Département d'Informatique
Institut Universitaire de Technologie
Université de la Méditerranée

Un exemple de méthodologie orientée objets : OMT

Vincent Risch

janvier 1997

Avertissement

Le présent support de cours en est à sa première édition. Non seulement il ne prétend à aucune originalité, mais il paraphrase abondamment et sans retenue l'ouvrage de référence de James Rumbaugh, *OMT, Modélisation et conception orientées objet* (version française éditée chez Masson). Toutefois chaque fois que cela nous a paru opportun, nous nous sommes permis de souligner explicitement certains présupposés implicitement présents dans la description faite par James Rumbaugh de sa méthode. Notre document présente probablement bien des lacunes. Parmi celles-ci, le manque de temps nous a empêché de développer les aspects analyse et conception, alors que l'accent a été mis sur la description des trois modèles qui composent la méthode. Inversement, il nous a été difficile d'échapper complètement au style jargonnant inhérent à la description de bien des méthodes d'analyse et de conception, OMT ne faisant pas exception. Nous espérons fermement que les remarques de nos collègues et des étudiants aidant, ce document se bonifiera avec le temps...

Remerciements

Pour la qualité de leur relecture et de leurs commentaires, merci à Didier Boitard et Rosine Ciccheti. Merci tout particulièrement à Jacques Le Maitre et Claude Sabatier pour avoir accepté de débattre longuement avec moi de certains points de la méthode.

Table des Matières

Avant-propos 1

Première Partie : Le modèle objet **4**

1 Introduction 5

2 Objets : instances et classes 5

3 Relations entre objets 6

4 Diagramme d'instances et diagramme de classes 7

5 Liens et associations : concepts évolués 9

5.1 Multiplicité 9

5.2 Nom de rôle 11

5.3 Qualifieur 11

5.4 Attribut de lien 12

5.5 Contraintes 14

5.6 Objet dérivé 15

5.7 Homomorphisme entre associations 15

6 Agrégation et Généralisation 15

6.1 Agrégation 16

6.2 Généralisation 18

6.3 Résumé comparatif de chacun des deux types de relation . . 21

7 Modèles Objet et Entité-association 21

Deuxième partie : Le modèle dynamique **23**

8 Introduction 24

9 Etats et événements 24

10 Scénarios et suivis d'événements 25

11 Diagramme d'états 26

12 Concepts objets et modèle dynamique 30
12.1 Diagrammes imbriqués/généralisation d'états 31
12.2 Généralisation d'événements 31
12.3 Etats concurrents 32
12.4 Activités concurrentes 33

Troisième partie : Le modèle fonctionnel **35**
13 Introduction 36
14 Diagrammes à flots de données 36
14.1 Traitements 36
14.2 Flots de données 36
14.3 Acteurs, Réservoirs et objets 37
15 Concepts évolués autour de la notion de flot 37
15.1 Création d'un objet 37
15.2 Flot de contrôle 38

Références bibliographiques **39**

Annexe A : Glossaire **40**

Annexe B : Notation **43**

Avant-propos

L'Analyse et la Conception des Systèmes Informatisés[1] est à certains égards l'une des plus "philosophiques" des disciplines informatiques. L'objet de l'ACSI est la définition de méthodologies adéquates permettant d'abstraire les éléments pertinents d'un problème en vue de son traitement informatique. A partir de cette définition se pose immédiatement la question d'une identification adéquate du critère de pertinence évoqué : or, aussi loin qu'on puisse aller, il semble inévitable qu'une part du processus d'abstraction reste irréductible à toute tentative de systématisation objective[2]. Ainsi, il est clair dès le départ que loin de toute tentation de modélisation "parfaite", une méthode d'analyse porte plus ou moins explicitement un "certain regard" sur la réalité qu'elle cherche à modéliser.

Parmi les quelques méthodologies répertoriées, une famille assez récente de méthodes se distingue par sa capacité à appréhender de façon relativement proche et naturelle une construction fondamentale dans bien des descriptions du monde réel : l'**objet**. L'idée sous-jacente est d'y combiner aussi étroitement que possible structures de données et comportement en *une seule* entité, par analogie avec le monde réel où l'on suppose un objet caractérisé uniquement par son identité et un comportement spécifique (par exemple, le téléphone posé sur mon bureau permet d'établir une télécommunication). L'expression "orienté objet" signifie que l'on organise à terme le logiciel comme une collection d'objets dissociés comprenant à la fois une structure de données et un comportement. Il s'agit là d'une différence essentielle avec la programmation conventionnelle dans laquelle les structures de données et le comportement ne sont que faiblement associés.

Les propriétés que l'on s'accorde voir généralement comme caractéristiques d'une approche orientée objet sont

identité: les données sont quantifiées en entités discrètes (i.e. sont en bijection avec les entiers naturels) que l'on peut distinguer et nommer objets (exemple : un paragraphe dans une documentation, une fenêtre sur l'écran d'une station de travail, la reine blanche aux échecs).

classification : les objets ayant la même structure de données (**attributs**) et le même comportement (**opérations**) sont regroupés dans une **classe** (exemple : paragraphe, fenêtre, pièce de jeux d'échecs).

polymorphisme : la *même* opération peut se comporter différemment sur différentes classes (exemple : l'opération "déplacer" sur les classes fenêtre et pièce de jeux d'échecs).

héritage : partage des **attributs** et des **opérations** entre **classes** s'appuyant sur une relation hiérarchique. Chaque classe hérite (incorpore) toutes les propriétés de sa super-classe et y ajoute ses propriétés spécifiques (exemple : fenêtre déroulante et fenêtre fixe sont des sous-classes de fenêtre).

[1] ou "ACSI" pour les intimes...

[2] Cf. à ce propos le conte philosophique intitulé *De la rigueur scientifique* dans [Borges, 60]

On notera que les objets peuvent être concrets (par ex. un fichier) ou conceptuels (par ex. une politique d'ordonnancement dans un système multitâche). Deux objets sont distincts même quand tous leurs attributs ont des valeurs identiques, un objet étant par ailleurs référencé par une clé unique.

Plus précisément, une classe est une abstraction qui décrit les propriétés pertinentes pour l'application et ignore les autres. Autrement dit, tout choix de classe est arbitraire et dépend de l'application. Chaque objet est dit **instance** de sa classe : noms d'attributs et opérations sont communs aux instances de la classe. Un objet contient une référence implicite à sa classe : *il sait ce qu'il est.*

L'implémentation spécifique d'une opération est appelée **méthode**. Parce qu'une opération est polymorphe, elle peut être implémentée par plus d'une méthode. Par analogie avec le monde réel où une opération est une abstraction de comportements analogues parmi différents types d'objets et où chaque objet "sait comment" effectuer ses propres opérations, un langage orienté objet sélectionne automatiquement la méthode correcte pour implémenter l'opération en se basant sur le nom de l'opération et sur la classe de l'objet sur lequel elle opère.

De nouvelles classes peuvent être ajoutées sans modifier le code existant pourvu que des méthodes soient fournies pour chaque opération applicable sur les nouvelles classes. La possibilité de factoriser des propriétés communes à plusieurs classes dans une super-classe commune et d'hériter les propriétés de celle-ci peut réduire de façon significative les répétitions dans la conception et dans les programmes.

A partir de ces caractéristiques, il est clair que les méthodes objet favorisent en outre

l'abstraction, qui consiste à se concentrer sur les aspects inhérents d'une entité et à en oublier les aspects accidentels,

l'encapsulation, qui consiste à séparer les aspects externes d'un objet, accessibles aux autres objets, des détails de son implémentation interne, rendus invisibles aux autres objets,

le partage des structures de données et la réutilisation du code.

Le présent document est consacré plus particulièrement à une des méthodes de modélisation orientée objet, baptisée OMT comme acronyme d'*Object Modeling Techniques*, et proposée par James Rumbaugh à la fin des années 80. A l'image d'autres méthodes rattachées à l'ACSI, OMT représente avant tout d'un standard de communication entre concepteurs qui s'accordent ici sur une vision du monde en termes d'objets. OMT se distingue par son organisation en trois modèles liés à des aspects complémentaires de l'analyse, et appelés respectivement *modèle objet, modèle dynamique,* et *modèle fonctionnel.* Toutefois, les concepts du modèle objet sont prédominants et s'introduisent au cœur même de la description des deux autres modèles. Globalement, le modèle objet prend en charge les aspects statiques de la modélisation, alors que le modèle dynamique a trait aux aspects temporels liés au contrôle, et le modèle fonctionnel aux transformations des données. Paraphrasé autrement, le modèle objet indique *sur quoi* cela se passe, le modèle dynamique *quand* cela se passe, et le modèle fonctionnel *ce* qui se passe. Mais puisque les notions développées dans le cadre des deux derniers modèles peuvent par ailleurs être appréhendées comme des objets, il devient possible de leur appliquer des manipulations caractéristiques du premier modèle.

C'est à la description détaillée des trois modèles d'OMT et de leurs interactions qu'est consacré le reste de ce document.

Première Partie

Le modèle objet

1 Introduction

Un *modèle objet* saisit la structure statique d'un système, en montrant :

- les objets dans le système;
- les **propriétés** qui caractérisent chaque **classe** d'objets;
- la nature des relations entre objets parmi lesquelles deux types de relations, **agrégation** et **généralisation**, jouent un rôle particulier.

2 Objets : instances et classes

Un objet est un concept, une abstraction, ou une chose, ayant des limites claires et un sens précis dans le contexte du problème étudié(!)... Comme cela a été souligné dans l'avant-propos, il est clair que la décomposition d'un problème en objets est un processus éminement subjectif dépendant à la fois de la nature du problème étudié et d'un présupposé sur la pertinence des choix nécessités par une représentation adéquate du problème : il n'existe *a priori* pas une "unique" et "bonne" représentation. Par ailleurs, la notion d'*objet* renvoie alternativement et suivant le cas à *une seule* chose ou à un *ensemble* caractérisant plusieurs choses. Aussi, au terme vague d'objet, il est préférable de substituer les expressions

instance d'objet pour désigner une occurence particulière d'une entité conceptualisée comme *objet* (par exemple, le téléphone posé sur mon bureau);

classe d'objets pour désigner un regroupement d'instances ayant des caractéristiques communes (attributs et opérations) et un objectif sémantique commun (par exemple, la classe des terminaux de télécommunication, incluant téléphones, télégraphes, postes TSF, talkie-walkies, etc...).

Les **propriétés** caractéristiques d'une classe sont déterminées par la nature de ses **attributs** et de ses **opérations** :

attribut *valeur de donnée* caractéristique détenue par les instances d'une classe (par exemple le *type de support* de connexion utilisé par un terminal de télécommunication : branchement à une ligne, onde hertzienne, etc...);

opération *fonction* ou *transformation* caractéristique subie ou appliquée par les instances d'une classe (par exemple, l'*établissement d'une connexion* pour la classe des terminaux de télécommunication).

Les instances d'une même classe ont le même type de comportement et les mêmes attributs. Mais bien que l'unicité de chaque instance provienne souvent de différences dans leurs valeurs d'attributs et dans leurs relations avec les autres classes d'objets, une conséquence remarquable de la propriété d'identité est qu'il est possible d'avoir des instances *distinctes* ayant *mêmes* valeurs d'attributs et de liens (par exemple deux instances de l'objet *ville* ayant toutes deux pour valeur d'attribut unique le nom "Paris" dénotent deux entités différentes dont l'une pourrait être "Paris, France" et l'autre "Paris, Texas"...).

On notera qu'une opération

- connaît sa classe;
- a un objet cible;
- peut s'appliquer à des classes différentes (par exemple, l'*établissement d'une connexion* est éventuellement une opération applicable aux conduits d'un réseau de plomberie[3]); l'implémentation d'une opération est appelée **méthode**;
- peut posséder des arguments, le nombre et le type des arguments ainsi que le type de la valeur résultat étant appelés **signature** de l'opération en question.

3 Relations entre objets

Outre la représentation des objets et de leurs propriétés, un aspect fondamental du modèle objet a trait à la nature des relations liant le cas échéant les objets entre eux. Les **liens** (respectivement les **associations**) permettent d'établir des relations entre les instances (respectivement les classes) d'un modèle objet. Plus précisément, la distinction se fait de la manière suivante :

lien connexion physique ou conceptuelle entre des instances d'objets;

association groupe de liens ayant une structure sémantique commune, et dénotant une relation entre des classes.

