annexe

A

Résumé du sous-ensemble de la notation UML 2 utilisé dans ce livre

Diagramme de cas d'utilisation

Diagramme de séquence

Diagramme de classes

Diagramme de packages

Diagramme d'états

Diagramme de cas d'utilisation

Montre les interactions fonctionnelles entre les acteurs et le système à l'étude

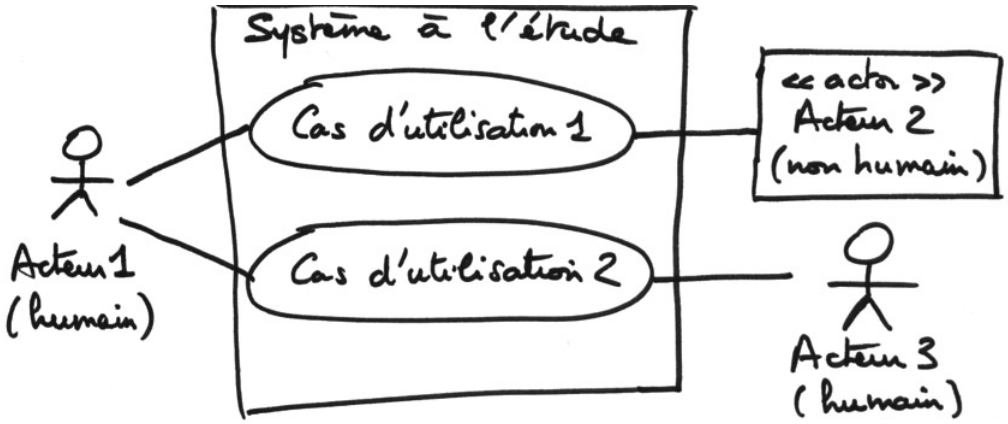

Figure A–1 Diagramme de cas d'utilisation (bases)

Acteur : rôle joué par un utilisateur humain ou un autre système qui interagit directement avec le système étudié. Un acteur participe à au moins un cas d'utilisation.

Cas d'utilisation (use case) : ensemble de séquences d'actions réalisées par le système produisant un résultat observable intéressant pour un acteur particulier. Collection de scénarios reliés par un objectif utilisateur commun.

Association : utilisée dans ce type de diagramme pour relier les acteurs et les cas d'utilisation par une relation qui signifie simplement « participe à ».

Inclusion : le cas d'utilisation de base en incorpore explicitement un autre, de façon obligatoire, à un endroit spécifié dans ses enchaînements.

Extension : le cas d'utilisation de base en incorpore implicitement un autre, de façon optionnelle, à un endroit spécifié indirectement dans celui qui procède à l'extension (déconseillé !)

Généralisation : les cas d'utilisation descendants héritent de la description de leur parent commun. Chacun d'entre eux peut néanmoins comprendre des relations spécifiques supplémentaires avec d'autres acteurs ou cas d'utilisation (déconseillé !). La généralisation d'acteurs est en revanche parfois utile.

