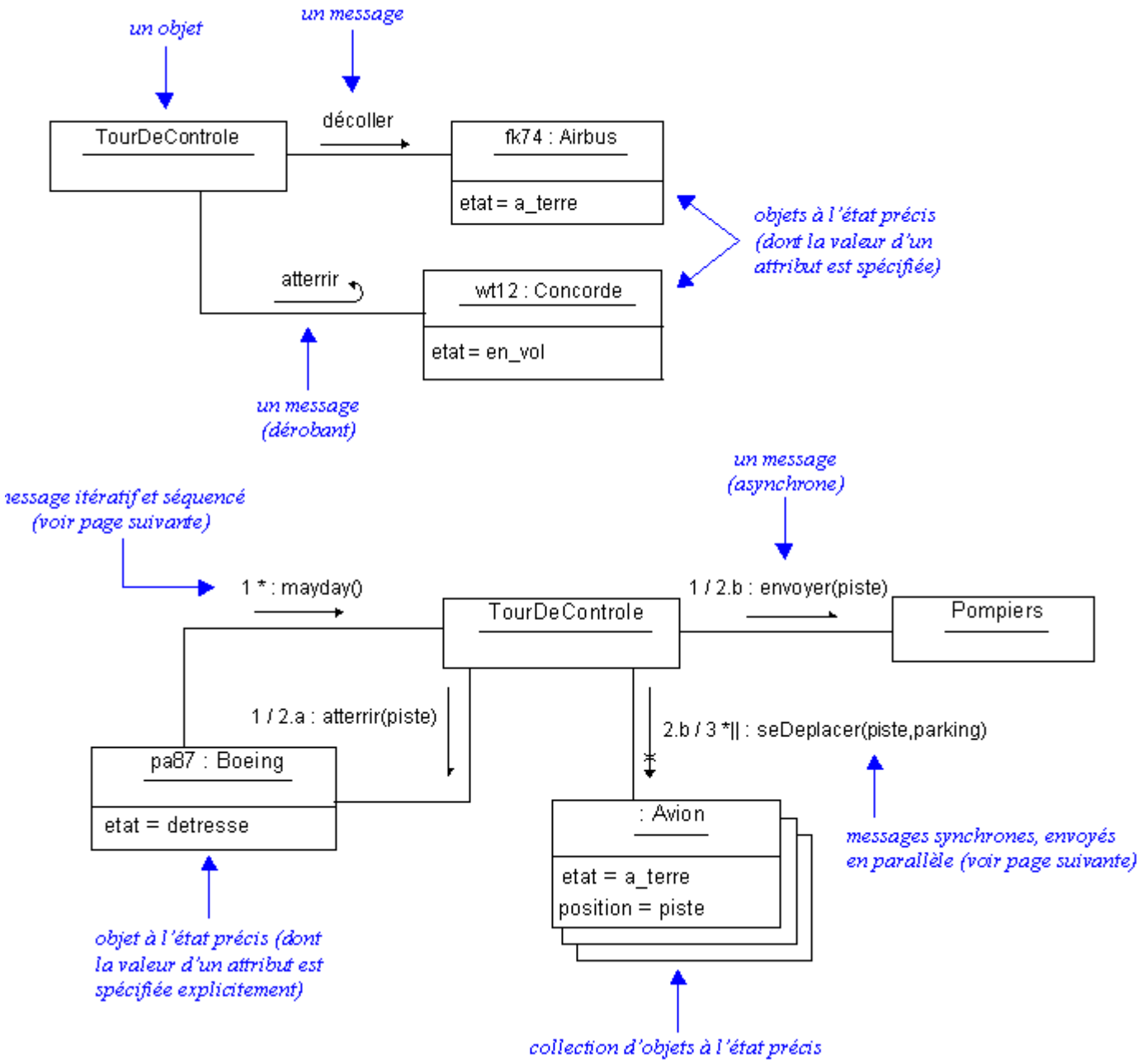
UML, les diagrammes de séquence



13.1 *Les diagrammes de collaboration*

Les diagrammes de collaboration montrent des interactions entre objets (instances de classes et acteurs). Ils permettent de représenter le contexte d'une interaction, car on peut y préciser les états des objets qui interagissent.