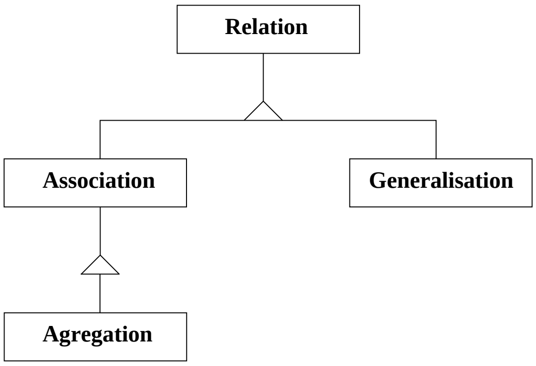
Globalement, le modèle objet fait la distinction entre trois types de relations :

- l'association générale sans caractéristique particulière;
- l'**agrégation**, qui décrit une forme particulière d'association entre un tout et ses parties, dans laquelle le tout *est composé* de ses parties;
- la **généralisation**, qui décrit une relation entre une classe et une ou plusieurs versions *raffinées* de celle-ci.

D'un point de vue strictement formel, association, agrégation et généralisation dénotent toutes les trois des relations entre classes. Toutefois, on notera que si l'agrégation est bien une forme particulière d'association, la généralisation est elle une relation dont la sémantique est différente parce que mettant en jeu des éléments d'un niveau supérieur d'abstraction. La hiérarchie de ces différents types de relations peut être

[3] pour autant qu'il y ait une représentation sémantique commune...

résumée par l'arborescence "objet" suivante[4] :



Il est revenu plus précisément sur ces distinctions dans la suite de ce chapitre.

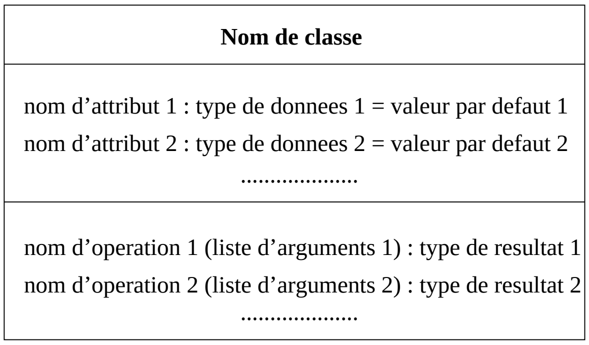
4 Diagramme d'instances et diagramme de classes

Les diagrammes d'objets proposent une notation graphique formelle qui permet de modéliser les instances d'objets, les classes, et les relations qu'elles entretiennent. Suivant le niveau d'abstraction nécessité par la modélisation, on distingue :

diagramme d'instance qui décrit comment des instances d'objets sont associées les unes aux autres;

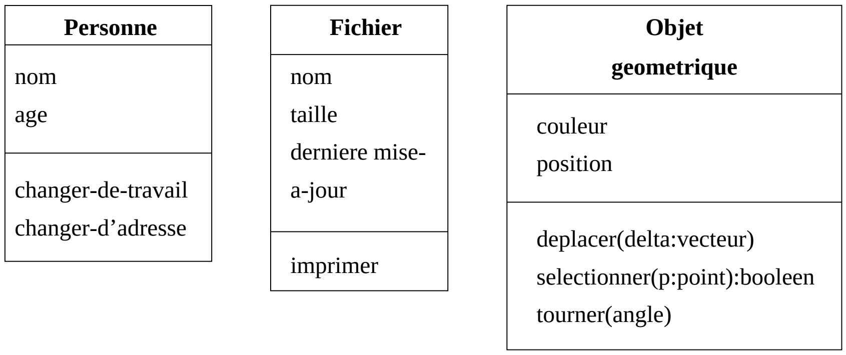
diagramme de classe qui décrit comment des classes d'objets sont associées les unes aux autres, et qui représente une abstraction d'un ensemble de diagrammes de classes.

La notation générique *complète* d'un diagramme de classe est une boite divisée en trois parties dénotant respectivement le nom de la classe (en gras), une liste d'attributs avec leur types et leur valeurs par défaut, une liste d'opérations avec leur signature :

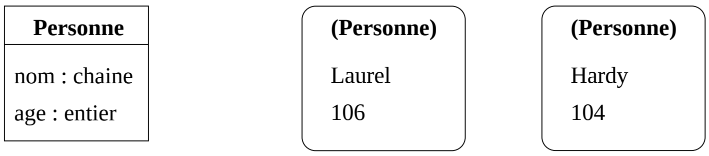


La figure suivante montre trois exemples de diagrammes de classe plus ou moins complets avec attributs et opérations :

[4] cf. les notations graphiques décrites plus loin dans ce chapitre



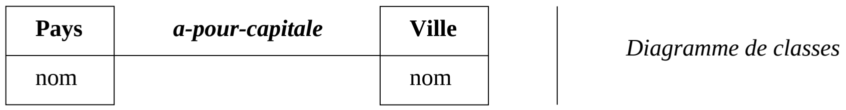
Dans la pratique, on s'autorise à ne représenter que les éléments dont on a besoin. Un diagramme d'instance est représenté par une boite arrondie contenant le nom de la classe (en gras et entre parenthèses), le nom de l'instance et les valeurs éventuelles d'attributs au-dessous. La figure ci-dessous montre un diagramme de classe (à gauche) et un possible diagramme d'instances (à droite) décrit par le diagramme de classe.

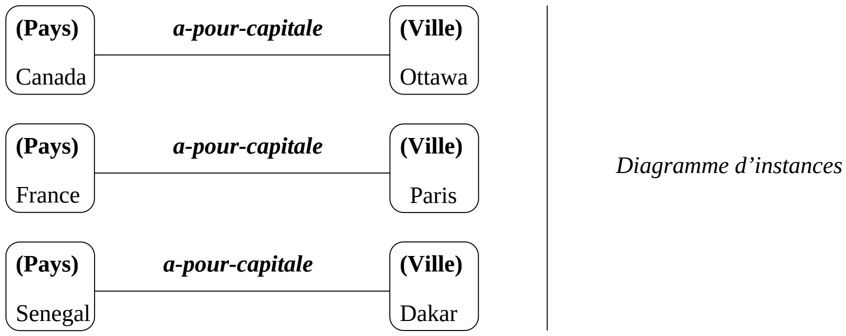


Classe avec attributs *Instances avec valeurs*

Les diagrammes de classe décrivent le cas général de la modélisation d'un système. Les diagrammes d'instance sont principalement utilisés pour montrer des exemples destinés à clarifier les diagrammes de classe complexes. Classes et instances peuvent apparaître sur le même diagramme d'objets, mais en général il n'est pas utile de les mélanger.

La représentation d'une association de type général (au niveau classes) et des liens (au niveau instances) est faite sous la forme d'une ligne reliant les objets concernés. Par exemple :





5 Liens et associations : concepts évolués

L'emploi brut de liens ou d'associations se trouve raffiné par la possibilité de représenter explicitement sur un diagramme les notions :

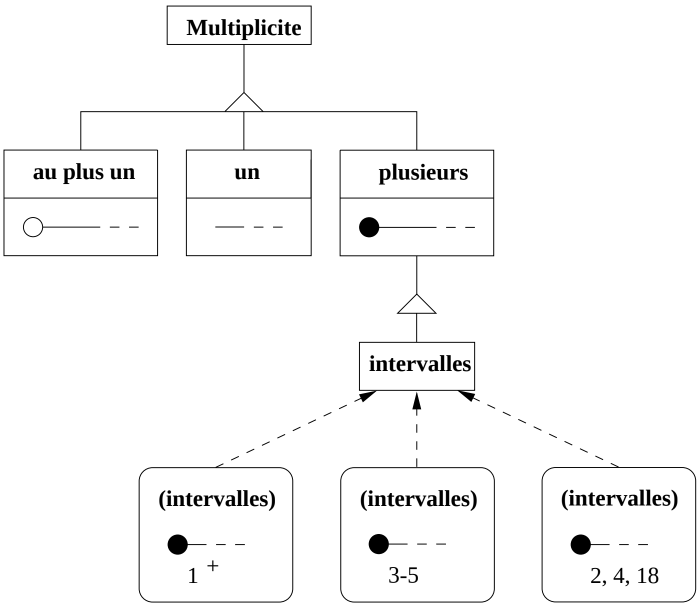
- de multiplicité;
- de noms de rôle;
- de qualifieur;
- d'attributs de liens;
- de contraintes;
- d'entité dérivée;
- d'homomorphisme entre associations.

5.1 Multiplicité

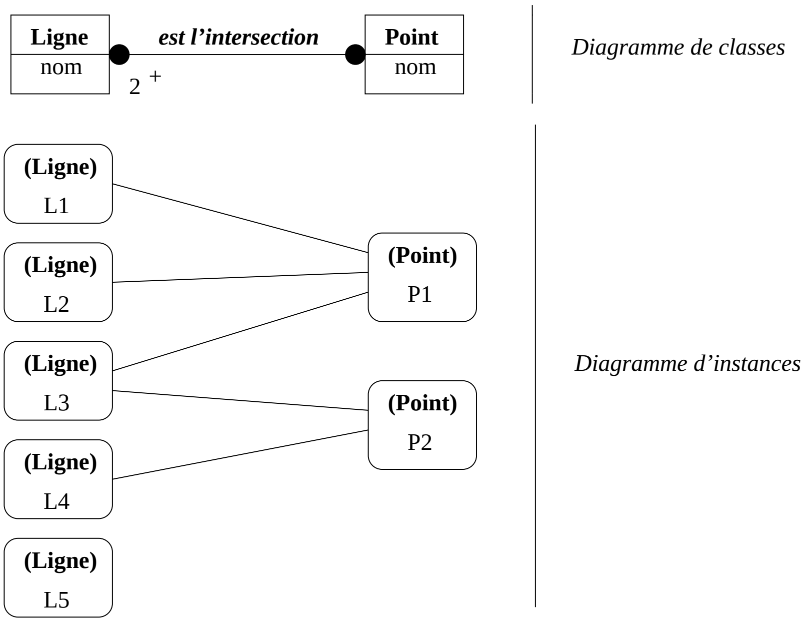
La **multiplicité** précise combien d'instances d'une classe peuvent se rattacher à une seule d'une classe donnée. On distingue essentiellement trois types de multiplicités : "au plus un", "un", et "plusieurs". Ce dernier type est lui même affiné par l'adjonction de symboles numériques et l'utilisation d'intervalles, par exemple "1^+" (un ou plus), "3-5" (trois à cinq, inclus), "2, 4, 18" (deux, quatre, ou dix-huit)[5].

[5]Clairement, le terme "intervalle" utilisé par les promoteurs de la méthode résulte ici d'un véritable abus de langage et devrait être remplacé *stricto sensus* par "unions d'intervalles", à quoi fait visiblement référence l'union de valeurs discrètes notée "2, 4, 18"(union de singletons), ou encore (pourquoi pas) "2, 3-5, 18". Nous suivrons néanmoins la terminologie d'origine (mal) choisie par les auteurs

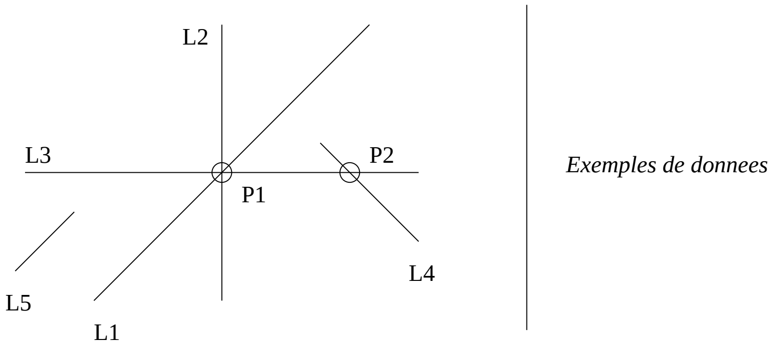
Les possibilités de spécification de la multiplicité d'une association peuvent être résumées par l'arborescence "objet" suivante[6] :



On notera que la multiplicité concerne bien une association et non un lien, ce qu'illustre l'exemple géométrique simple suivant : on y considère une classe de lignes, une classe de points et une association *est l'intersection de* de multiplicité supérieure ou égale à deux (notée 2^+) entre ces deux classes.



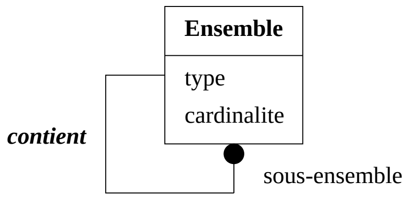
[6]cf. les notations graphiques décrites plus loin dans ce chapitre



Le diagramme de classes y apparaît clairement pour ce qu'il est : la représentation synthétique d'un ensemble (ici potentiellement infini) de situations particulières.