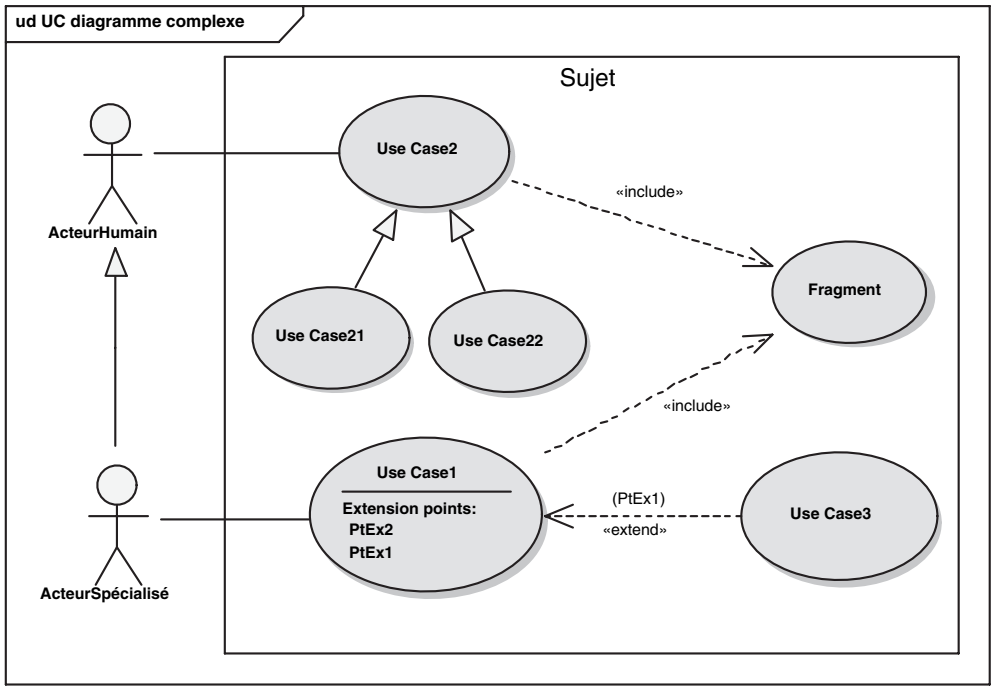

Figure A–2 Diagramme de cas d'utilisation (avancé)

Diagramme de séquence

Montre la séquence verticale des messages passés entre objets au sein d'une interaction

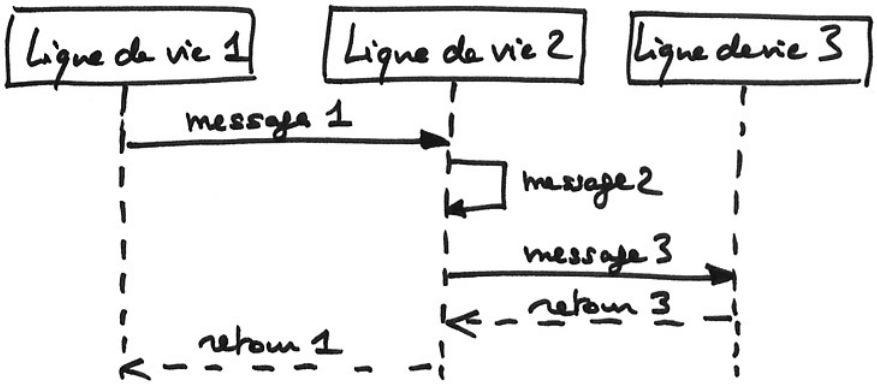

Figure A–3
Diagramme de séquence (bases)

Ligne de vie : représentation de l'existence d'un élément participant dans un diagramme de séquence. Cela peut être un acteur ou le système en modélisation d'exigences, des objets logiciels en conception préliminaire ou conception détaillée.

Message : élément de communication unidirectionnel entre objets qui déclenche une activité dans l'objet destinataire. La réception d'un message provoque un événement dans l'objet récepteur. La flèche pointillée représente un retour au sens UML. Cela signifie que le message en question est le résultat direct du message précédent.

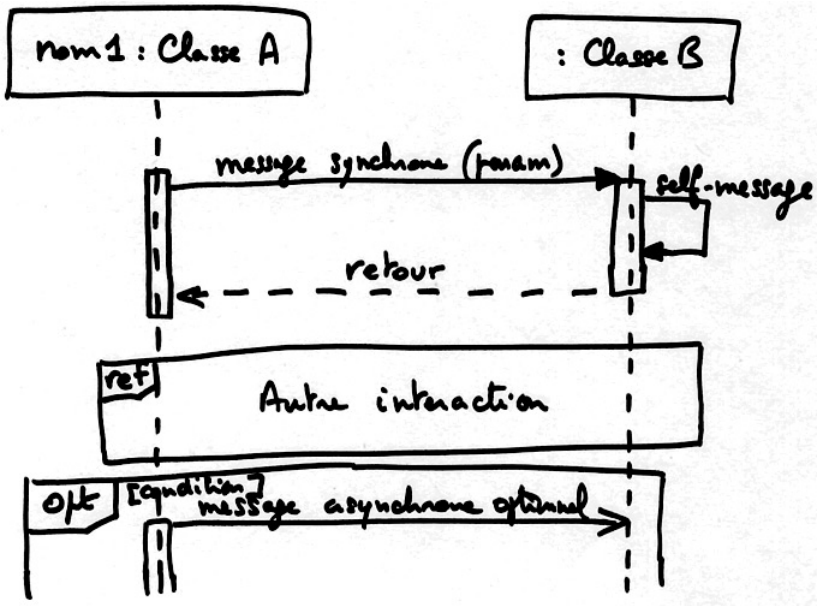

Figure A–4
Diagramme de séquence (avancé)

Spécification d'activation : bande blanche qui représente une période d'activité sur une ligne de vie.