FIGURE 13.1 Exemples de diagrammes de collaboration



13.1.1 *Synchronisation des messages*

UML permet de spécifier de manière très précise l'ordre et les conditions d'envoi des messages sur un diagramme dynamique.

Pour chaque message, il est possible d'indiquer :

- les clauses qui conditionnent son envoi,
- son rang (son numéro d'ordre par rapport aux autres messages),
- sa récurrence,
- ses arguments.

La syntaxe d'un message est la suivante :

```
[pré "/"] [["["cond"]"] [séq] ["*"["||"]["["iter"]"]] ":"] [r ":="]
msg"("[par]")"
```

pré : prédécesseurs (liste de numéros de séquence de messages séparés par une virgule ; voir aussi "séq"). Indique que le message courant ne sera envoyé que lorsque tous ses prédécesseurs le seront aussi (permet de synchroniser l'envoi de messages).

cond : garde, expression booléenne. Permet de conditionner l'envoi du message, à l'aide d'une clause exprimée en langage naturel.

séq : numéro de séquence du message. Indique le rang du message, c'est-à-dire son numéro d'ordre par rapport aux autres messages. Les messages sont numérotés à la façon de chapitres dans un document, à l'aide de chiffres séparés par despoints. Ainsi, il est possible de représenter le niveau d'emboîtement des messages et leur précédence.

Exemple : l'envoi du message 1.3.5 suit immédiatement celui du message 1.3.4 et ces deux messages font partie du flot (de la famille de messages) 1.3.

Pour représenter l'envoi simultané de deux messages, il suffit de les indexer par une lettre.

Exemple : l'envoi des messages 1.3.a et 1.3.b est simultané.

iter : récurrence du message. Permet de spécifier en langage naturel l'envoi séquentiel (ou en parallèle, avec "||") de messages. Notez qu'il est aussi possible de spécifier qu'un message est récurrent en omettant la clause d'itération (en n'utilisant que "*" ou "*||").

r : valeur de retour du message. Permet d'affecter la valeur de retour d'un message, pour par exemple la retransmettre dans un autre message, en tant que paramètre.

msg : nom du message.

par : paramètres (optionnels) du message.

Exemples :

```
3 : bonjour()
```

Ce message a pour numéro de séquence "3”

```
[heure = midi] 1 : manger()
```

Ce message n'est envoyé que s'il est midi.

```
1.3.6 * : ouvrir()
```

Ce message est envoyé de manière séquentielle un certain nombre de fois.

```
3 / *||[i := 1..5] : fermer()
```

Représente l'envoi en parallèle de 5 messages. Ces messages ne seront envoyés qu'après l'envoi du message 3.

```
1.3,2.1 / [t < 10s] 2.5 : age := demanderAge(nom,prenom)
```

Ce message (numéro 2.5) ne sera envoyé qu'après les messages 1.3 et 2.1, et que si "t < 10s".