5.2 Nom de rôle

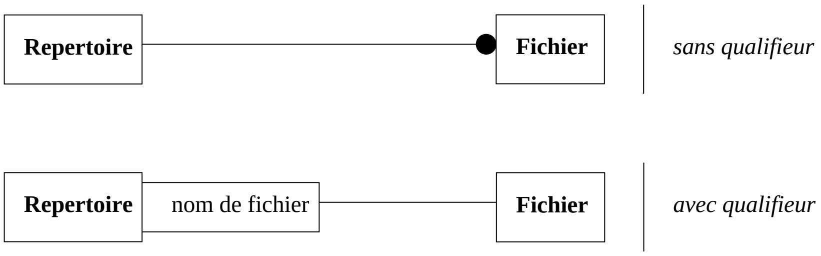
Un **nom de rôle** identifie de façon unique une des deux extrémités d'une association. Il est écrit sur le diagramme près de la ligne de l'association, à côté de l'objet qui joue le rôle. L'utilisation des noms de rôle est en général facultative mais elle permet de préciser le sens d'une association. Elle est nécessaire pour des associations entre deux objets d'une même classe. Par exemple *sous-ensemble* permet d'identifier la fonction assurée par la classe *ensemble* à l'extrémité concernée de l'association *contient* du diagramme suivant :



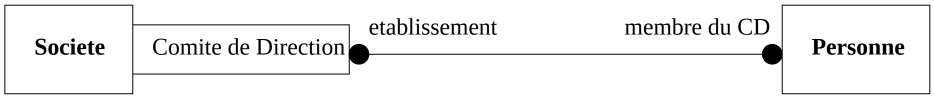
5.3 Qualifieur

Un **qualifieur** est un attribut spécial qui réduit généralement la multiplicité effective d'une association. Les associations dont la multiplicité est de type "plusieurs" à l'une de leur extrémité peuvent être qualifiées : le qualificatif caractérise *l'ensemble d'objets* à l'extrémité "plusieurs"[7]. Un qualificatif est dessiné comme une petite boîte près de la classe qu'il qualifie. Par exemple, on considère le cas d'un système d'exploitation dans lequel un fichier ne peut appartenir qu'à un seul répertoire. Dans le contexte d'un répertoire, le nom de fichier spécifie un fichier unique :

[7] à ne pas confondre avec un nom de rôle qui, lui, permet de préciser la *fonction* assurée par une classe à une extrémité de l'association



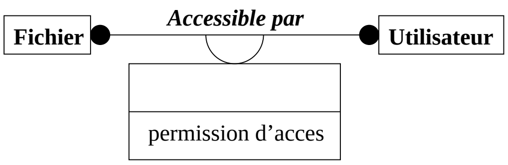
La qualification réduit ici la multiplicité effective de l'association de "un-à-plusieurs" à "un-à-un". Toutefois la réduction de la multiplicité ne s'opère pas toujours. Par exemple, dans l'exemple de qualification "plusieurs-à-plusieurs" suivant, la qualification effectue la partition d'un ensemble d'objets reliés en des sous-ensembles disjoints, mais les sous-ensembles peuvent contenir plus d'un objet (puisque qu'un comité de direction compte plusieurs membres).



EDF	President	Dubois
EDF	Directeur	Dupont
EDF	Directeur	Durand
France Telecom	President	Dupont
France Telecom	Directeur	Duvert

5.4 Attribut de lien

Un **attribut de lien** est une propriété des liens d'une association. Chaque attribut de lien possède une valeur pour chaque lien. La notation OMT pour un attribut de lien est une boîte attachée à l'association par une boucle. Un ou plusieurs attributs de liens peuvent apparaître dans la deuxième partie de la boîte. Par exemple :

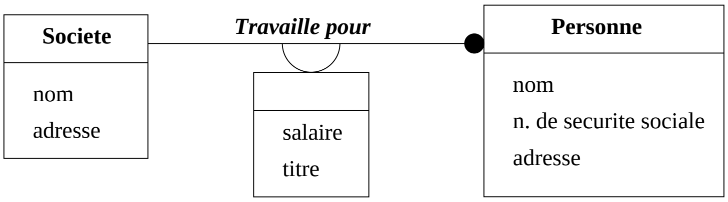


/etc/termcap	(lire)	Dupont
/etc/termcap	(lire-ecrire)	Durand
/usr/dubois/.login	(lire-ecrire)	Dubois

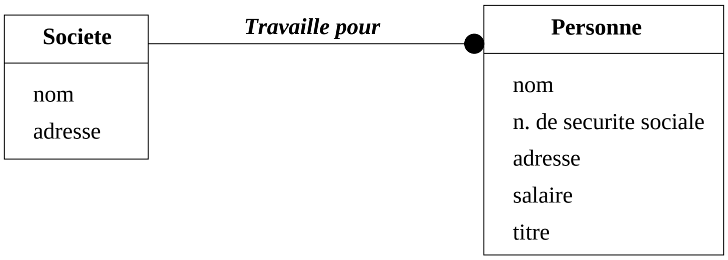
On peut noter que :

- Un attribut de lien représente une propriété conjointe des classes en association, et qui ne peut *a priori* être rattachée à l'une de ces classes sans perte d'information. En ce sens les associations de type "plusieurs-à-plusieurs" sont la justification la plus contraignante de l'existence des attributs de liens, comme l'illustre l'exemple précédent.

- Dans le cas d'une association de multiplicité "un" à une au moins de ses extrémités, il est toutefois possible de ranger sans perte les attributs de lien dans la classe opposée au "un". Néanmoins, la souplesse d'évolution est réduite si les multiplicités de l'association viennent à changer. Ainsi, dans l'exemple suivant, seule la forme ayant un attribut de lien demeure correcte si la multiplicité de *travaille pour* est changée en "plusieurs-à-plusieurs".

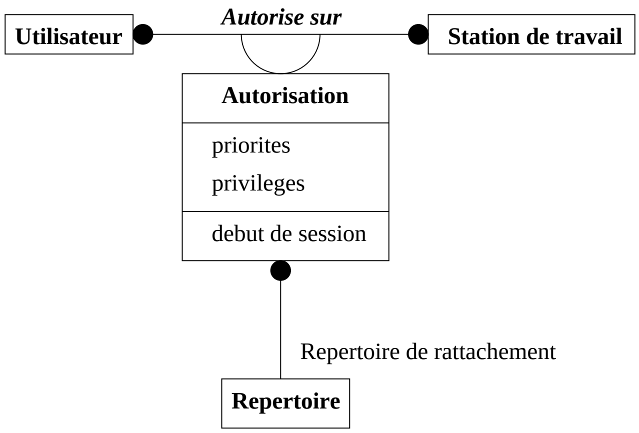


Forme avec attributs de lien



Forme sans attributs de lien

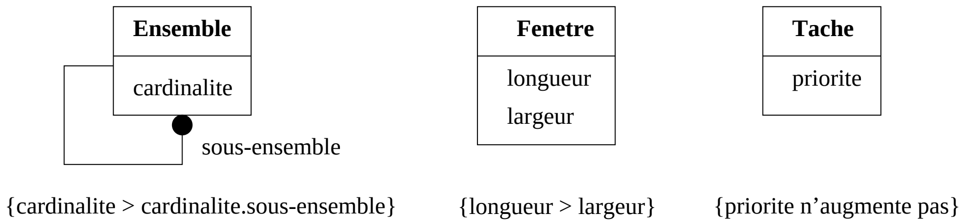
- La notion d'attribut de lien ouvre la voie à une extension plus générale, qui autorise la modélisation d'une association en classe. Chaque lien est alors une instance de la classe. La boîte de l'attribut de lien est un cas spécial de la modélisation d'une association en classe et peut avoir un nom et des opérations en plus des attributs.



Comme on peut s'en rendre compte à partir de l'exemple précédent, il est utile de modéliser une association comme une classe quand des liens peuvent participer à des associations avec d'autres objets ou quand des liens peuvent être sujets à des opérations.

5.5 Contraintes

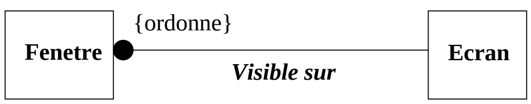
Les **contraintes** sont des relations fonctionnelles portant sur les entités d'un modèle objet (i.e. classes, attributs, liens ou associations). Le rôle d'une contrainte est de restreindre l'ensemble sur lequel les entités peuvent prendre leurs valeurs. Elle prend la forme d'une expression placée entre accolades près des entités sur lesquelles elle porte.



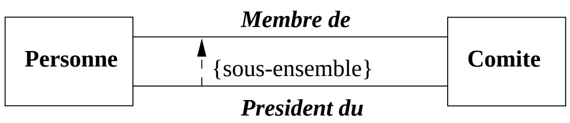
- Les contraintes fournissent un critère d'évaluation de la qualité d'un modèle; un "bon" modèle objet traduit généralement beaucoup de contraintes dans sa structure.

- La multiplicité est un type particulier de contrainte sur une association, car elle permet de restreindre le nombre d'objets reliés à un objet donné.

- Un autre type particulier de contrainte sur association est l'**ordre** sur les objets du côté "plusieurs" d'une association. Dans le cas général, ces objets n'ont pas d'ordre explicite et peuvent être considérés comme des ensembles. Parfois cependant, ils sont explicitement ordonnés, comme dans l'exemple suivant, qui tient compte de l'empilement des fenêtres sur l'écran d'une station de travail :

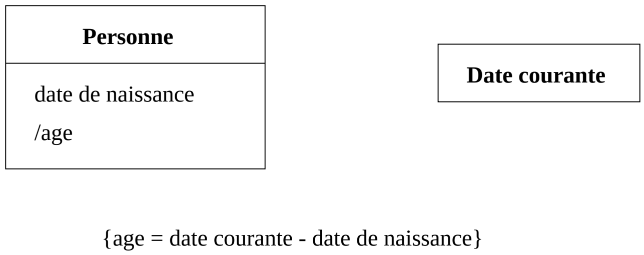


- Les contraintes générales dénotent non pas des contraintes *sur* des entités mais *entre* des entités du modèle objet. Outre la contrainte entre accolades, une ligne pointillée relie les classes concernées par la contrainte. Par exemple :



5.6 Objet dérivé

Un *objet dérivé* (précisément ici une *classe dérivée*) est défini comme une fonction d'un ou plusieurs objets qui peuvent eux-même être dérivés par ailleurs. L'arbre de dérivation se termine par des objets de base. L'objet dérivé est complètement déterminé par les autres objets. La notion de dérivation s'étend aux liens et aux attributs sous la forme de *liens dérivés* et *attributs dérivés.* Une entité dérivée est identifié par une barre oblique dans le coin de la boîte de la classe, sur la ligne d'association, ou en face des attributs). La contrainte qui détermine l'entité dérivée est montrée entre accolades. Par exemple :



Il est clair qu'un objet dérivé est redondant, mais son inclusion dans un diagramme objet d'une part facilite la compréhension du modèle, d'autre part permet des optimisations de ce modèle.

5.7 Homomorphisme entre associations

Un homomorphisme est une *représentation* d'une association vers une autre. En quelque sorte, l'homomorphisme permet d'établir une relation de type analogique entre deux associations. Il s'agit d'une notion particulièrement intéressante dans le cas d'applications manipulant des méta-données. Nous ne nous étendons pas plus avant sur les homomorphismes dans le cadre de ce polycopié, et renvoyons le lecteur interessé à la section 4.7.5. de [Rumbaugh et Al., 91].

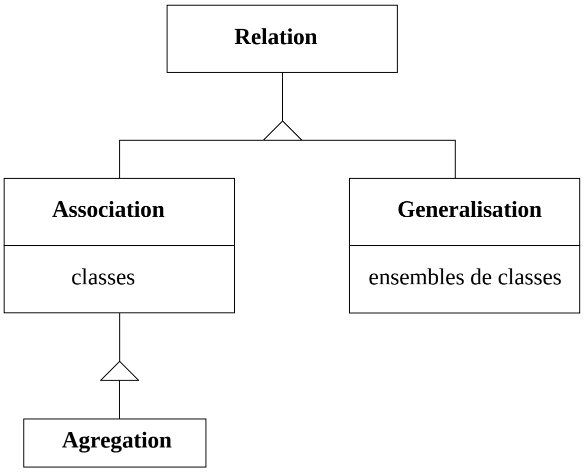
6 Agrégation et Généralisation

Agrégation et **Généralisation** sont deux formes particulières de relation qui apparaîssent de fait comme des composantes fondamentales d'une démarche objet. L'agrégation décrit une relation de composition entre un tout et ses parties (par exemple, un téléphone est composé d'un clavier, d'un combiné, d'un haut-parleur, et d'un dispositif électronique assurant la connexion à une ligne) alors que la généralisation décrit une relation entre une classe et des versions raffinées de celle-ci (par exemple, la classe des appareils de télécommunication est une généralisation de la classe des téléphones, la classe des talkie-walkies, la classe des appareils TSF...). Bien que formellement toutes deux des relations -et c'est pourquoi nous les traitons ici ensemble- agrégation et généralisation se distinguent par une différence sémantique fondamentale :

- l'agrégation porte sur des *ensembles d'instances* (autrement dit des *classes...*), et par là peut être simplement considérée comme un type particulier d'association;

- la généralisation porte sur des *ensembles de classes* (autrement dit sur des entités issues d'un niveau d'abstraction supplémentaire), et doit donc être vue comme une sorte de *méta*-association.

