IFT2255 - Génie logiciel

Diagrammes de séquence

Bruno Dufour, Houari Sahraoui, Julie Vachon
{dufour,sahraouh,vachon}@iro.umontreal.ca

Cas d'utilisation

- Décrit les interactions du logiciel avec son environnement
  - point de vue des utilisateurs (actions et réactions)
- Permet d'identifier et de décrire les fonctionnalités d'un logiciel qui sont significatives pour ses utilisateurs
  - humains, matériels, logiciels

Principaux concepts

- **Acteur** : représentation idéalisée d'une personne, d'un logiciel ou d'un processus qui interagit (depuis l'extérieur) avec le logiciel
- **Scénario** : une séquence spécifique d'interactions entre les acteurs et le système
  - Un cas d'utilisation est composé d'un ou plusieurs scénarios connexes

Cas d'utilisation - exemple

1. Le client introduit sa carte bancaire
2. Le logiciel lit la carte et vérifie que la carte est valide
3. Le logiciel demande au client de taper son code
4. Le client tape son code confidentiel
5. Le logiciel vérifie que le code correspond à la carte
6. Le client choisi une opération de retrait
7. Le logiciel demande le montant à retirer

   etc.

Diagramme de séquence

- Un cas d'utilisation est réalisé par une **collaboration**
  - Ensemble d'objets qui s'échangent des messages et travaillent ensemble et pour accomplir une tâche
- Un diagramme de séquence
  - permet décrire les cas d'utilisation de façon à mettre en évidence les interactions entre les instances des classes (objets) du logiciel
  - montre la séquence dans le temps des échanges de messages entre les objets participant à un scénario

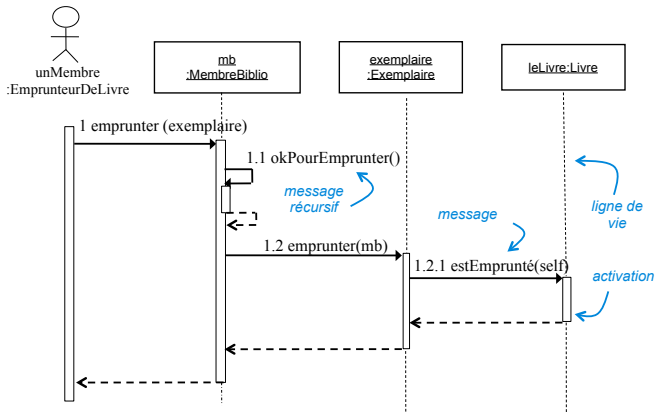
Dimensions

- Dimension verticale : le temps
  - 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
- Dimension horizontale : les objets (et les acteurs)
  - L'ordre de disposition des objets sur l'axe horizontal est sans importance

Éléments graphiques

- Les **objets** qui interagissent dans le scénario
- Représentation graphique de la **ligne de vie** de chaque objet et de ses **activations**
- Les différents types de **messages** envoyés
  - simple, synchrone, asynchrone
- Les indications de **contrôle**
  - branchement conditionnel et itération
  - création et destruction d'objets
  - délais de transmission
  - contraintes temporelles

Diagramme de séquence - exemple



Modes d'intéraction

- Mode procédural (synchrone)
  - Au plus un objet détient le contrôle à la fois
- Mode concurrent (asynchrone)
  - Plusieurs peuvent être actifs à un instant donné

Activation et contrôle

- Un objet obtient le contrôle quand il reçoit un message (une de ses méthodes est invoquées), ce moment détermine le début de son **activation**
- Un objet rend le contrôle à son destinateur lorsqu'il lui renvoie sa réponse, ce moment détermine la fin de son activation
- Lorsqu'un objet en activation a le contrôle, il peut
  - Faire des calculs
  - Envoyer des messages : dans ce cas, l'objet demeure en activation mais cède à son tour le contrôle à son destinataire, il ne peut rien faire jusqu'à ce qu'il reçoive la réponse de son destinataire (message « bloquant »)

Mode procédural

- Les objets sont dits « passifs », car on doit leur envoyer un message pour déclencher leur activation
- Un seul fil d'exécution
- Dans ce mode, seul un acteur peut envoyer un message ou faire des calculs sans l'intervention de qui que ce soit
  - Son activation est constante