Message synchrone : envoi de message pour lequel l'émetteur se bloque en attente du retour et qui est représenté par une flèche pleine. Un message asynchrone, au contraire, est représenté par une flèche ouverte.

Occurrence d'interaction : une interaction peut faire référence explicitement à une autre interaction grâce à un cadre avec le mot-clé `ref` et indiquant le nom de l'autre interaction.

UML 2 a ajouté une nouvelle notation très utile : les cadres d'interaction. Chaque cadre possède un opérateur et peut être divisé en fragments. Les principaux opérateurs sont :

- `loop` : boucle. Le fragment peut s'exécuter plusieurs fois, et la condition de garde explicite l'itération.

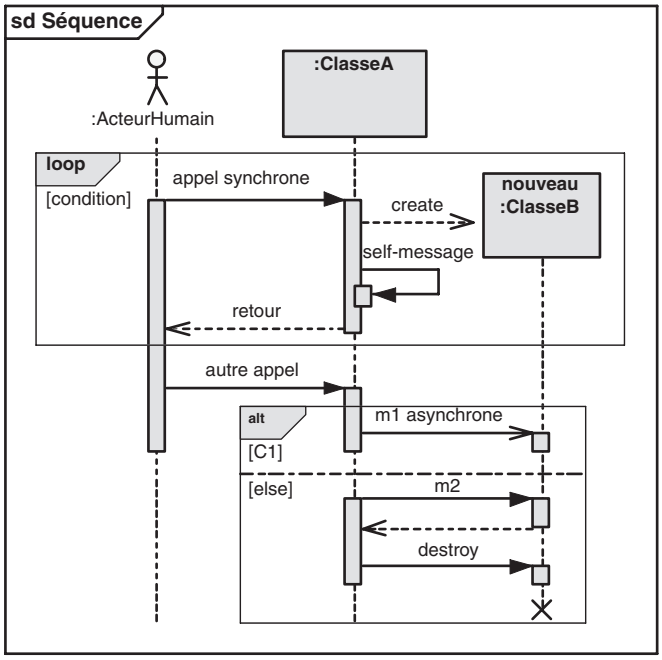

Figure A–5
Diagramme de séquence (avancé-suite)

- `opt` : optionnel. Le fragment ne s'exécute que si la condition fournie est vraie.
- `alt` : fragments alternatifs. Seul le fragment possédant la condition vraie s'exécutera.

Diagramme de classes

Montre les briques de base statiques : classes, associations, interfaces, attributs, opérations, généralisations, etc.

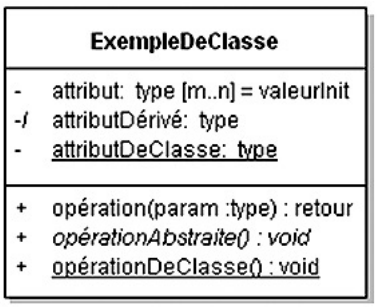

Figure A–6
Diagramme de classes (bases)

Classe : description abstraite d'un ensemble d'objets qui partagent les mêmes propriétés (attributs et associations) et comportements (opérations et états).

Attribut : donnée déclarée au niveau d'une classe, éventuellement typée, à laquelle chacun des objets de cette classe donne une valeur. Un attribut peut posséder une multiplicité et une valeur initiale. Un attribut dérivé (« `/` ») est un attribut dont la valeur peut être déduite d'autres informations disponibles dans le modèle.

Opération : élément de comportement des objets, défini de manière globale dans leur classe. Une opération peut déclarer des paramètres (eux-mêmes typés) ainsi qu'un type de retour.

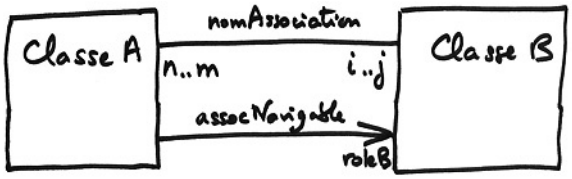

Figure A–7
Diagramme de classes (bases-suite1)

Association : relation sémantique durable entre deux classes, qui décrit un ensemble de liens entre instances. Une association est bidirectionnelle par défaut, sauf si l'on restreint sa navigabilité en ajoutant une flèche.

Rôle : nom donné à une extrémité d'une association ; par extension, manière dont les instances d'une classe voient les instances d'une autre classe au travers d'une association.