Ainsi la figure présentée au paragraphe 3 peut-être affinée de la façon suivante :



6.1 Agrégation

L'agrégation est une relation "composé-composant" ou "partie-de" dans laquelle les objets représentant les composants d'une chose sont associés à un objet représentant l'assemblage entier. L'agrégation est une forme d'association finement couplée avec une sémantique additionnelle :

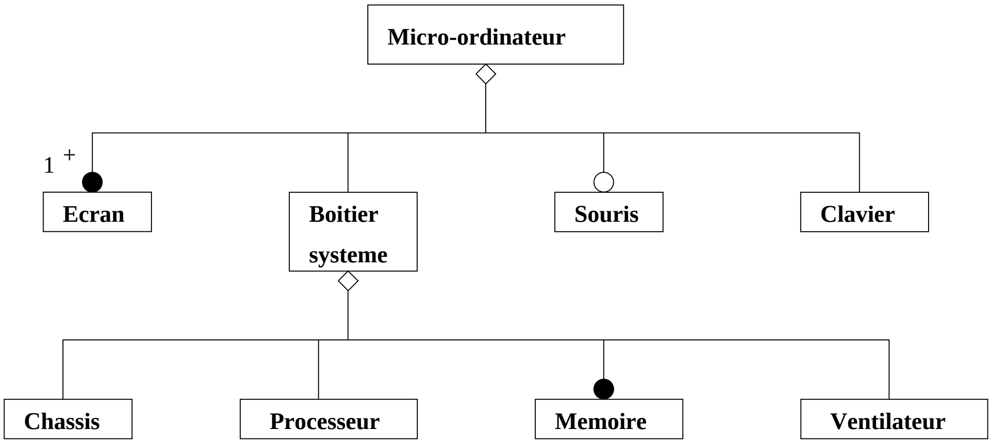
- La propriété la plus significative de l'agrégation est la *transitivité.* Par exemple, si A est une partie de B, B une partie de C, alors A est une partie de C.

- L'agrégation est *asymétrique,* c'est-à-dire que si A est une partie de B, B n'est pas une partie A.

- Enfin, les propriétés de l'assemblage *se propagent* vers les composants avec d'éventuelles modifications. Par exemple, la vitesse et la position d'une poignée de porte de voiture sont obtenues à partir de la porte dont elle est une partie; la porte à son tour reçoit ses propriétés de la voiture dont elle est une partie.

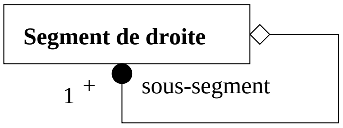
L'agrégation est dessinée comme une association à l'exception d'un losange qui indique l'extrémité assemblage de la relation. Par exemple, la figure suivante décrit un fragment de modèle objet d'un traitement de texte. Un document consiste en un ensemble de paragraphes, chacun consistant en un ensemble de phrases.

La figure suivante illustre un autre exemple d'agrégation. Un micro-ordinateur est composé d'un ou plusieurs moniteurs, d'un boîtier système, d'une souris facultative, et d'un clavier. Un boîtier système possède à son tour un châssis, une unité centrale, plusieurs puces de RAM et un ventilateur.

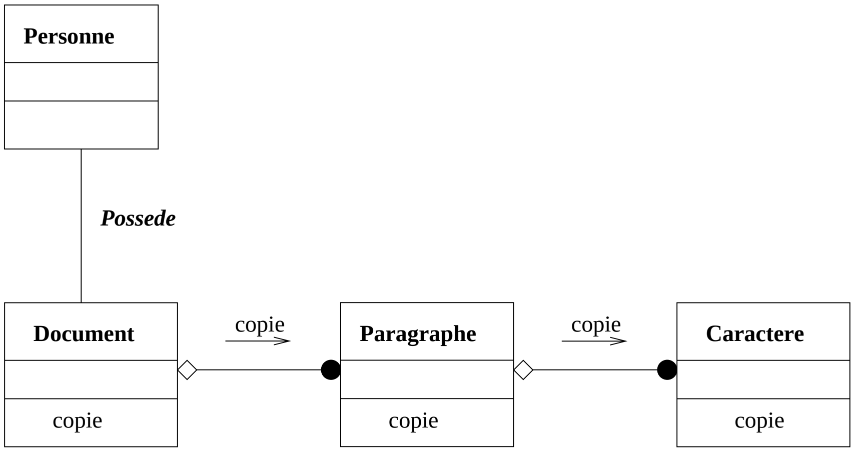


On peut noter que :

- Une classe composante peut être vide (par exemple, la classe souris dans l'exemple précédent).
- La présence des composants de la classe agrégative présente un caractère *nécessaire.*
- Une agrégation est une relation de type "et" (par exemple un boîtier système est fait d'un chassis *et* d'une unité centrale *et* de plusieurs puces de RAM *et* d'un ventilateur éventuel).
- Un agrégat peut être construit de façon *récursive;* un agrégat récursif contient directement ou indirectement une instance de la même sorte d'agrégat. Par exemple, un segment de droite est directement composé de segments de droite :



- L'agrégation permet d'assurer la *propagation* des opérations de la classe agrégative vers les classes composantes. Par exemple, bouger un agrégat bouge ses parties. La propagation des opérations aux parties est souvent un bon indicateur d'agrégation. La propagation est signalé de façon spéciale sur le diagramme. Le comportement de propagation étant lié à une agrégation, une direction, et une opération, on l'indique par une flèche et un nom d'opération près de l'association affectée; la flèche indique la direction de la propagation. Dans l'exemple suivant, une personne possède plusieurs documents. Chaque document est composé de paragraphes à leur tour composé de caractères. L'opération *copier* se propage du document aux paragraphes, puis aux caractères. L'opération ne se propage pas à rebours :

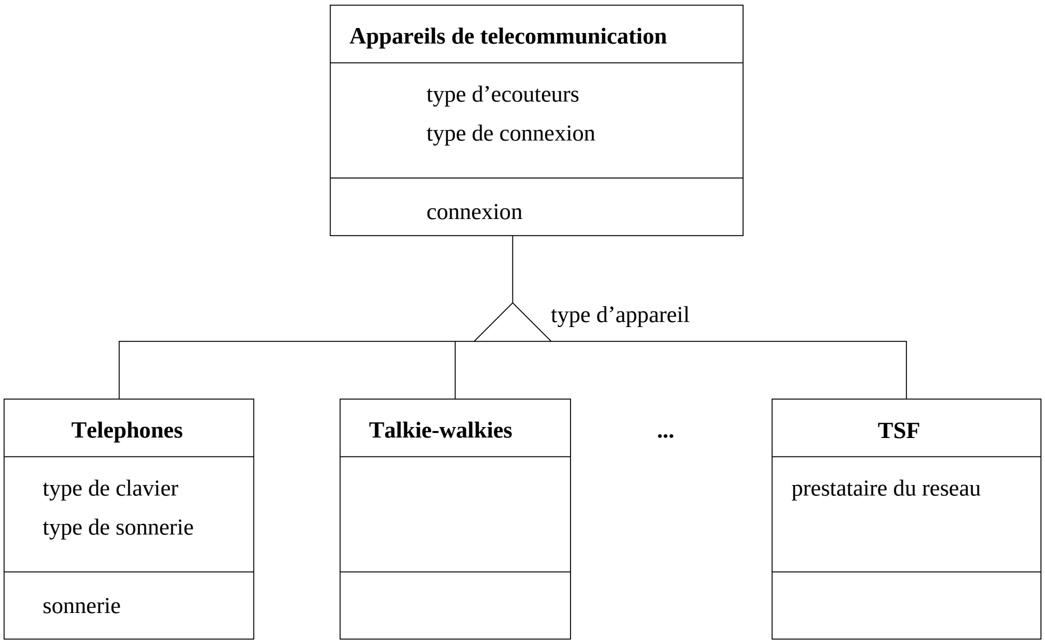


6.2 Généralisation

La généralisation est une relation de type "est un" ou "est une sorte de" entre une classe et une ou plusieurs versions affinées de cette classe. La classe de généralisation est appelée *super-classe* ou *classe-ancêtre* et chaque version affinée est appelée *sous-classe* ou *descendant.* L'**héritage** (autrement dit le partage) des propriétés de la super-classe par les sous-classes est un mécanisme intrinsèquement lié à la généralisation. En plus, chaque sous-classe possède aussi ses propres attributs et opérations spécifiques.

Généralisation et héritage sont transitifs à travers un nombre arbitraire de niveaux : une instance d'une sous-classe est simultanément une instance de toutes ses super-classes et toute opération sur toute super-classe est applicable aux instances.

La notation pour la généralisation est un triangle reliant une super-classe à ses sous-classes. Un exemple (pas très original) de généralisation est donné ci-dessous. La classe des appareils de télécommunication y est vue comme une généralisation de classes d'appareils plus spécialisés. Par exemple un appareil spécialisé de type téléphone "est une sorte d'"appareil de télécommunication. La classe téléphone hérite des attributs *type d'écouteur, type de connexion* et de l'opération *connexion* de sa super classe :



Les trois points de suspension indiquent qu'il y a des sous-classes supplémentaires qui ne sont montrées sur le diagramme. L'étiquette *type d'appareil* accolée au triangle de généralisation est un **discriminant**. Un discriminant est un attribut qui indique que la propriété d'un objet a été rendue abstraite par une relation de généralisation particulière.

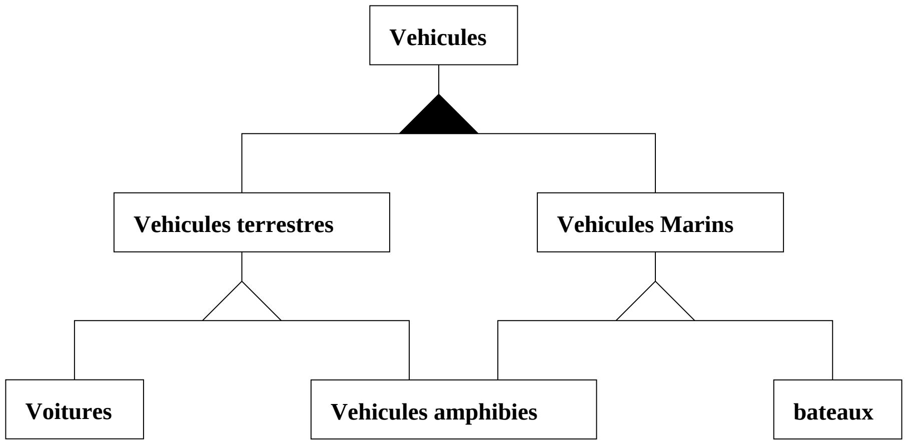
On peut noter que :

- Une sous-classe peut être vide, mais de par la nature même de la relation de généralisation (relation de type "est une sorte de") la présence de symboles de multiplicité n'a pas de sens ici.

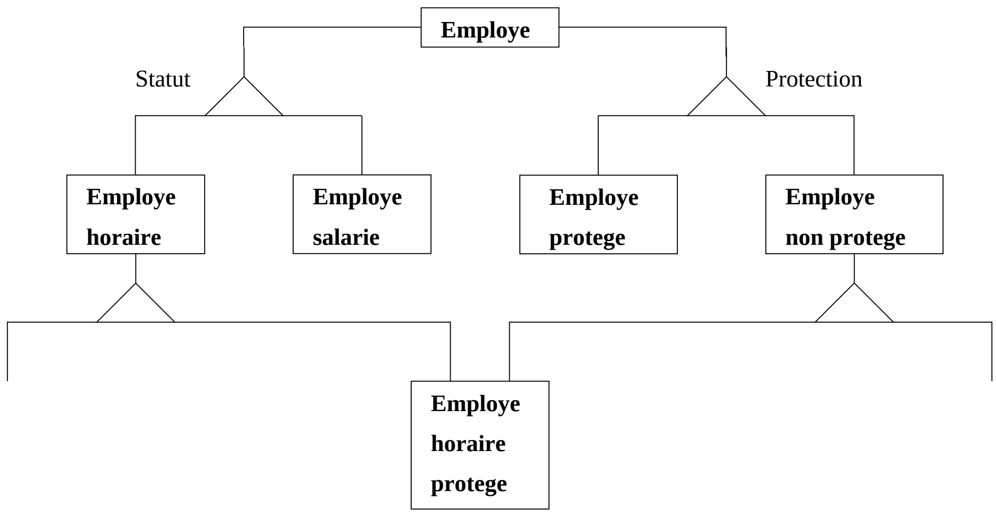
- Pour la même raison, l'énumération des sous-classes prend un caractère *suffisant* par rapport à la super-classe associée.