Mode procédural

- La période d'activation d'un objet est représentée par une bande rectangulaire superposée à sa ligne de vie
  - on dédouble la bande d'activation dans le cas d'appels récursifs
- L'identité des objets en activation est gérée par une pile
  - Dessus de la pile = objet en activation qui possède le contrôle
  - On empile l'identité d'un objet qui débute une activation
  - On dépile l'identité d'un objet qui termine une activation

Ordonnancement des messages

- Les messages sont numérotés de façon à refléter l'imbrication des envois de messages.
  - ex: le message 2.2. est envoyé après que la réponse au message 2.1 ait été reçue
  - ex: il faut attendre la réponse au message 2.2.3. avant de pouvoir répondre au message 2.2.

Mode concurrent

- Certains objets disposent d'un propre fil d'exécution
  - Il ont leur propre contrôle et leur activation est constante
  - Ces objets sont dits « actifs », car il peuvent faire des calculs et envoyer des messages sans l'intervention de qui que ce soit
- Les objets actifs envoient deux types de messages
  - Synchrones : ils attendent la réponse de leur destinataire avant de poursuivre
  - Asynchrones : ils poursuivent leurs activités sans attendre la réponse à leur message, celle-ci leur sera signalée

Messages synchrones

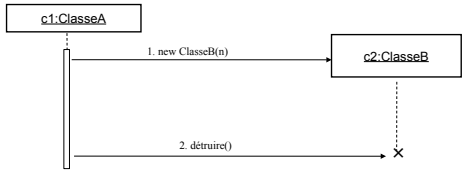
- Suite à l'envoi de son message, l'expéditeur (objet passif ou actif) perd le contrôle mais demeure en activation →
  - Il demeure bloqué jusqu'à ce que le destinataire ait fini de traiter le message
- Il retrouve ensuite le contrôle avec la réponse à son message ⇠

Messages asynchrones

- L'expéditeur envoie un message au destinataire sans attendre de réponse →
- Si l'expéditeur est un objet passif (rare)
  - l'envoi du message équivaut à l'envoi d'un message simple (non synchrone)
  - l'expéditeur perd le contrôle et termine son activation après l'envoi du message
- Si l'expéditeur est un objet actif :
  - cet envoi de message équivaut à l'envoi d'un message asynchrone
  - suite à l'envoi de son message, l'expéditeur demeure en activation et conserve son flot de contrôle, son exécution n'est pas interrompue, le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré (jamais traité), une réponse n'est pas nécessairement attendue

Création et destruction d'instance

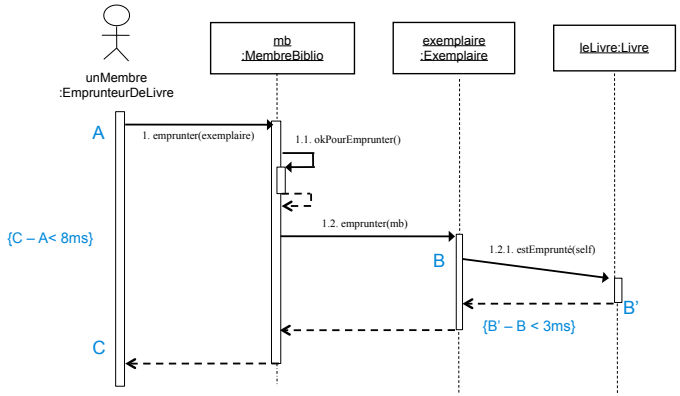
- Création : lorsqu'une instance est créée, on place la boîte qui la représente légèrement en bas, à l'endroit où elle est créée
- Destruction : la destruction d'une instance est représentée par un X à la fin de sa ligne de vie, là où elle se termine
  - Si l'instance est détruite par une autre instance (non par elle-même), un message issu de l'instance destructrice pointe sur le X



