Pascal Roques

7e édition

UML 2 par la pratique

Études de cas et exercices corrigés

EYROLLES

Pascal Roques

Consultant senior et formateur chez A2 – Artal Innovation, Pascal Roques a plus de vingt ans d'expérience dans la modélisation de systèmes complexes (SADT, OMT, UML, SysML…). Il a ainsi été responsable des formations Valtech Training sur le thème « Analyse, conception et modélisation avec UML » pendant de nombreuses années. Auteur de plusieurs livres consacrés à UML et SysML aux éditions Eyrolles, il a obtenu la certification OMG-Certified UML Advanced Professional proposée par l'OMG.

Septième édition augmentée : un cours pratique magistral sur UML 2

Cette septième édition mise à jour et augmentée de l'ouvrage *UML 2 par la pratique* constitue un support de cours exemplaire sur UML 2. Il traite les axes fonctionnel, statique et dynamique de la modélisation UML par des études de cas et des exercices corrigés donnant les bases d'une démarche méthodique. Chaque choix de modélisation est minutieusement commenté ; des conseils issus de l'expérience de l'auteur ainsi que de nombreux avis d'experts sont donnés. En fin d'ouvrage, un glossaire reprend les définitions des principaux concepts étudiés. Les nouveaux concepts et diagrammes UML 2 sont traités en détail : diagramme de structure composite, nouveautés du diagramme d'activité et du diagramme de séquence, etc., en tenant compte des méthodes de développement agiles. Enfin, une étude de cas complète illustre le processus de développement itératif depuis la modélisation métier jusqu'à la conception détaillée en Java et C#.

À qui s'adresse ce livre ?

- Aux étudiants en informatique (cursus génie logiciel ou modélisation UML) et à leurs professeurs, qui y trouveront un matériel précieux pour illustrer par des cas réels les concepts étudiés en cours.
- À toutes les personnes impliquées dans des projets de développement logiciel : maîtres d'ouvrage, chefs de projet, analystes et concepteurs, architectes logiciel, développeurs, etc.

Au sommaire

I. POINT DE VUE FONCTIONNEL – **Cours et définitions – Étude d'un guichet automatique de banque – Exercices corrigés et conseils méthodologiques.** Mots-clés : acteur, cas d'utilisation, *user story*, diagramme de cas d'utilisation, description textuelle, scénario, diagramme de séquence, diagramme d'activité, inclusion, extension et généralisation, *interaction overview diagram*. II. POINT DE VUE STATIQUE – **Cours et définitions – Étude d'un système de réservation de vols – Exercices corrigés et conseils méthodologiques.** Mots-clés : classe, objet, opération, attribut, association, multiplicité, agrégation, composition, généralisation, classe abstraite, classe d'association, qualificatif, contrainte, package, pattern, classe structurée, diagramme de structure composite, SysML. III. POINT DE VUE DYNAMIQUE – **Cours et définitions – Étude d'un Publiphone – Exercices corrigés et conseils méthodologiques.** Mots-clés : message, diagramme d'états, état, transition, événement, condition, effet, action, activité, diagramme d'activité, flot de contrôle, flot d'objet, décision, région d'expansion, région interruptible. IV. ÉTUDE DE CAS – **Une étude de cas complète : de la modélisation métier à la conception détaillée en Java et C# – Études de cas complémentaires.** Mots-clés : modélisation métier, processus métier, itération, architecture en couches, package, opération système, contrat d'opération, diagramme de communication, diagramme de classes de conception, collection, méthode, polymorphisme, navigabilité, dépendance, code Java, code C#, composant, interface, artéfact, diagramme de déploiement, design pattern. **Conclusion : UML, 12 ans après.** V. ANNEXES – **Correspondances UML/Java/C# – Glossaire – Bibliographie – Carte de référence UML 2.**

www.editions-eyrolles.com
Groupe Eyrolles | Diffusion Geodif | Distribution Sodis

UML 2

par la pratique

Études de cas et exercices corrigés

7e édition

CHEZ LE MÊME ÉDITEUR

Du même auteur

P. Roques. – **UML 2.** *Modéliser une application web.*
N°12389, 4e édition, 2008, 246 pages (collection Cahiers du programmeur).

P. Roques, F. Vallée. – **UML 2 en action.** *De l'analyse des besoins à la conception*.
N°12104, 4e édition, 2007, 382 pages (collection Architecte logiciel).