- Une généralisation est une relation de type "ou" (par exemple un appareil de télécommunication un téléphone *ou* un talkie-walkie *ou* un appareil de TSF...).

- Les sous-classes d'une généralisation peuvent être non disjointes. Dans ce cas, on parle de *généralisation avec recouvrement.* Par exemple, un *Véhicule amphibie* est à la fois un *Véhicule marin* et un *Véhicule terrestre.* Un triangle vide indique des sous-classes disjointes, un triangle plein indique des sous-classes se chevauchant :



- La description du mécanisme d'héritage prend une forme plus sophistiquée lorqu'il y a **héritage multiple**. L'héritage multiple permet à une classe, dite **classe de jointure** d'avoir plus d'une super-classe et d'hériter des propriétés de tous ses parents. Dans l'exemple précédent, la classe *Véhicules amphibies* hérite à la fois des classes *Véhicules terrestres* et *Véhicules marins.* Il s'agit d'un héritage multiple avec recouvrement. Dans l'exemple suivant, la classe *Employé horaire protégé* dénote un cas d'héritage multiple sans recouvrement :



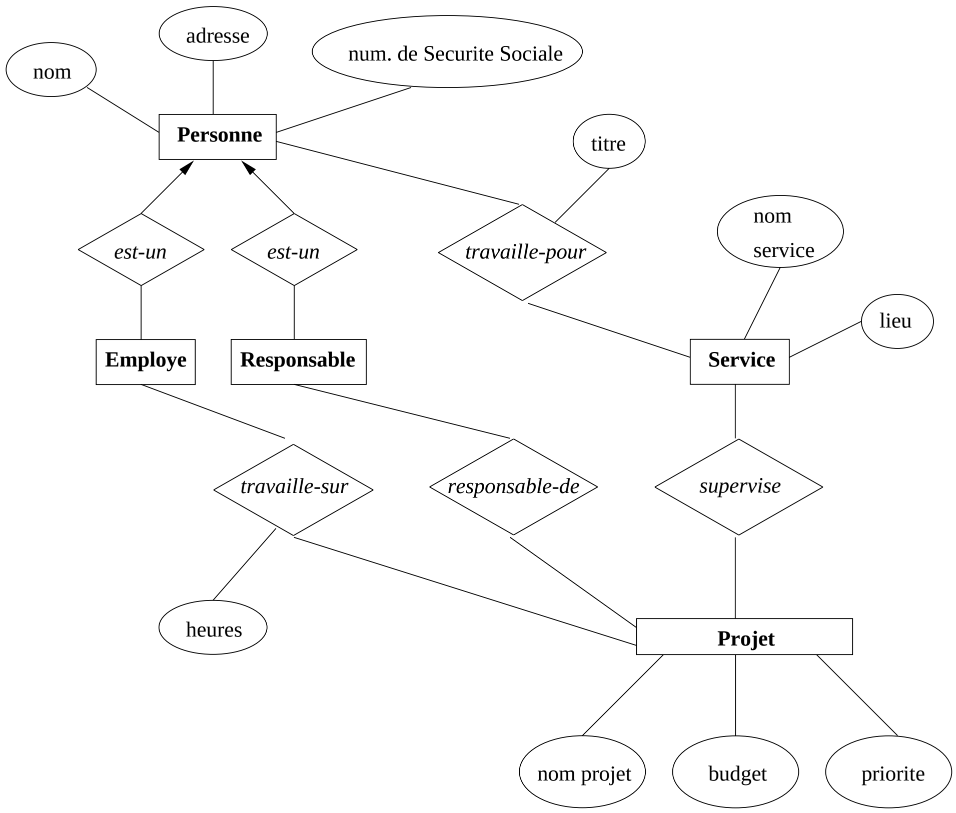
Une classe peut hériter de deux généralisations distinctes ou de différentes classes à l'intérieur d'une généralisation avec recouvrement, mais jamais de deux classes d'une même généralisation disjointe.

6.3 Résumé comparatif de chacun des deux types de relation

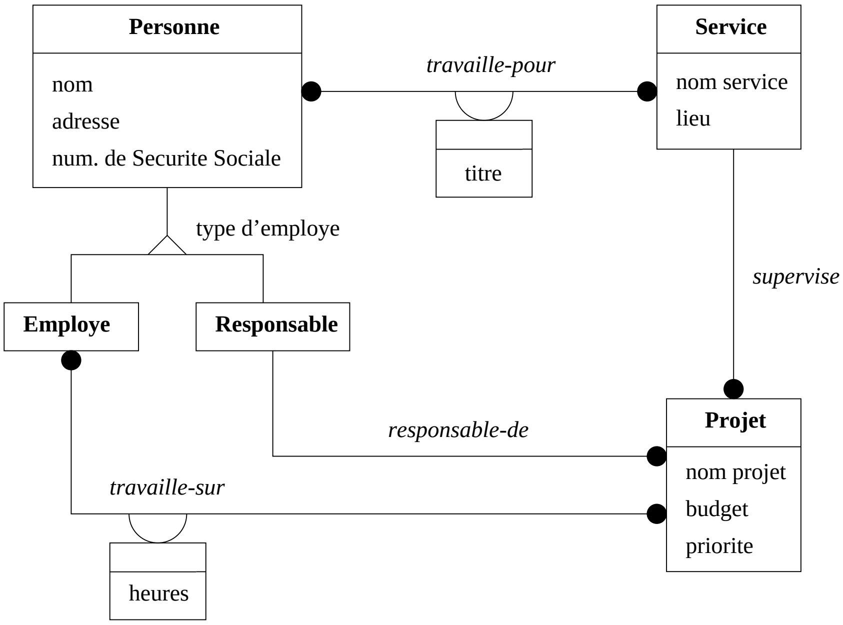
Agrégation	Généralisation
met en relation des *instances*	met en relation les *classes*
caractère *nécessaire* des composants	caractère *suffisant* des composants
relation de type “et”	relation de type “ou”
possibilité de construction *récursive*	
propagation des opérations	*héritage* des propriétés

7 Modèles Objet et Entité-association

Le lecteur averti aura constaté bien des similitudes entre le modèle objet qui vient d’être présenté et le modèle entité-association introduit par [Chen, 76]. Ces similitudes ne sont pas fortuites, et la traduction d’un modèle objet vers un modèle entité-association est quasi-immédiate comme l’illustre l’exemple ci-dessous (également inspiré de [Rumbaugh et Al., 91] et [Elmasri, Navathe, 94]). Au modèle Entité-association :



est associé le modèle objet :



La possibilité de concevoir une base de données relationnelle à partir d'un modèle objets est une conséquence immédiate de cette similitude. Nous renvoyons au chapitre 17 de [Rumbaugh et Al., 91] pour plus de détails.

On notera que :

- Le modèle entité-association permet d'intégrer la notion d'attribut composé (par exemple *adresse* ci-dessus) et de distinguer les attributs multivalués.
- Les connaisseurs ne manqueront pas de remarquer l'analogie forte entre la notation d'un MCD en MERISE et celle du modèle objet correspondant.
- Il est clair pourtant que la similitude des deux types de représentations masque des approches sémantiques différentes :
  - par exemple la présence d'une super-classe *Personne* est inimaginable en entité-association tant que l'existence de cette super-classe n'est pas motivée par une relation la liant avec une autre classe, ici la classe *Service*. Inversement la sémantique sous-jacente à la généralisation encourage la conceptualisation de la super-classe en question, indépendamment de la présence ou non d'une relation la liant à une autre classe.
  - Dans le même ordre d'idée, la notion d'attribut composé en entité-association serait *a priori* représentée en objet comme une classe agrégative à part entière.

Deuxième partie

Le modèle dynamique

8 Introduction

Un *modèle dynamique* décrit les modifications des objets et de leurs relations dans le temps, en montrant :

- les différents **états** que peuvent prendre ces objets;
- les **événements** auquels ils sont soumis.

9 Etats et événements

L'**état** d'un objet est défini par les valeurs de ses attributs et de ses liens à un *moment* donné. Un **événement** est un stimulus reçu ou envoyé par un objet à un *instant* donné. Il s'agit d'une voie de transmission d'information à sens unique d'un objet vers un autre. La réponse à un événement dépend de l'état de l'objet le recevant : il peut changer d'état et/ou générer lui-même un événement à destination d'un autre objet. De fait, l'adoption d'une échelle ou d'une autre pour décrire le temps induit une caractérisation complète de la distinction entre état et événement : alors qu'un état s'inscrit dans la *durée,* un événement est *instantané* eu égard à la granularité adoptée pour la description. Cette distinction s'étend naturellement à la la façon dont les objets interagissent les uns avec les autres : si un état est associé à la description d'une **activité** possible d'un objet, un événement est associé à une **action** éventuelle entreprise par un objet dans son environnement.

De cette distinction encore, il apparaît clairement qu'états et événements sont des notions parfaitement duales : prosaiquement, un événement est ce qui sépare deux états et un état, ce qui sépare deux événements...

	Etat	Evénement
Temps	dans la *durée*	*instantané*
Opérations	*activité*	*action*

On notera pour finir que chaque événement, à l'instant où il se produit, est une occurence unique, mais que les événements ayant mêmes structure et comportements se regroupent en classes hiérarchisées. Par exemple, les événements *emprunt du livre cote ISBN 0-8053-1748-1 par le lecteur 1263* et *emprunt du livre cote ISBN 2-225-84684-7 par le lecteur 1512* sont deux instances de la classe d'événements *emprunt d'un livre* ayant cote et numéro de lecteur pour attributs représentant l'information transmise par cette classe.

A partir de la description "à plat" d'une classe objet, la question est de dégager de façon pertinente les états et événements associés à cette classe : c'est le rôle des scénarios[8] et du diagramme de suivi d'événements.

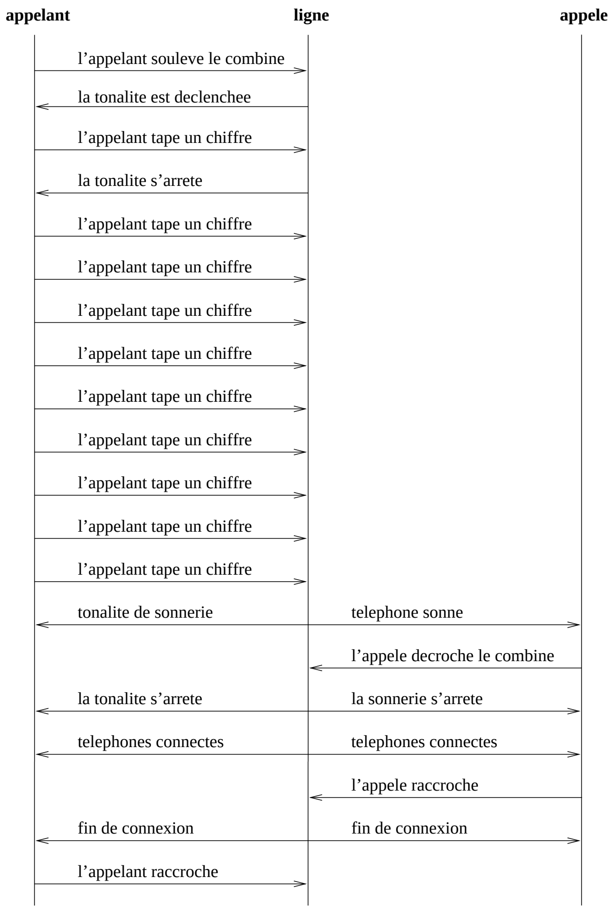
[8]ou *scenarii* pour les puristes...