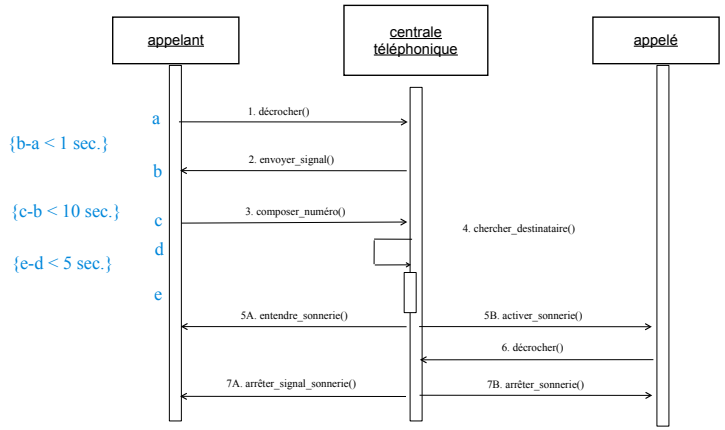
Contraintes temporelles

- On peut spécifier le temps et des contraintes temporelles
- Les intervalles sur la ligne de vie des objets indiquent des intervalles de temps
- On peut attribuer une étiquette au moment d'émission et de réception d'un messages et spécifier des contraintes sur ces étiquettes
- L'envoi d'un message est généralement considéré comme instantané, pour marquer une durée significative, on incline la flèche du message vers le bas (au lieu de la laisser horizontale)

Contraintes temporelles



Contraintes temporelles



Scénarios

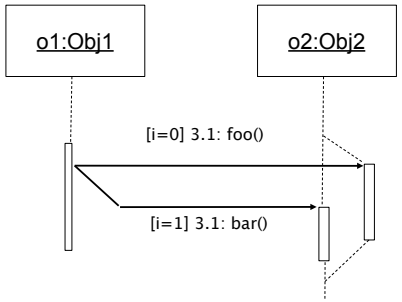
- On peut vouloir décrire les différents scénarios d'un cas d'utilisation sur un même diagramme de séquence
  - Conditions (gardes) sur les messages
  - Ligne de vie à branches multiples
  - Itérations
  - Interactions concurrentes

Messages avec garde

- Permet d'exprimer les alternatives de comportement
  - Message envoyé seulement si la garde est vraie
  - Les gardes doivent être mutuellement exclusives pour signifier que l'envoi est conditionnel (et non concurrent)

Ligne de vie à branches multiples

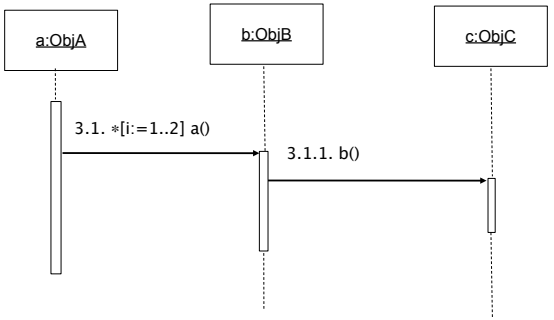
- Si deux messages, dont l'envoi est conditionnel à l'évaluation d'une garde, sont destinés à un même objet, on doit pouvoir exprimer le comportement de l'objet dans chaque cas



Itération

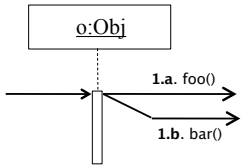
- On ajoute une clause d'itération (introduite par un `*`) à côté du message pour spécifier le nombre d'itérations ou l'invariant de l'itération
  - `*[i:=1..10] m()`
    - Le message est envoyé 10 fois
  - `*[x<10] m()`
    - Le message est envoyé de façon répétée jusqu'à ce que x soit plus grand ou égal à 10
  - `*[item not found] m()`
    - Le message est envoyée de façon répétée jusqu'à ce que l'item soit trouvé

Itération



Interactions concurrentes

- Il existe différente façon de créer de nouveaux fils de contrôle (*threads*)
  - À la réception d'un message, un objet peut découpler le fil d'exécution en envoyant plusieurs messages simultanément (*fork*)



Branchement : lorsque les messages n'ont pas de garde mutuellement exclusive, ils sont concurrents

Interactions concurrentes

- Il existe différente façon de créer de nouveaux fils de contrôle (*threads*)
  - Un acteur peut spontanément décider d'envoyer un message et crée ainsi un nouveau fil d'exécution.
  - Un objet actif peut également découpler le fil d'exécution en envoyant un message asynchrone.