P. Roques. – **Mémento UML.**
N°11725, 2006, 14 pages.

UML, modélisation objet, processus de développement

C. Soutou. – **UML 2 pour les bases de données.** *Avec 20 exercices corrigés*.
N°12091, 2007, 314 pages (collection Noire).

X Blanc, I. Mounier. – **UML 2 pour les développeurs.** *Cours avec exercices corrigés.*
N°12029, 2006, 202 pages (collection Noire).

F. Vallée. – **UML pour les décideurs**.
N°11621, 2005, 282 pages (collection Architecte logiciel).

H. Balzert. – **UML 2 Compact**.
N°11753, 2006, 88 pages.

H. Bersini, I. Wellesz. – **Programmation orientée objet**. *Cours et exercices en UML 2 avec Java, C# 2, C++, Python, PHP 5 et LINQ.*
N° 12441, 4e édition, 2008, 616 pages (collection Noire).

B. Meyer. – **Conception et programmation orientées objet.**
N°12270, 2008, 1222 pages (collection Blanche).

V. Messager Rota. – **Gestion de projet**. *Vers les méthodes agiles*.
N°12518, 2e édition, 2009, 272 pages (collection Architecte logiciel).

A. Lonjon, J.-J. Thomasson. – **Modélisation XML**.
N°11521, 2006, 498 pages (collection Architecte logiciel).

X. Blanc. – **MDA en action.** *Ingénierie logicielle guidée par les modèles*.
N°11539, 2005, 270 pages avec CD-Rom (collection Architecte logiciel).

J.-L. Bénard, L. Bossavit , R. Médina , D. Williams. – **Gestion de projet eXtreme Programming.**
N°11561, 2002, 300 pages (collection Architecte logiciel).

Autres ouvrages

A. Tasso. – **Apprendre à programmer en Actionscript 3.**
N°12550, 2009, 542 pages (collection Noire).

C. Delannoy. – **Programmer en langage C**.
N°12546, 5e édition, 2009, 276 pages (collection Noire).

J. Engels. – **PHP 5**.
N°12486, 2e édition, 2009, 638 pages (collection Noire).

G. Swinnen. – **Apprendre à programmer avec Python**.
N°12474, 2009, 342 pages (collection Noire).

G. Pujolle. – **Cours réseaux et télécoms.**
N°12415, 3e édition, 2008, 544 pages (collection Noire).

I Canivet. – **Bien rédiger pour le Web.**
N°12433, 2009, 412 pages (collection Accès libre).

F. Potencier, H. Hamon. **– Symfony.**
N°12494, 2009, 496 (collection Cahiers du programmeur).

Pascal Roques

UML 2 par la pratique

Études de cas et exercices corrigés

7e édition

EYROLLES

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

A est un bon modèle de B si A permet de répondre de façon satisfaisante à des questions prédéfinies sur B.

Douglas. T. Ross

La différence entre la théorie et la pratique, c'est qu'en théorie, il n'y a pas de différence entre la théorie et la pratique, mais qu'en pratique, il y en a une.

Jan van de Sneptscheut

Depuis des temps très anciens, les hommes ont cherché un langage à la fois universel et synthétique, et leurs recherches les ont amenés à découvrir des images, des symboles qui expriment, en les réduisant à l'essentiel, les réalités les plus riches et les plus complexes.

Les images, les symboles parlent, ils ont un langage.

O.M. Aïvanhov

Préface

Adopté et standardisé par l'Object Management Group depuis 1997, UML est aujourd'hui un outil de communication incontournable, utilisé sur des centaines de projets de par le monde ; en conséquence, la connaissance d'UML est désormais une des compétences qui sont exigées quasi systématiquement lors d'un recrutement. Pourtant, trop nombreux sont encore les concepteurs qui s'imaginent, à tort, posséder cette compétence parce qu'ils connaissent la représentation graphique d'une classe d'association, d'une activité ou d'un état, tandis qu'ils ne savent pas expliquer ni défendre leur emploi dans un modèle, même simple. C'est que, malgré la rigueur apportée à la spécification d'UML, il y a parfois différentes façons de représenter une même idée, ou, à l'inverse, une idée donnée pourra être représentée avec plus ou moins de précision selon que l'on aura utilisé telle ou telle particularité syntaxique d'UML.