10 Scénarios et suivis d'événements

Un **scénario** est une séquence particulière d'événements décrivant une parmi les exécutions possibles du système d'objets. L'exemple suivant est un scénario illustrant l'utilisation d'une ligne de téléphone :

l'appellant soulève le combiné
la tonalité est déclenchée
l'appelant tape un chiffre
la tonalité s'arrête
l'appelant tape un chiffre
l'appelant tape un chiffre
l'appelant tape un chiffre
l'appelant tape un chiffre
l'appelant tape un chiffre
l'appelant tape un chiffre
l'appelant tape un chiffre
l'appelant tape un chiffre
l'appelant tape un chiffre
le téléphone appelé commence à sonner
la tonalité de sonnerie commence dans le téléphone appelant
l'appelé répond
le téléphone de l'appelé cesse de sonner
la tonalité de sonnerie cesse dans le téléphone appelant
les téléphones sont connectés
l'appelé racroche
les téléphones sont déconnectés
l'appelant racroche

Un scénario est donc une des histoires possibles pouvant survenir du point du vue d'*un* objet, l'idée étant de commencer par dégager les scénarios correspondant aux fonctions les plus "normales" ou "attendues" de cet objet avant de considérer les cas d'utilisation limite. A partir d'un ensemble représentatif de scénarios, le diagramme de suivi d'événements a pour but de situer les objets et d'identifier leurs états par rapport aux événements. Le diagramme de suivi d'événements de la ligne téléphonique prise en exemple plus haut est :

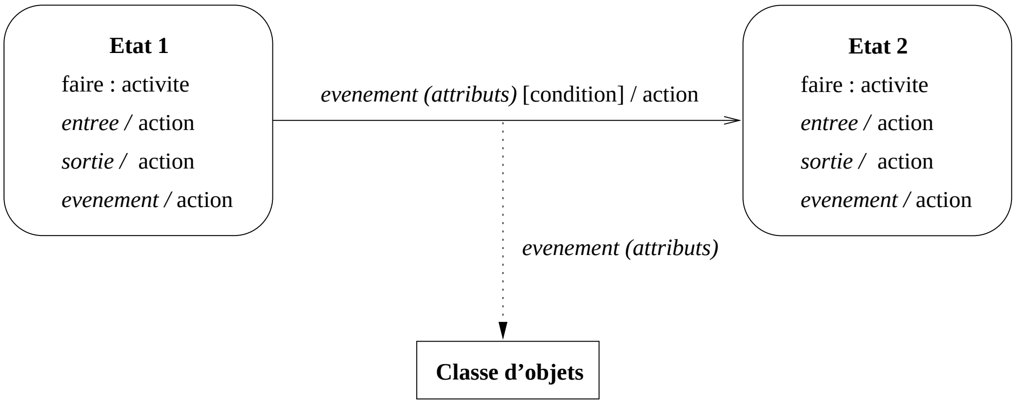


L'extension d'un diagramme de suivi d'événements à la prise en compte de toutes les situations possibles est l'objet du diagramme d'états.

11 Diagramme d'états

Le rôle d'un **diagramme d'états** est la description de l'ensemble des états et des événements qui s'appliquent à *une* classe d'objets. Le formalisme adopté par les concepteurs d'OMT s'inspire directement d'un sous-ensemble simple des *automates cellulaires*[9]. Un diagramme d'états est un graphe orienté dont les noeuds sont des états et les arcs des transitions entre états désignées par les noms d'événements. Un état est une boîte aux angles arrondis et contenant un nom optionnel. Une transition entre états est représentée par une flèche de l'état receveur vers l'état cible, l'étiquette désignant la flèche étant celui de l'événement associé. Une séquence d'événements est donc un chemin dans le graphe. Précisément les notations adoptées sont les suivantes :

[9]encore appelés ailleurs *automates à états finis*, parmi lesquels notre préférence va sans hésitation aux *automates à la provencale...* Sur la théorie des automates cellulaires, et parmi l'abondante littérature sur sujet, le lecteur interessé peut consulter [Hopcroft, Ullman, 79] ou [Salomaa, 85]

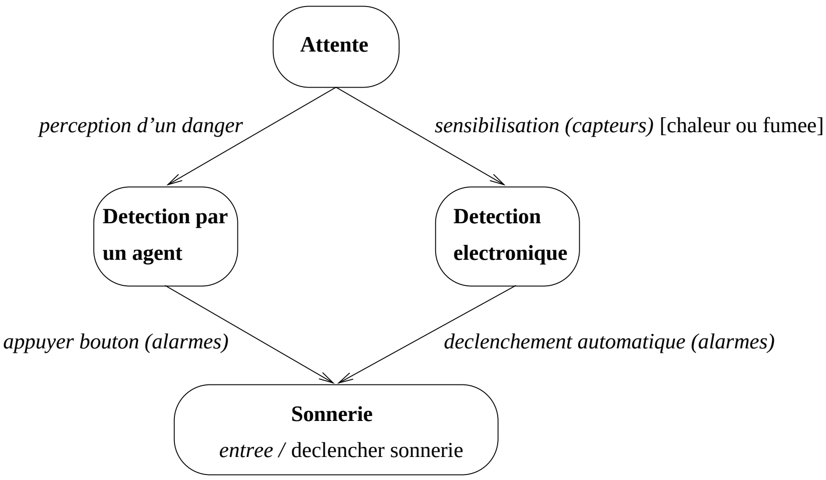


On notera que :

- La transition d'un état vers un autre peut être envisagée lorsque c'est nécessaire comme associée à la manipulation d'attributs qui interviennent dans le déroulement de l'événement.

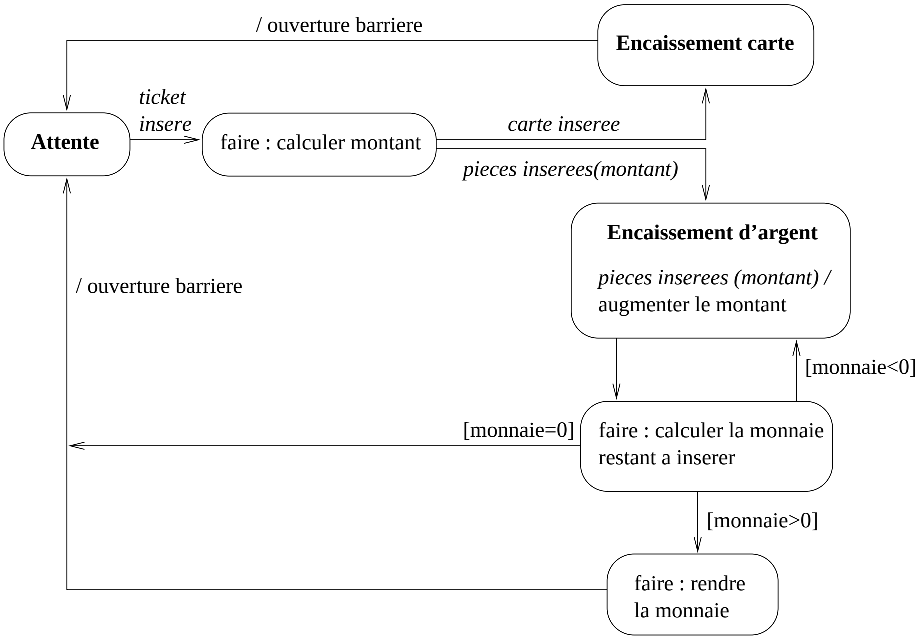
- Une *condition* est une fonction booléenne contrôlant le déclenchement d'un événement. On notera qu'une condition peut-être valide durant un certain laps de temps à la différence des événements qui sont sans durée. Les conditions peuvent alors être utilisées comme *gardes* sur les transitions. Une transition gardée est franchie quand survient un événement, seulement si la condition de garde est vérifiée. Par ailleurs, une condition étant, tout comme un état, associée à une durée il est raisonable d'envisager la définition d'un état précisément comme ne représentant rien d'autre que la validation d'une ou plusieurs conditions; réciproquement, être dans un état est une condition.

- Une *action* est associée soit à un événement, soit aux entrée-sorties d'un état, et notée après une barre oblique. Il s'agit, on l'a dit, d'une opération intantanée. Lorsque toutes les transitions arrivant ou partant d'un état exécutent la même action, il est plus concis de rattacher l'exécution de cette action à l'*entrée* ou la *sortie* de l'état en question. Ainsi, dans l'exemple suivant un système d'alarme incendie peut être mobilisé soit par un détecteur électronique, soit par un agent humain. Dans les deux cas le système en question déclenche une sonnerie d'alerte :



- L'*activité* associée à un état est indiquée après la notation "faire : " à l'intérieur de la boîte d'état. Cette activité débute en entrant dans l'état et se termine à sa sortie.

- Un événement peut provoquer l'exécution d'une action sans provoquer de changement d'état. Un tel événement est *interne* à l'état. On notera la distinction entre une action interne à un état et une transition réflexive provoquant l'exécution de la même action : dans ce dernier cas l'événement provoque l'exécution des actions d'entrée et de sortie pour l'état, ce qui n'est pas le cas si l'événement est interne à l'état. Le diagramme suivant modélise le fonctionnement d'un guichet automatique autoroutier. L'état "Encaissement d'argent" comporte un événement et une action associée, internes à cet état :



L'utilisation d'un événement interne trouve ici au moins deux justifications :

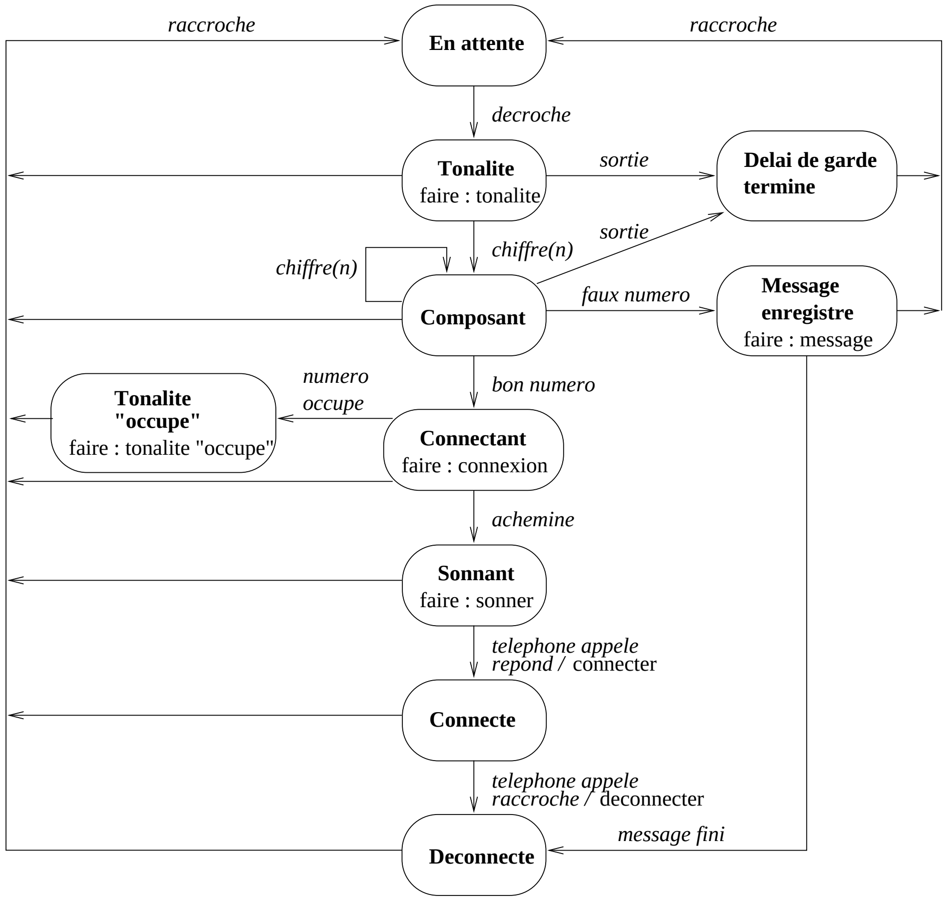
– comme cela vient d'être signalé, il s'agit d'éviter le déclenchement indésirable d'actions éventuelles liées aux entrées-sorties; mais en outre

– il est nécessaire de ne pas lier le déclenchement de l'événement en question à la condition "[monnaie< 0]", ce qui semble difficilement évitable dans le cas d'une transition réflexive de l'état vers lui-même.

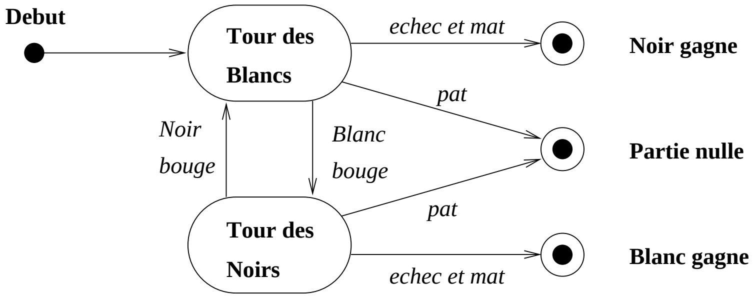
Au passage, on notera aussi que l'action de fermeture de la barrière n'est pas représentée ici car elle dépend du capteur de passage, donc d'un autre objet.