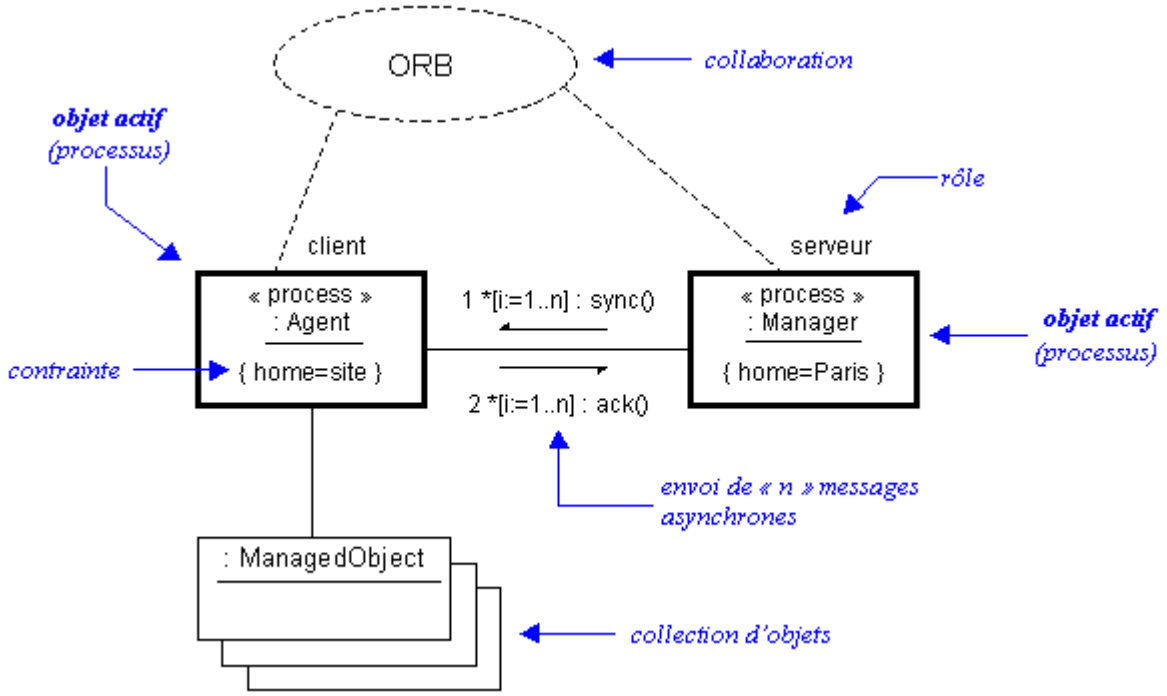
```
1.3 / [disk full] 1.7.a * : deleteTempFiles()
1.3 / [disk full] 1.7.b : reduceSwapFile(20%)
```

Ces messages ne seront envoyés qu'après l'envoi du message 1.3 et si la condition "disk full" est réalisée. Si cela est le cas, les messages 1.7.a et 1.7.b seront envoyés simultanément. Plusieurs messages 1.7.a peuvent être envoyés.

13.1.2 *Objets actifs (threads)*

UML permet de représenter des communications entre objets actifs de manière concurrente. Cette extension des diagrammes de collaboration permet notamment de représenter des communications entre processus ou l'exécution de threads.