Si l'on trouve aujourd'hui pléthore de formations à UML, il faut bien reconnaître que la plupart d'entre elles se trompent d'objectif. L'enseignement d'UML en tant que tel ne présente pas plus de difficulté ni d'intérêt que l'enseignement de n'importe quel autre langage de modélisation. Toutefois, ce qui importe en la matière, c'est d'enseigner efficacement les principes qui sous-tendent une démarche objet dédiée à l'analyse et à la conception de systèmes complexes, mais aussi qu'un emploi judicieux des éléments syntaxiques d'UML permette une représentation pertinente des concepts métier et des choix de conception.

Consultant expérimenté et pionnier de l'utilisation d'UML en France, Pascal est aussi un excellent concepteur de cours et un formateur à la pédagogie sans faille. Il n'en est pas non plus à son premier coup d'essai en tant qu'auteur. Après *UML 2 en action*, il nous livre maintenant *UML 2 par la pratique*, un excellent recueil de modèles UML issus de projets réels, ou construits pour des études de cas, et minutieusement commentés. Le caractère unique de ce livre réside dans la mise par

écrit de commentaires essentiels, qui visent à expliquer un choix de modélisation et qui, généralement, ne sont donnés que par oral sur le « terrain ». Le lecteur pourra ainsi revenir à loisir sur ce qui lui fait souvent défaut, par exemple la justification de l'emploi de tel ou tel trait syntaxique plutôt qu'un autre pour représenter une idée particulière. La mise en parallèle de solutions « alternatives » permet également de comprendre les avantages d'une modélisation sur une autre et la précision sémantique apportée par la solution choisie.

C'est toute la puissance d'expression d'UML 2 et la complétude de sa syntaxe qui sont démontrées par l'exemple dans cet ouvrage pragmatique et très accessible, que le lecteur, j'en suis certain, prendra plaisir comme moi à lire d'un bout à l'autre. Je me permets cependant de recommander de bien réaliser les exercices complémentaires, excellents travaux pratiques, mais aussi remarquables tests d'auto-évaluation. En couvrant tous les aspects de l'analyse et de la conception orientée objet, Pascal Roques illustre et explique les variations possibles dans l'usage d'UML pour chacune de ces étapes.

En tant que responsable de la définition de l'offre formation chez Valtech Training, j'ai longtemps eu beaucoup de difficulté à recommander un livre qui serait un bon complément à nos formations à la modélisation objet avec UML. Merci Pascal, grâce à la qualité pédagogique de ton livre, *UML 2 par la pratique*, cette tâche est aujourd'hui considérablement plus simple…

Gaël Renault

Directeur pédagogique

Valtech Training

Sommaire

Introduction **9**

Objectifs du livre 9
Structure de l'ouvrage 10
Conventions typographiques 12
Remerciements 13

PARTIE I – POINT DE VUE FONCTIONNEL

Chapitre 1 • Modélisation fonctionnelle : étude de cas **17**

Principes et définitions de base 18
Étude d'un guichet automatique de banque 21
Étape 1 – Identification des acteurs du GAB 22
Étape 2 – Identification des cas d'utilisation 25
Étape 3 – Réalisation de diagrammes de cas d'utilisation 26
Étape 4 – Description textuelle des cas d'utilisation 30
Étape 5 – Description graphique des cas d'utilisation 36
Étape 6 – Organisation des cas d'utilisation 40
Étape 7 – Dynamique globale : Interaction Overview Diagram 49

Chapitre 2 • Modélisation fonctionnelle : exercices corrigés et conseils méthodologiques **53**

Étude d'un terminal point de vente (TPV) 54
Étape 1 – Réalisation du diagramme de cas d'utilisation 55
Étape 2 – Descriptions essentielle et réelle d'un cas d'utilisation 59
Étape 3 – Description graphique des cas d'utilisation 66
Étape 4 – Réalisation d'un diagramme d'états au niveau système 71
Conseils méthodologiques 74

PARTIE II – POINT DE VUE STATIQUE

Chapitre 3 • Modélisation statique : étude de cas **83**

Principes et définitions de base 84
Étude d'un système de réservation de vol 88
Étape 1 – Modélisation des phrases 1 et 2 89
Étape 2 – Modélisation des phrases 6, 7 et 10 91
Étape 3 – Modélisation des phrases 8 et 9 95
Étape 4 – Modélisation des phrases 3, 4 et 5 99
Étape 5 – Ajout d'attributs, de contraintes et de qualificatifs 102
Étape 6 – Utilisation de patterns d'analyse 107
Étape 7 – Structuration en packages 110
Étape 8 – Inversion des dépendances 116
Étape 9 – Généralisation et réutilisation 118