- Au risque de se répéter, on notera qu'un diagramme d'états décrit le comportement d'*une seule* classe d'objets. Le modèle dynamique est constitué d'une collection de diagrammes d'états qui interagissent les uns avec les autres, par l'intermédiaire d'événements partagés. L'envoi d'événements d'une classe d'objets à une autre est notée par une ligne pointillée vers la classe cible. La flèche peut aussi être connectée directement à une transition à l'intérieur du diagramme d'états de l'objet cible lorsqu'on veut indiquer que la transition cible dépend de l'événement.

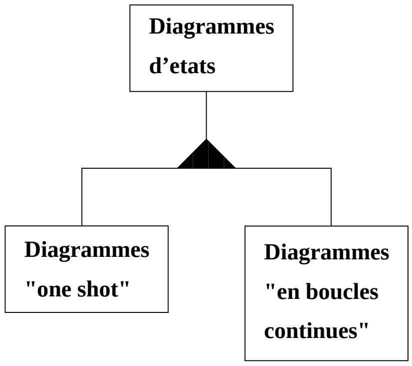
Remarque : Les concepteurs d'OMT distinguent diagrammes d'états pouvant s'exécuter en un seule fois (on parle de diagrammes "one shot"), ou en boucles continues. Le diagramme de la ligne téléphonique est un exemple de diagramme en boucle :



Les diagrammes "one shot" possèdent (au moins) un état initial (i.e. auquel n'arrive aucune transition) et (au moins) un état final (i.e. duquel ne part aucune transition). Ces états sont distingués respectivement par un rond noir et par un cercle avec un rond noir à l'intérieur. L'exemple suivant, qui modélise le cycle de vie d'une partie d'échecs, illustre un tel type de diagramme :



Implicitement, un diagramme "one shot" est censé dénoter un objet ayant une durée de vie finie. Cela semble naturel. Toutefois les choses ne sont pas aussi claires lorsqu'au coeur d'un tel diagramme on trouve la possibilité d'une boucle sur une ou plusieurs transitions, malgré la présence des états terminaux. Ainsi dans la modélisation d'une partie d'échecs évoquée ci-dessus, rien n'empêche *a priori* les noirs et les blancs d'osciller alternativement entre deux positions, bloquant le jeu de telle façon qu'aucun état final soit jamais atteint, rendant la partie virtuellement éternelle... Le partage entre diagrammes à boucles continues explicites et diagrammes "one shot" dénote donc une généralisation avec recouvrement[10] :



12 Concepts objets et modèle dynamique

Rien n'empêche d'appréhender états et événements comme deux classes particulières d'objets. Généralisation et agrégation apparaissent alors comme des notions du modèle objet tout ce qu'il y a de plus fécondes pour le modèle dynamique. De façon synthétique l'apport du modèle objet au modèle dynamique est résumé dans le tableau

[10]mais une généralisation sans recouvrement eut été possible (sinon préférable?) à partir précisément de la distinction purement formelle entre diagrammes possédant des états terminaux et diagrammes n'en possédant pas...

suivant :

	Etats	Evénements
Généralisation	**diagrammes imbriqués**	**généralisation d'événements**
Agrégation	**états concurrents**	**activités concurrentes**

12.1 Diagrammes imbriqués/généralisation d'états

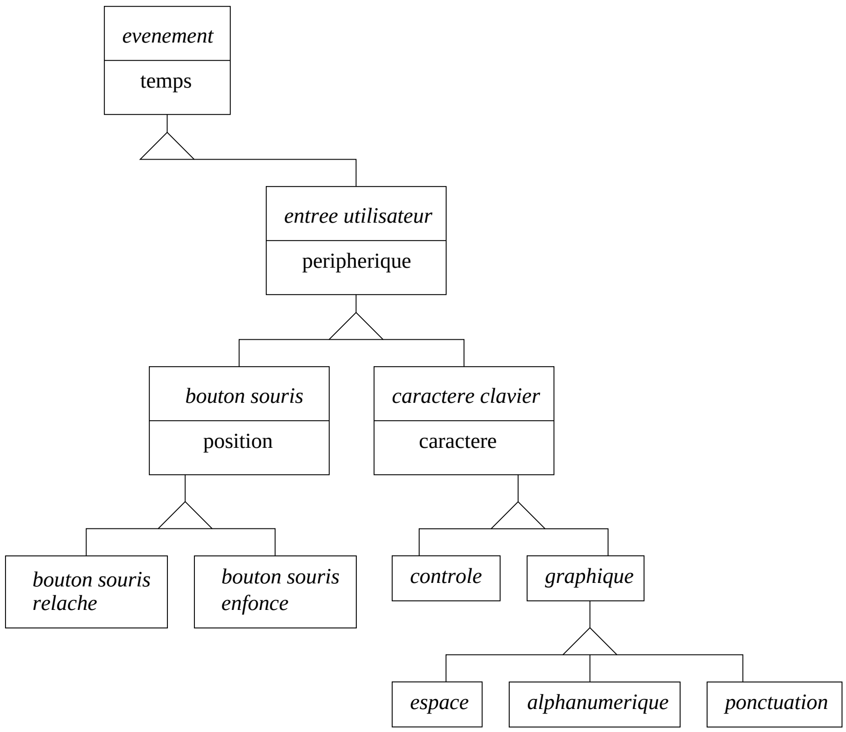
L'idée sur laquelle repose la notion de diagramme imbriqué est que la description d'une activité complexe associée à un état donné au sein d'un diagramme peut être raffinée en un diagramme de plus bas niveau. Un tel "sous"-diagramme, apparenté à un "sous-programme", possède un état d'entrée et un état de sortie (autrement dit, est un diagramme "one shot"). De fait, un diagramme d'états imbriqué correspond bien à une généralisation d'états en ce sens qu'un objet dans un état d'un diagramme de haut niveau doit être precisément dans *un des* état du diagramme de bas niveau : il doit être dans le premier état *ou* dans le second état *ou* dans un des autres états... Par exemple, l'activité "connexion" du diagramme de la ligne téléphonique correspond à une opération complexe dont le détail n'est pas pertinent à ce niveau de la description. Inversement, il eut été possible de ne considérer que deux super-états duaux au plus haut niveau de description de l'activité de la ligne : *En attente* et *En activité*, ce dernier état donnant lieu à un diagramme détats imbriqué : tous les états du diagramme d'origine, sauf *En attente*, devenant alors sous-états de *En activité.* On notera que :

- les sous-états *héritent* des transitions des super-états;

- les états d'un diagramme imbriqué ne sont pas affectés par des transitions dans le diagramme haut niveau, mais dans des situations complexes ces états peuvent interagir directement avec des états du diagramme de haut niveau;

12.2 Généralisation d'événements

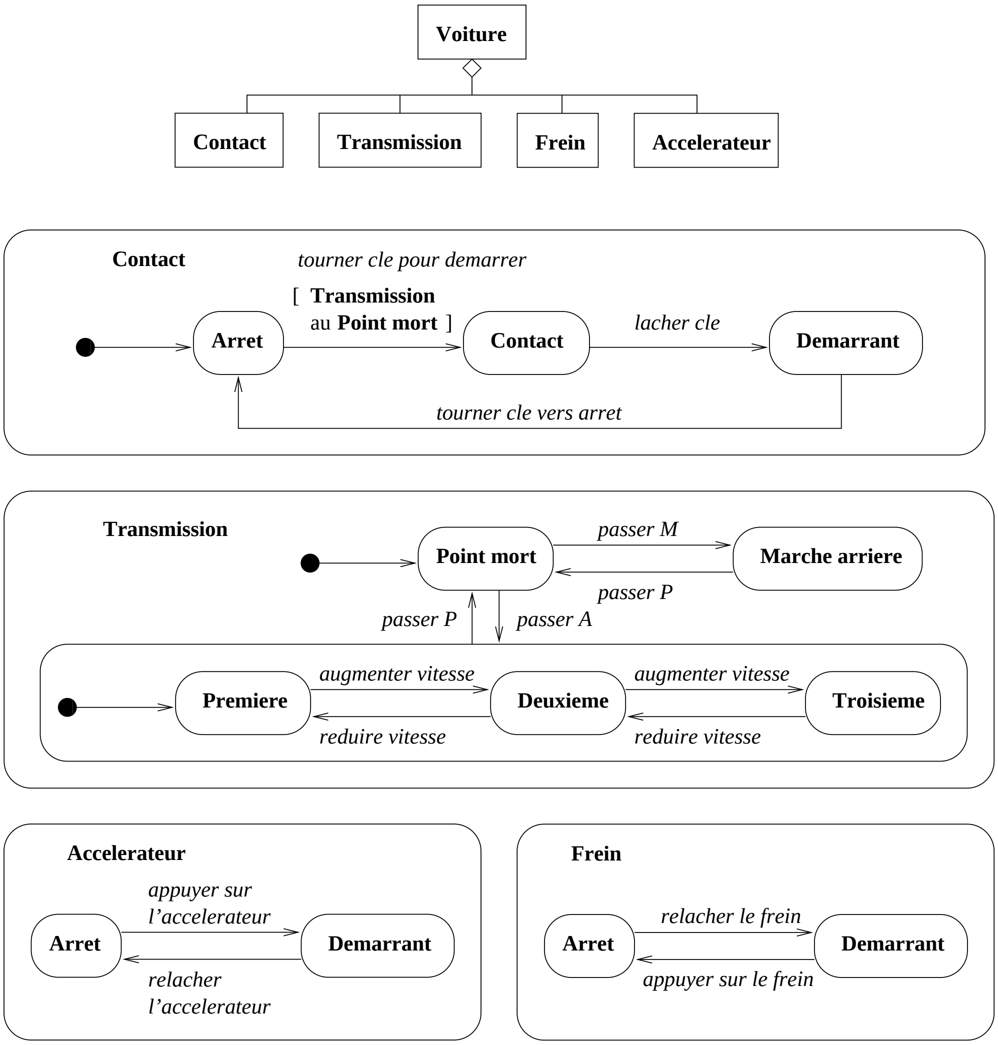
Les événements peuvent être organisés en une hiérarchie de généralisation avec héritage des attributs d'événements. L'intérêt est de pouvoir utiliser des niveaux d'abstraction différents à différents endroits du modèle. Le schéma suivant illustre une hiérarchisation de généralisation d'une partie des événements d'entrée pour une station de travail :



La partie haute de la boîte contient le nom de l'événement et la partie basse les attributs. Les événements de bas niveaux *héritent* des attributs d'événements de haut niveau, par exemple *bouton souris relâché* et *bouton souris enfoncé* de l'attribut *position* de *bouton souris*. Comme cela a été dit plus haut, ce diagramme de généralisation permet d'indiquer que suivant le niveau d'abstraction adopté on peut raffiner ou pas le traitement des attributs, par exemple décider à certains endroits du modèle de traiter indifférement ou pas les caractères de contrôle et les caractères imprimables.

12.3 Etats concurrents

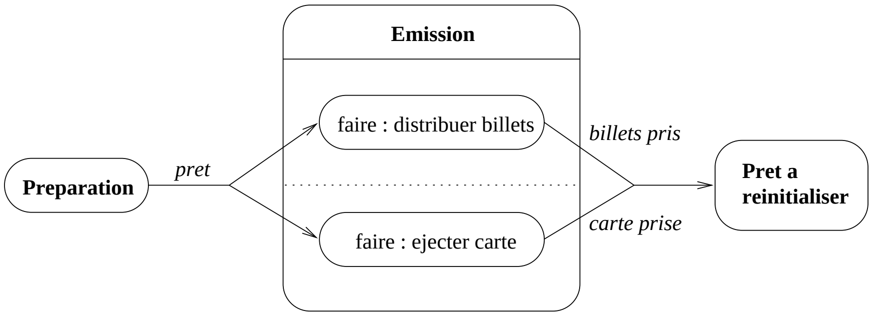
L'agrégation de plusieurs classes d'objets implique la concurrence entre états des diagrammes associés à chaque classe. L'état global de la classe agrégative correspond aux états combinés des diagrammes des composants. L'agrégation exprimant une relation *et*, l'état agrégé est un état du premier diagramme *et* un état du second diagramme *et* un état de chaque autre diagramme. L'exemple suivant montre l'état d'une voiture à boite automatique comme une agrégation d'états composants :



On notera que dans une situation complexe les états composants peuvent interagir. Ainsi la transition gardée entre les états *Arrêt* et *Contact* de l'objet *Contact* dépend de l'état dans lequel se trouve l'objet *Transmission.*

12.4 Activités concurrentes

La concurrence entre les états des diagrammes associés *aux* objets d'une agrégation sous-tend une concurrence implicite entre événements qui arrivent à ou partent de ces états. Inversement, l'idée d'agréger des événements indépendamment des états sous-tend une concurrence implicite entre états auquels arrivent ou desquels partent les événements en question. Un état n'étant que la représentation d'une activité, c'est donc d'*activités concurrentes* réalisées par *un* objet qu'il s'agit ici. L'exemple suivant illustre la réalisation de deux activités concurrentes par un distributeur automatique de billets de banques. Le distributeur ne doit pas se réinitialiser tant que l'utilisateur n'a pas pris sa carte et les billets, mais l'ordre dans lequel ils sont pris n'a aucune d'importance : seul compte le fait que carte *et* billets soient pris.