Multiplicité : le nombre d'objets (`min..max`) qui peuvent participer à une relation avec un autre objet dans le cadre d'une association. Multiplicités fréquentes :

- `0..1` = optionnel (mais pas multiple)
- `1` = exactement 1
- `0..*` = `*` = quelconque
- `1..*` = au moins 1

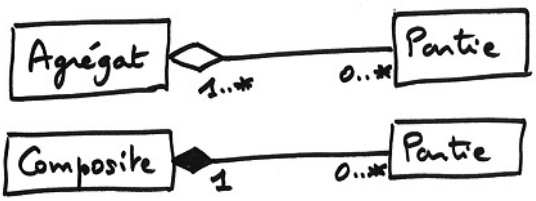

Figure A–8
Diagramme de classes (bases-suite2)

Agrégation : cas particulier d'association non symétrique exprimant une relation de contenance.

Composition : forme forte d'agrégation, dans laquelle les parties ne peuvent appartenir à plusieurs agrégats et où le cycle de vie des parties est subordonné à celui de l'agrégat.

Figure A–9
Diagramme de classes (bases-suite3)

Super-classe : classe générale reliée à d'autres classes plus spécialisées (sous-classes) par une relation de généralisation.

Généralisation : relation entre « classifieurs » où les descendants héritent des propriétés de leur parent commun. Ils peuvent néanmoins comprendre chacun des propriétés spécifiques supplémentaires, mais aussi modifier les comportements hérités.

Figure A–10
Diagramme de classes (bases-suite4)

Note : commentaire visible dans le diagramme. Peut être attaché à un élément du modèle (ou plusieurs) par un trait pointillé. Utilisable dans tout type de diagramme UML !

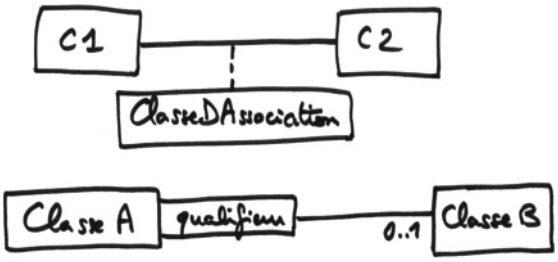

Figure A–11
Diagramme de classes (avancé)

Classe d'association : association promue au rang de classe. Elle possède tout à la fois les caractéristiques d'une association et celles d'une classe et peut donc porter des attributs qui prennent des valeurs pour chaque lien entre objets.

Qualifieur (ou qualificatif) : attribut qui permet de « partitionner » l'ensemble des objets en relation avec un objet donné dans le cadre d'une association multiple.

Figure A–12
Diagramme de classes (avancé-suite1)

Contrainte : condition portant sur un ou plusieurs élément(s) du modèle qui doit être vérifiée par les éléments concernés. Elle est notée entre accolades `{ }`, et souvent insérée dans une note graphique (le post-it).

Figure A–13
Diagramme de classes (avancé-suite2)

Dépendance : relation sémantique entre deux éléments, dans laquelle la modification d'un des éléments (l'élément indépendant) peut affecter la sémantique de l'autre élément (l'élément dépendant).

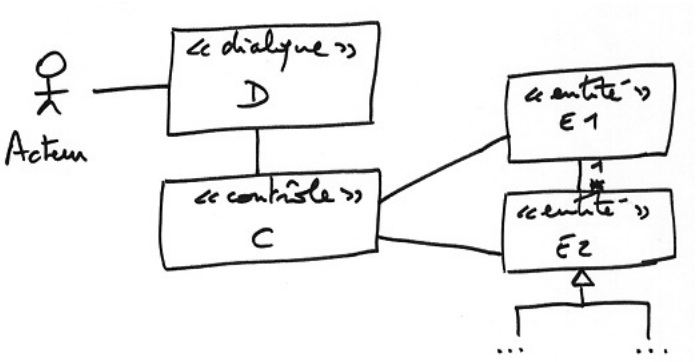

Figure A–14
Diagramme de classes (avancé-suite3)

Catégorisation des **classes d'analyse** en trois catégories qui a été proposée par I. Jacobson et popularisée ensuite par le RUP (Rational Unified Process) :

- Celles qui permettent les interactions entre le site web et ses utilisateurs sont qualifiées de « dialogues ». Il s'agit typiquement des écrans proposés à l'utilisateur.
- Les classes qui contiennent la cinématique de l'application sont appelées « contrôles ». Elles font la transition entre les dialogues et les concepts du domaine, et contiennent les règles applicatives.
- Celles qui représentent les concepts métier sont qualifiées d'« entités ». Elles sont très souvent persistantes.