Chapitre 4 • Modélisation statique : exercices corrigés et conseils méthodologiques **125**

Compléments sur les relations entre classes 126
Modélisation du domaine en pratique 136
Les classes structurées UML 2 143
Découverte d'un « pattern » 152
Conseils méthodologiques 156

PARTIE III – POINT DE VUE DYNAMIQUE

Chapitre 5 • Modélisation dynamique : étude de cas 163

Principes et définitions de base 164
Étude d'un Publiphone à pièces 169
Étape 1 – Identification des acteurs et des cas d'utilisation 170
Étape 2 – Réalisation du diagramme de séquence système 171
Étape 3 – Représentation du contexte dynamique 174
Étape 4 – Description exhaustive par un diagramme d'états 177

Chapitre 6 • Modélisation dynamique : exercices corrigés et conseils méthodologiques 191

Concepts de base du diagramme d'états 192
Concepts avancés du diagramme d'états 197
Concepts de base du diagramme d'activité 213
Concepts avancés du diagramme d'activité 219
Conseils méthodologiques 222

PARTIE IV – CONCEPTION

Chapitre 7 • Étude de cas complète : de la modélisation métier à la conception détaillée en Java ou C# 229

Étape 1 – Modélisation métier (business modeling) 231
Étape 2 – Définition des besoins du système informatique 235
Étape 3 – Analyse du domaine (partie statique) 244
Étape 4 – Analyse du domaine (partie dynamique) 258
Étape 5 – Définition des itérations 262
Étape 6 – Définition de l'architecture système 263
Étape 7 – Définition des opérations système (itération #1) 268
Étape 8 – Diagrammes d'interaction (itération #1) 271
Étape 9 – Diagrammes de classes de conception (itération #1) 280

Étape 10 – Définition des opérations système (itérations #2 et #3) 287
Étape 11 – Contrats d'opérations (itérations #2 et #3) 291
Étape 12 – Diagrammes d'interaction (itérations #2 et #3) 293
Étape 13 – Diagrammes de classes de conception (itérations #2 et #3) 294
Étape 14 – Retour sur l'architecture 296
Étape 15 – Passage au code objet 297
Étape 16 – Déploiement de l'application 307

Chapitre 8 • **Études de cas complémentaires** **309**

Étude du système d'information d'une bibliothèque 310
Analyse et conception du jeu de démineur 325
Conseils méthodologiques 340
Conclusion 345

Conclusion **347**

UML, 12 ans après... 347

ANNEXES

Annexe 1 • **Correspondances UML – Java – C#** **363**

La structure statique 364
Les relations 368

Annexe 2 • **Glossaire** **377**

Annexe 3 • **Bibliographie** **387**

Bibliographie ch1-2 387
Bibliographie ch3-4 388
Bibliographie ch5-6 388
Bibliographie ch7-8 399

Index **391**

Introduction

Objectifs du livre

Depuis quelques années, les ouvrages sur le langage UML et la modélisation objet se sont multipliés. *De Merise à UML*[1], *UML et SQL*[2], *Bases de données avec UML*[3], etc., représentent autant de sujets intéressants pour les professionnels de l'informatique.

Cependant, ma pratique de la formation (plus d'un millier de personnes formées à OMT, puis UML, depuis 1993…) m'a convaincu qu'il existait encore un besoin qui n'est pas couvert par la multitude d'ouvrages disponibles actuellement : un livre d'exercices corrigés[4]. Je consacre en effet de plus en plus de temps pendant les sessions que j'anime à des séances de discussion avec les stagiaires sur les mérites comparés de telle ou telle solution de modélisation. Et je suis intimement persuadé que ces périodes d'interactivité sur des sujets concrets ont un impact bien plus durable pour eux que la présentation théorique des subtilités du formalisme UML !