Les activités concurrentes à l'intérieur d'une activité composée unique associée à un super-état sont représentées en réalisant une division du super-état par des lignes pointillées. Chaque zone est un sous-diagramme qui représente une activité concurrente à l'intérieur d'une activité composée. La division du contrôle est représentée par une transition se séparant en parties concurrentes vers ou à partir des états concurrents. Rien n'est dit sur la synchronisation des activités concurrentes qui peuvent être simultanées ou non.

Troisième partie

Le modèle fonctionnel

13 Introduction

Un modèle *fonctionnel* décrit les calculs à l'intérieur du système, en traitant :

- la dérivation des valeurs sortantes d'un calcul à partir de valeurs entrantes;
- les contraintes entre valeurs à l'intérieur d'un modèle objet.

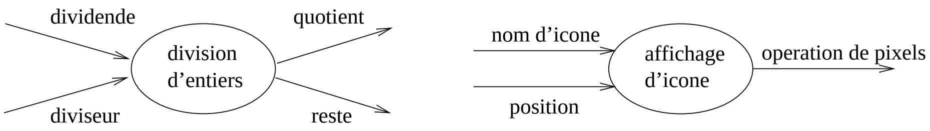
14 Diagrammes à flots de données

Un modèle fonctionnel consiste en un ensemble de **diagrammes à flots de données**[11] qui précisent la signification des opérations et des contraintes. Un diagramme à flots de données contient :

- des **traitements** qui *transforment* la valeur des données;
- des **flots de données** qui *transportent* les données;
- des **objets acteurs** qui *produisent* ou *consomment* les données;
- des **objets réservoirs** qui *stockent* les données.

14.1 Traitements

Un traitement est représenté par une ellipse contenant une description de la transformation. A un traitement est associé un certain nombre de flèches de données entrantes ou sortantes, portant une valeur de donnée. Par exemple :



On notera :

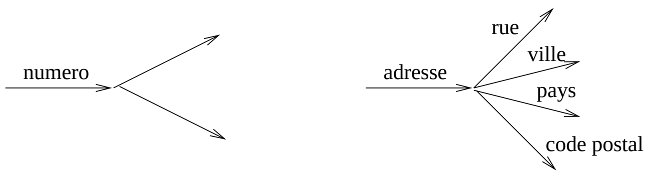
- qu'un traitement de haut niveau peut être étendu en un diagramme à flots de données de plus bas niveau, le niveau élémentaire étant celui des traitements atomiques (équations, opérations simples, etc...);
- que les traitements sont implémentés comme des *méthodes* sur des classes d'objets.

14.2 Flots de données

Un flot de données est représenté par une flèche entre le producteur et le consommateur de la valeur de donnée. La flèche est étiquetée par le type des données. Par

[11] DFD pour les intimes... Pour une présentation théorique, cf. [Yourdon, 89]

exemple :



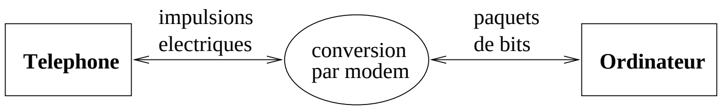
On notera :

- qu'une même valeur peut être envoyée à plusieurs endroits;

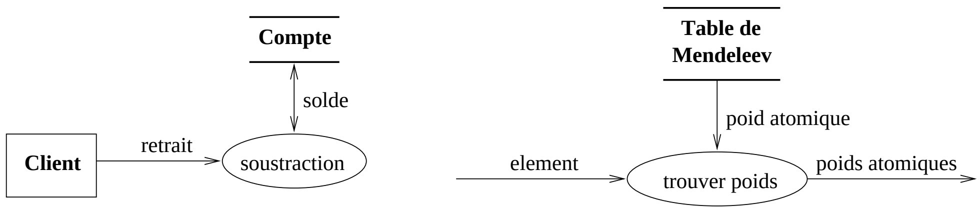
- qu'une valeur peut dénoter un agrégat de composants dirigés vers des traitements différents.

14.3 Acteurs, Réservoirs et objets

Un acteur est un objet qui *produit* ou *consomme activement* des valeurs. Un acteur est représenté par un rectangle pour montrer qu'il s'agit d'un objet.



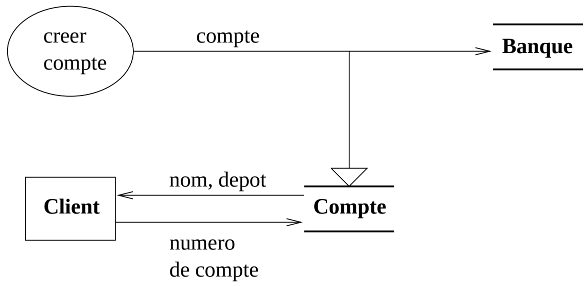
Un réservoir de données est un objet qui *stocke passivement* des données en vue d'un accés futur. A la différence d'un acteur, un réservoir de données n'engendre pas d'opération par lui-même, mais se contente de répondre à des requêtes de stockage ou d'accés aux données qu'il contient. Un réservoir est représenté par une paire de lignes parallèles contenant le nom du réservoir. Par exemple :



15 Concepts évolués autour de la notion de flot

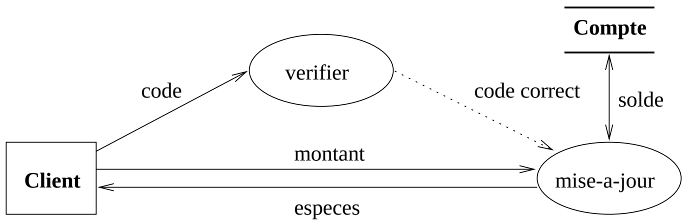
15.1 Création d'un objet

Un flot de données peut provoquer la création d'un objet, acteur ou réservoir, utilisé comme cible d'une opération. Un tel cas de figure est représenté par un triangle vide à l'extrémité du flot arrivant sur l'objet crée. Par exemple :



15.2 Flot de contrôle

Le contrôle des opérations fait normalement l'objet du modèle dynamique. Il peut pourtant être utile de faire relever certaines fonctions de décision directement du modèle dynamique lorsque l'échange des données en dépend. On utilise alors un type particulier de flot, appelé **flot de contrôle**. Un flot de contrôle est une valeur booléenne qui permet de savoir si un traitement est évalué; contrairement au flot de données, il n'est pas une valeur entrante vers le traitement lui-même. Un flot de contrôle est représenté par une ligne pointillée partant d'un traitement produisant une valeur booléenne vers le traitement contrôlé. Voici par exemple un DFD pour un retrait sur un compte en banque :



On notera tout de même que les flots de contrôle, dans la mesure où ils dupliquent l'information du modèle dynamique, doivent être consommés avec modération...

Bibliographie

[Borges, 60] Borges J. L., *El Hacedor.* Emece Editores. Traduction française sous le titre *L'auteur et autre textes.* Gallimard, 1965, 1982. ISBN 2-07-024037-1.

[Chen, 76] Chen P., *The Entity Relationship Mode - Towards a Unified View of Data.* ACM Transactions on Database System 1, 1, mars 1976.

[Elmasri, Navathe, 94] Elmasri R., Navathe S. B., *Fundamentals of Database Systems.* The Benjamin/Cummings publishing Company, Inc. ISBN 0-8053-1748-1.

[Hopcroft, Ullman, 79] Hopcroft J., Ullman J., *Introduction to Automata Theory, Langages, and Computation.* Addison-Wesley. ISBN 0-201-02988-X.

[Rumbaugh et Al., 91] Rumbaugh J., Blaha M., Eddy F., Premerlani W., Lorensen W., *Object Oriented Modeling and Design.* Prentice Hall. Traduction française sous le titre *Modélisation et Conception Orientées Objet.* Masson, 1995. ISBN 2-225-84684-7.

[Salomaa, 85] Salomaa A. *Computation and Automata.* Encyclopedia of Mathematics and its Applications, vol. 25, Cambridge University Press. ISBN 0-521-30245-5.

[Yourdon, 89] Yourdon E. *Modern Structured Analysis.* Englewood Cliffs, New Jersey: Yourdon Press.

Annexe A

Glossaire

acteur : objet *producteur* ou *consommateur actif* de données (voir **réservoir**) (p. 37).

action : opération *instantanée* (p. 24).

activité : opération ayant une *durée* (p. 24).

agrégation : décrit une forme particulière d'association entre un tout et ses parties, dans laquelle le tout *est composé* de ses parties (p. 16).

association : groupe de **liens** ayant une structure sémantique commune, et dénotant une relation entre des **classes** (p. 6).

attribut : valeur de donnée caractéristique détenue par les **instances** d'une **classe** (p. 5).

classe : abstraction qui décrit les propriétés pertinentes pour l'application et ignore les autres; collection **d'instances d'objets** (p. 5).

classification : les objets ayant la même structure de données (**attributs**) et le même comportement (**opérations**) sont regroupés dans une **classe**; une des quatre propriétés caractéristiques de toute approche orientée objet (voir **identité**, **polymorphysme**, et **héritage**) (p. 1).

concurrence : plusieurs **actions** ou **activités** dont l'exécution peut se chevaucher (p. 32-33).

condition : fonction booléenne de valeurs d'objets valide sur un intervalle de temps (p. 27).

contrainte : relation entre objets, classes, attributs, liens ou associations, dont la valeur doit être maintenue à *vrai* (p. 14).

état : valeurs des attributs et des liens d'un objet à un *moment* donné; représente la validation d'un ensemble de **conditions**, et éventuellement associé à une **activité** (voir **événement**) (p. 24).

événement: stimulus reccu ou envoyé par un objet à un *instant* donné; éventuellement associé à une **action** (voir **état**) (p. 24).

flot de contrôle : valeur booléenne qui décide de l'exécution d'un **traitement** (p. 38).

flot de données : connexion entre les sorties d'un objet ou d'un **traitement** et les entrées dans un autre (p. 36).

généralisation : décrit une forme particulière de relation entre une **classe** et une ou plusieurs versions *raffinées* de celle-ci (p. 18).

héritage : partage des **attributs** et des **opérations** entre **classes** s'appuyant sur une relation hiérarchique; une des quatre propriétés caractéristiques de toute approche orientée objet (voir **identité**, **classification**, et **polymorphysme**) (p. 1).

identité : les données sont quantifiées en entités discrètes que l'on peut distinguer et nommer objets; une des quatre propriétés caractéristiques de toute approche orientée objet (voir **classification**, **polymorphysme**, et **héritage**) (p. 1).

instance d'objet : désigne une occurence particulière d'une entité conceptualisée comme "objet" (p. 5).

lien : connection physique ou conceptuelle entre des instances d'objets (p. 6).

méthode : implémentation spécifique d'une **opération** (p. 2).

multiplicité : nombre d'**instances** d'une **classe** qui peuvent se relier à une **instance** unique d'une **classe** associée (p. 9).

nom de rôle : nom précisant la fonction assurée par une **classe** à une extrémité d'une **association** (voir **qualifieur**) (p. 11).

opération : action ou une transformation qu'un objet effectue ou subit (p. 5).

polymorphysme : la même **opération** peut se comporter différemment sur différentes **classes**; une des quatre propriétés caractéristiques de toute approche orientée objet (voir **identité**, **classification**, et **héritage**) (p. 1).

qualifieur : attribut d'une classe permettant de discriminer les instances à l'extrémité "plusieurs" de l'association (p. 11).

réservoir : objet qui *stocke passivement* des données en vue d'un accés futur (voir **acteur**) (p. 37).

scénario : une séquence particulière d'**événements** survenant du point de vue d'*un* objet (p. 25).

suivi d'événements : diagramme permettant d'identifier émetteur, récepteur, et séquence d'**événements** (p. 25).

traitement : opération de transformation des *valeurs de données* (p. 36).

transition : changement d'**état** provoqué par un **événement** (p. 26-27).

Annexe B

Notation