FIGURE 13.2 Objets actifs



13.2 *Les diagrammes de séquence*

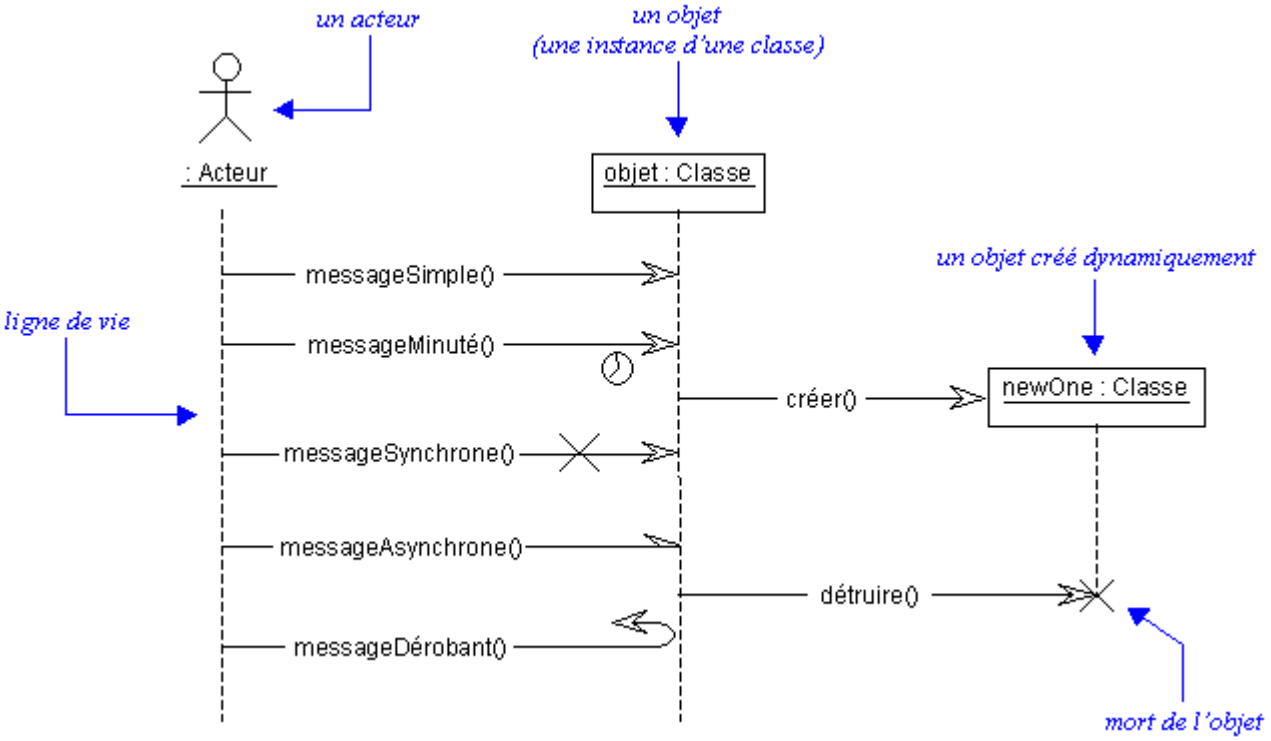
13.2.1 *Diagramme de séquence : sémantique*

Les diagrammes de séquences permettent de représenter des collaborations entre objets selon un point de vue temporel, on y met l'accent sur la chronologie des envois de messages. Contrairement au diagramme de collaboration, on n'y décrit pas le contexte ou l'état des objets, la représentation se concentre sur l'expression des interactions.

Les diagrammes de séquences peuvent servir à illustrer un cas d'utilisation.

L'ordre d'envoi d'un message est déterminé par sa position sur l'axe vertical du diagramme ; le temps s'écoule "de haut en bas" de cet axe. La disposition des objets sur l'axe horizontal n'a pas de conséquence pour la sémantique du diagramme. Les diagrammes de séquences et les diagrammes d'état-transitions sont les vues dynamiques les plus importantes d'UML.

FIGURE 13.3 Exemple de diagramme de séquence



13.2.2 *Types de messages*

Comme on peut le voir dans l'exemple ci-dessus, UML propose un certain nombre de stéréotypes graphiques pour décrire la nature du message (ces stéréotypes graphiques s'appliquent également aux messages des diagrammes de collaborations) :

message simple

Message dont on ne spécifie aucune caractéristique d'envoi ou de réception particulière.

message minuté (timeout)

Bloque l'expéditeur pendant un temps donné (qui peut être spécifié dans une contrainte), en attendant la prise en compte du message par le récepteur. L'expéditeur est libéré si la prise en compte n'a pas eu lieu pendant le délai spécifié.

message synchrone

Bloque l'expéditeur jusqu'à prise en compte du message par le destinataire. Le flot de contrôle passe de l'émetteur au récepteur (l'émetteur devient passif et le récepteur actif) à la prise en compte du message.

message asynchrone

N'interrompt pas l'exécution de l'expéditeur. Le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré (jamais traité).