Cela m'a amené à constituer une importante base de données d'exercices, rassemblés au fil des années à partir des nombreuses formations que j'ai animées. Je me suis également inspiré des livres fondamentaux qui m'ont aidé personnellement dans mon approfondissement de ce sujet, en particulier celui de J. Rumbaugh sur OMT[5] (l'un des premiers à proposer des exercices après chaque chapitre de présentation) et le best-seller de C. Larman[6] sur l'analyse et la conception objet.

C'est ce matériel pédagogique, fondé sur des heures de discussions enrichissantes avec des stagiaires de tous horizons, que je voudrais partager aujourd'hui avec vous. Par leurs questions et leurs propositions, ils m'ont forcé à prendre en compte les points de vue les

1. *De Merise à UML*, N. Kettani *et al.*, Eyrolles, 1998.
2. *UML et SQL* : *Conception de bases de données,* C. Soutou, Eyrolles, 2002.
3. *Bases de données avec UML*, E.Naiburg R.Maksimchuk, Campus Press, 2002.
4. Mon idée semble avoir fait des émules depuis la parution de la première édition en 2001, puisque d'autres livres du même type sont parus au fil des années : *Exercices corrigés d'UML*, Ellipses 2005, *UML 2 – Initiation, exemples et exercices corrigés*, Eni 2008, etc.
5. *Object-Oriented Modeling and Design*, J. Rumbaugh *et al.*, Prentice Hall, 1991. Une version mise à jour est parue dernièrement en français, sous le titre : *Modélisation et conception orientées objet avec UML2*, Pearson Education, 2005 (mais la magie ne semble plus opérer comme la première fois…).
6. *Applying UML and Patterns*, C. Larman, Prentice Hall, 1997. La troisième édition de cet ouvrage a été traduite en français : *UML 2 et les Design Patterns*, Campus Press, 2005.

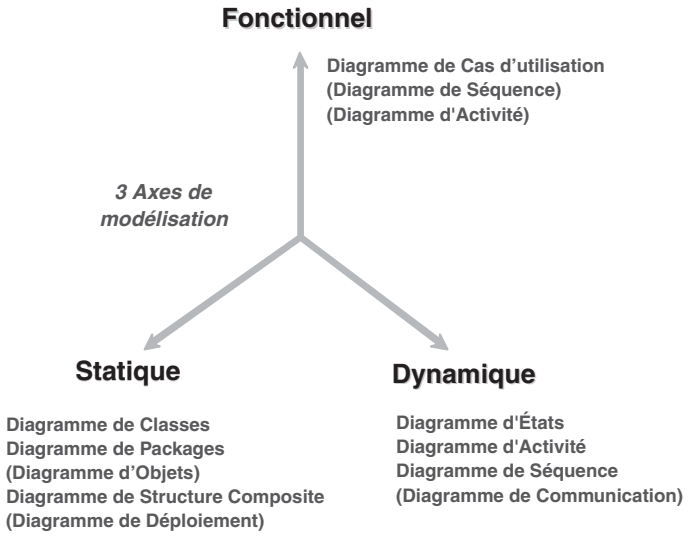
plus divers sur un même problème de modélisation, à améliorer mon argumentation et parfois à envisager de nouvelles solutions auxquelles je n'avais absolument pas pensé !

Il est à noter que cette septième édition incorpore de nombreux avis d'experts auxquels j'ai demandé leur sentiment sur l'actualité et la pertinence d'UML en 2009, près de quinze ans après sa création. Vous retrouverez ces avis, parfois différents, toujours instructifs, dans des encadrés au fil des chapitres ainsi qu'à la fin de l'ouvrage.

J'ai également voulu évoquer le nouveau langage SysML de modélisation de systèmes, dérivé d'UML 2, qui apporte quelques concepts supplémentaires au diagramme de structure composite[7] et au diagramme d'activité.

Structure de l'ouvrage

Pour ne pas mélanger les problèmes, le livre est découpé suivant les trois points de vue classiques de modélisation : fonctionnel, statique et dynamique, en insistant pour chacun sur le ou les diagrammes UML prépondérants (les diagrammes entre parenthèses sont moins détaillés que les autres).