Diagramme de packages

Montre l'organisation logique du modèle et les relations entre packages

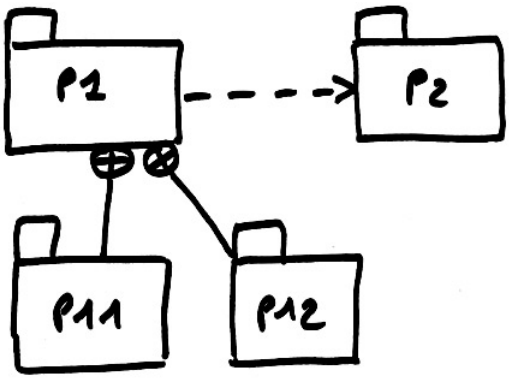

Figure A–15
Diagramme de packages

Package (ou paquetage) : mécanisme général de regroupement d'éléments tels que classes, interfaces, mais aussi acteurs, cas d'utilisation, etc. Les packages peuvent être imbriqués dans d'autres packages.

Importation : relation de dépendance entre packages qui rend visibles les éléments publics de l'un des packages au sein d'un autre.

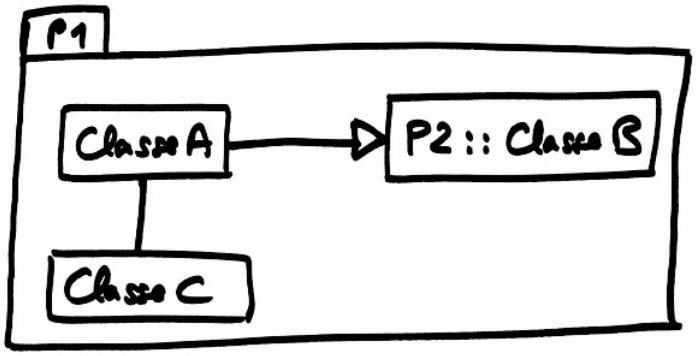

Figure A–16
Diagramme de packages (suite)

Les packages sont des espaces de noms (*namespace*) : deux éléments ne peuvent pas porter le même nom au sein du même package. En revanche, deux éléments appartenant à des packages différents peuvent porter le même nom. Du coup, UML propose une notation complète pour un élément : `nomPackage::nomÉlément`.

Diagramme d'états

Montre les différents états et transitions possibles des objets d'une classe à l'exécution

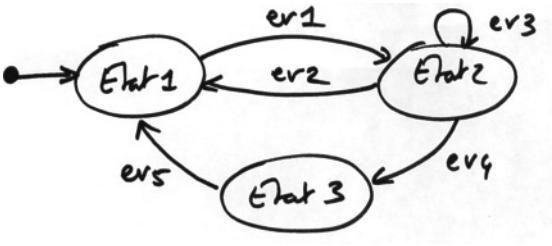

Figure A–17
Diagramme d'états (bases)

État : condition ou situation qui se produit dans la vie d'un objet pendant laquelle il satisfait une certaine condition, exécute une activité particulière ou attend certains événements.

Événement : spécification d'une occurrence remarquable qui a une localisation dans le temps et l'espace. Un événement peut porter des paramètres qui matérialisent le flot d'information ou de données entre objets.

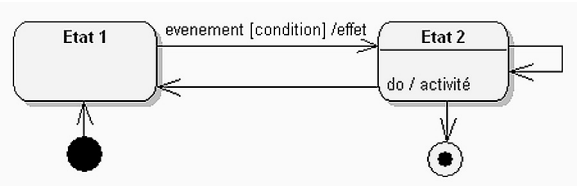

Figure A–18
Diagramme d'états (bases-suite)

Transition : connexion entre deux états d'un automate, qui est déclenchée par l'occurrence d'un événement, et conditionnée par une condition de garde, induisant certains effets.

Effet : action ou activité qui s'exécute lorsqu'une transition se déclenche. L'exécution de l'effet est unitaire et ne permet de traiter aucun événement supplémentaire pendant son déroulement.

Un état peut contenir une **activité durable** (do-activity), qui est interruptible par un événement, mais peut également se terminer d'elle-même.