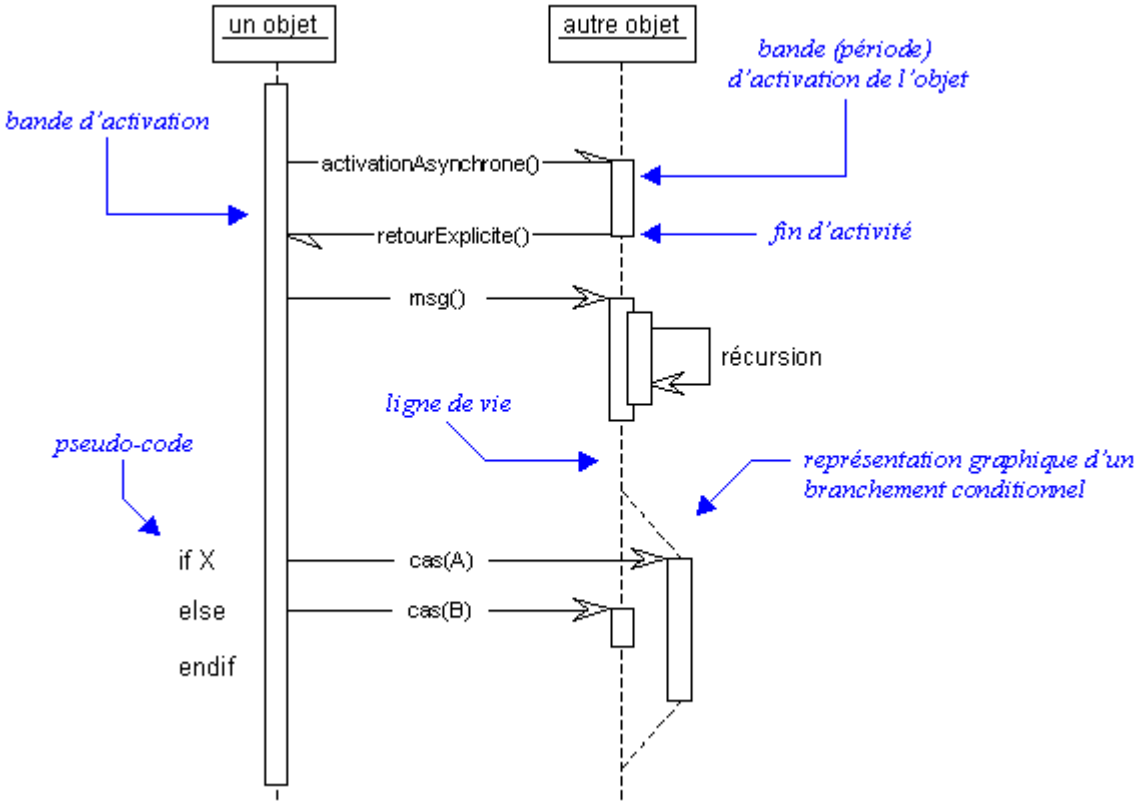
message dérobant

N'interrompt pas l'exécution de l'expéditeur et ne déclenche une opération chez le récepteur que s'il s'est préalablement mis en attente de ce message.

13.2.3 *Activation d'un objet*

Sur un diagramme de séquence, il est aussi 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. On peut aussi représenter des messages récursifs, en dédoublant la bande d'activation de l'objet concerné. 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 13.4 **Exemple : bande d'activation**



Commentaires :

Ne confondez pas la période d'activation d'un objet avec sa création ou sa destruction. Un objet peut être actif plusieurs fois au cours de son existence (voir exemple ci-dessus).

Le pseudo-code peut aussi être utilisé pour indiquer des itérations (avec incrémentation d'un paramètre d'un message par exemple).