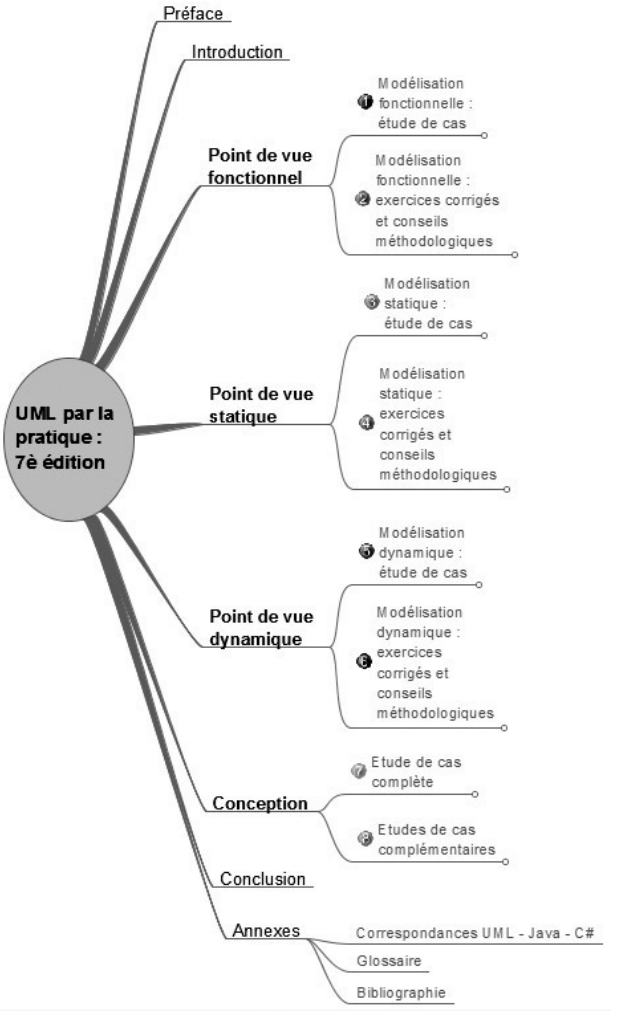
Les trois premières parties du livre correspondent donc chacune à un point de vue de modélisation. Pour chaque partie, une étude de cas principale, spécifique, sert de premier chapitre. Avant l'étude de cas, on trouve un rappel des principales définitions utilisées dans la partie concernée. Des exercices complémentaires et un récapitulatif des conseils méthodologiques se trouvent dans un deuxième chapitre.

7. Voir en particulier le chapitre 4. Pour les lecteurs désireux d'en savoir plus sur le langage SysML, je viens de faire paraître un ouvrage sur le sujet : *SysML par l'exemple*, P. Roques, éditions Eyrolles 2009.

La quatrième partie contient plusieurs études de cas. La première, très détaillée, prend bien sûr en compte les trois points de vue précités, mais couvre également en amont la modélisation métier, et en aval la programmation en Java et C# ! La dernière étude de cas aborde le sujet passionnant des *design patterns*.

Retrouvez sur le rabat de la couverture une carte de référence regroupant les principaux schémas UML 2[8] utilisés dans ce livre.

Une table des matières condensée est donnée par la carte ci-après.



8. J'ai utilisé comme référence pour ce livre la toute dernière spécification de l'OMG, à savoir le document 09-02-02 : UML 2.2 Superstructure.

Conventions typographiques

Pour apporter plus de clarté dans la lecture de ce livre, les exercices et les solutions sont mis en évidence par des polices de caractères et des symboles graphiques différents de la façon suivante :

EXERCICE 1–1. Les acteurs

Identifiez les acteurs du GAB.

Quelles sont les entités externes qui interagissent directement avec le GAB ?

Considérons linéairement les phrases de l'énoncé.

Pour guider un peu plus le lecteur, les questions ont un niveau de difficulté évalué de un à quatre, et représenté graphiquement :

question facile,

question moyenne,

question assez difficile mettant en jeu des concepts UML avancés,

question difficile nécessitant une argumentation complexe.

De façon ponctuelle, pour rompre la monotonie du texte, j'ai également utilisé des encadrés « À retenir » pour isoler une note sur une question d'un niveau avancé :

À RETENIR

Représentations graphiques d'un acteur

La représentation graphique standard de l'acteur en UML est l'icône appelée « stick man », avec le nom de l'acteur sous le dessin. On peut également montrer un acteur sous la forme rectangulaire d'une classe, avec le mot-clé <<`actor`>>. Une troisième représentation (intermédiaire entre les deux premières) est également possible, comme indiqué ci-après.