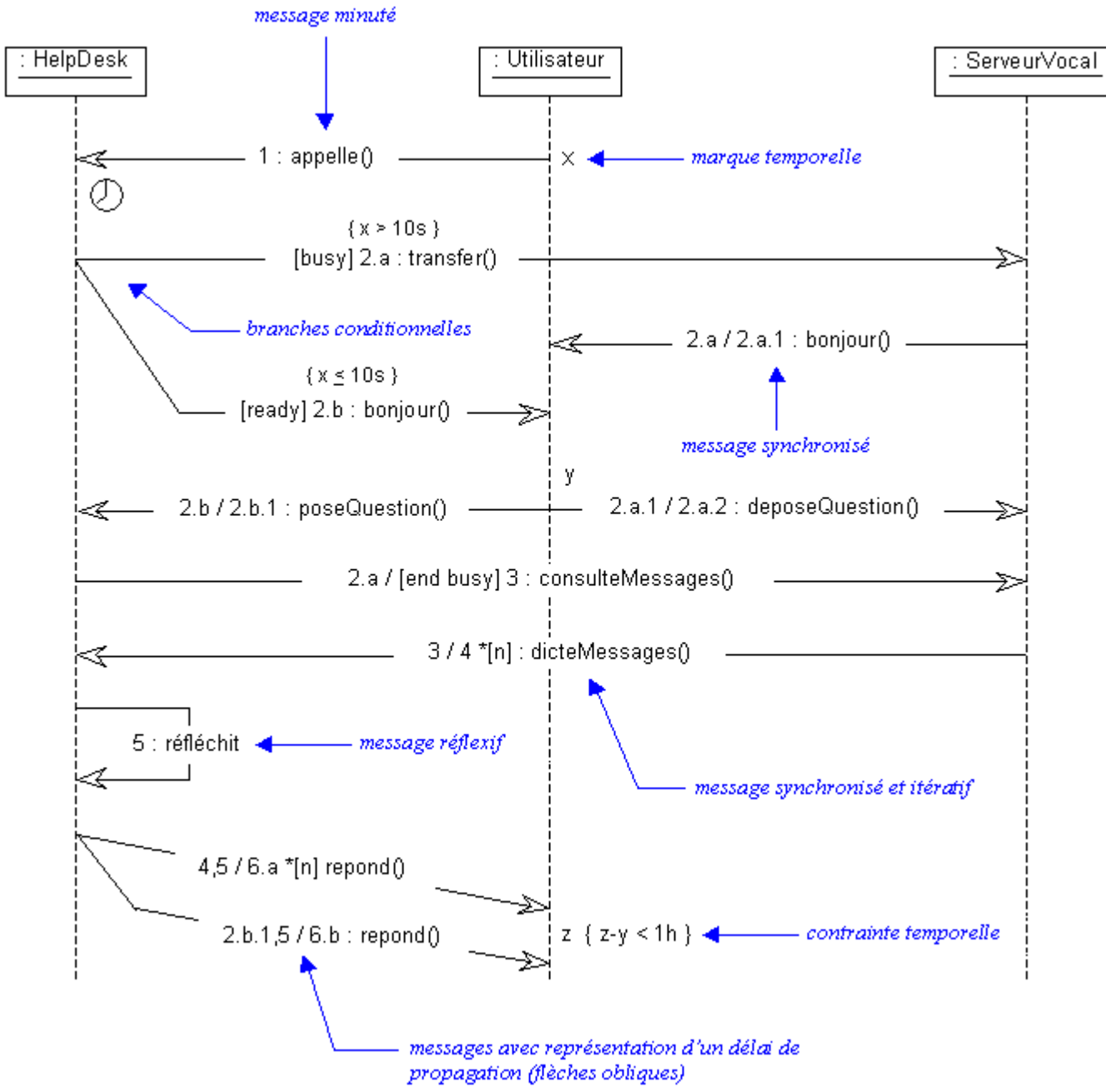
Le retour des messages asynchrones devrait toujours être matérialisé, lorsqu'il existe.

Notez qu'il est fortement recommandé de synchroniser vos messages, comme sur l'exemple qui suit...

L'exemple qui suit présente aussi une alternative intéressante pour la représentation des branchementsconditionnels. Cette notation est moins lourde que celle utilisée dans l'exemple ci-dessus.

Préférez aussi l'utilisation de contraintes à celle de pseudo-code, comme dans l'exemple qui suit.

FIGURE 13.5 Exemple complet



Afin de mieux comprendre l'exemple ci-dessus, veuillez vous référer aux chapitres sur la synchronisation des messages.

Notez aussi l'utilisation des contraintes pour documenter les conditions d'envoi de certains messages.

Commentaire :

Un message réflexif ne représente pas l'envoi d'un message, il représente une activité interne à l'objet (qui peut être détaillée dans un diagramme d'activités) ou une abstraction d'une autre interaction (qu'on peut détailler dans un autre diagramme de séquence).