Remerciements

Je tiens en premier lieu à remercier tous ceux qui ont participé au fil des années à l'élaboration des supports de cours UML que j'ai eu l'occasion d'animer, comme Pierre Chouvalidzé, Thibault Cuvillier, Michel Ezran, Patrick Le Go, Franck Vallée, Philippe Riaux, Philippe Dubosq, Yann Le Tanou, Françoise Caron, Christophe Addinquy, Saïd Lahlouh, Birol Berkem, Fabien Brissoneau, Daniel Rosenblatt, Michel André, etc., ainsi que les nombreux autres experts qui ont répondu à mon appel de retour d'expérience : Claude Aubry, Nicolas Belloir, Jean-Michel Bruel (un merci supplémentaire pour l'idée de la carte de référence !), Agusti Canals, Olivier Casse, Thierry Cros, Philippe Desfray, Jean-Claude Grosjean et Marc Tizzano.

Je voudrais également remercier tous ceux dont les discussions, les remarques et les suggestions m'ont conduit à améliorer mon argumentation. Je pense tout d'abord à mes nombreux stagiaires (et la liste ne cesse de grossir !), ainsi qu'à mes correspondants lors de prestations de conseil sur l'introduction d'UML dans des projets de tous types.

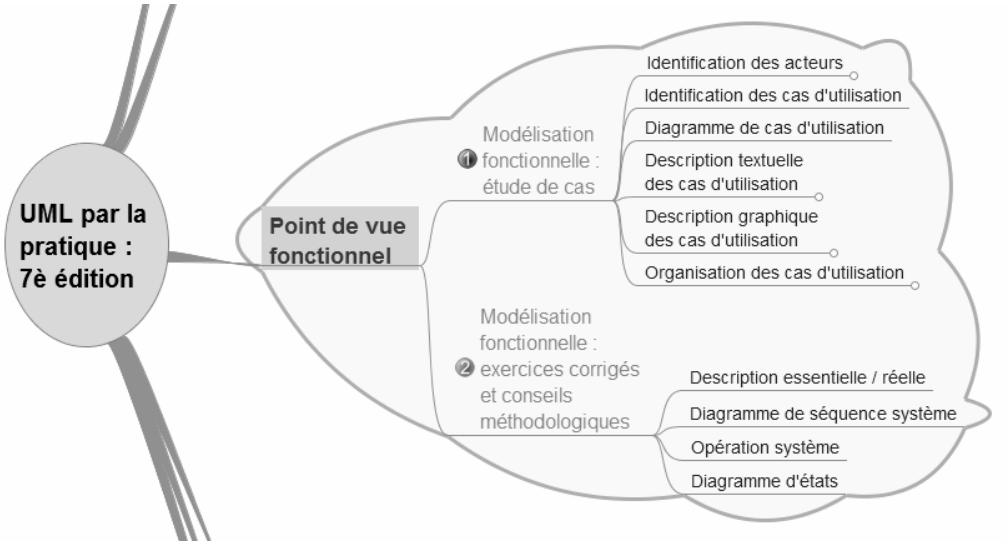
Merci aussi à Éric Sulpice des Éditions Eyrolles pour son témoignage renouvelé de confiance et, surtout, qui a su me motiver en me proposant de réaliser ce livre d'exercices corrigés. Un grand bravo également à l'équipe des éditrices : Muriel et Sandrine qui ont contribué notablement à la réussite de ce projet.

Enfin, un grand merci à Sylvie qui m'apporte tous les jours l'équilibre sans lequel je ne pourrais pas continuer à avancer…

Pascal Roques, mai 2009
Consultant Senior – A2 (Artal Innovation)
pascal.roques@a2-artal.fr
http://consultants.a2-artal.fr/proques

PARTIE I

Point de vue fonctionnel



1

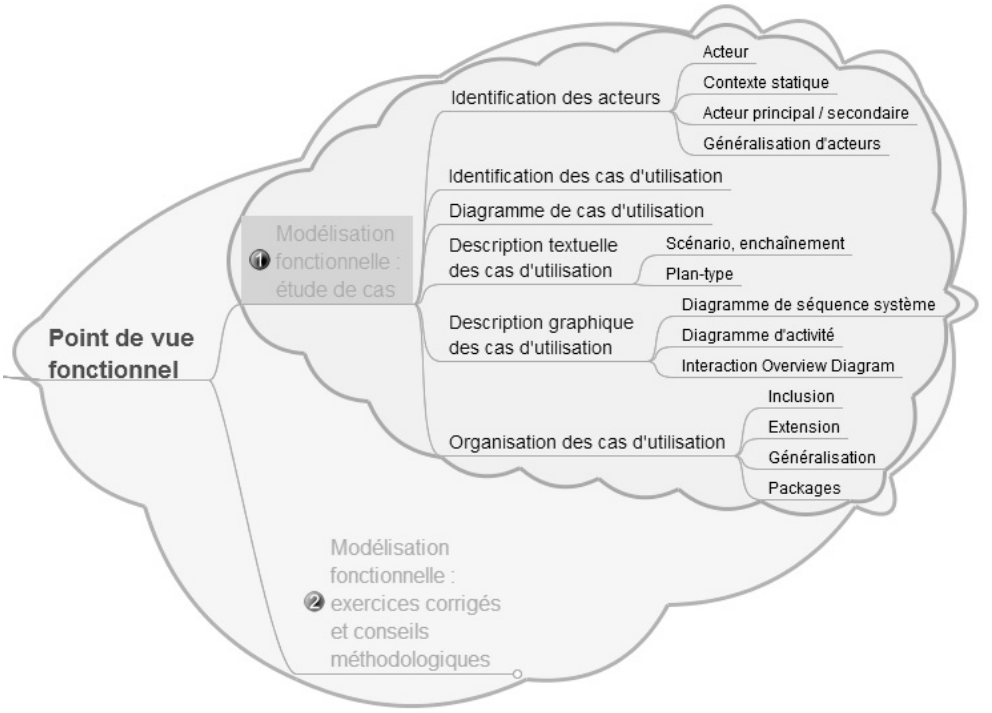
Modélisation fonctionnelle : étude de cas

Mots-clés

■ Acteur ■ Contexte statique ■ Cas d'utilisation ■ Acteur principal, acteur secondaire ■ Diagramme de cas d'utilisation ■ Scénario, enchaînement ■ Fiche de description d'un cas d'utilisation ■ Diagramme de séquence système ■ Diagramme d'activité ■ Inclusion, extension et généralisation de cas d'utilisation ■ Généralisation/spécialisation d'acteurs ■ Package de cas d'utilisation ■ *Interaction Overview Diagram*.

Ce chapitre va nous permettre d'illustrer pas à pas, sur une première étude de cas, les principales difficultés liées à la mise en œuvre de la technique des cas d'utilisation.

Après avoir identifié les acteurs qui interagissent avec le système, nous y développons un premier modèle UML de haut niveau, pour pouvoir établir précisément les frontières du système. Dans cette optique, nous apprenons à identifier les cas d'utilisation et à construire un diagramme reliant les acteurs et les cas d'utilisation. Ensuite, nous précisons le point de vue fonctionnel en détaillant les différentes façons dont les acteurs peuvent utiliser le système. À cet effet, nous apprenons à rédiger des descriptions textuelles de cas d'utilisation, ainsi qu'à dessiner des diagrammes UML complémentaires (comme les diagrammes de séquence ou d'activité).



Principes et définitions de base

Acteur

Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié.

Un acteur peut consulter et/ou modifier directement l'état du système, en émettant et/ou en recevant des messages susceptibles d'être porteurs de données.

Comment les identifier ?

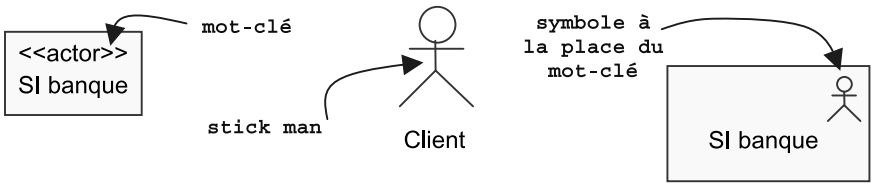
Les acteurs candidats sont systématiquement :

- les utilisateurs humains directs : faites donc en sorte d'identifier tous les profils possibles, sans oublier l'administrateur, l'opérateur de maintenance, etc. ;
- les autres systèmes connexes qui interagissent aussi directement avec le système étudié, souvent par le biais de protocoles bidirectionnels.

Comment les représenter ?

La représentation graphique standard de l'acteur en UML est l'icône appelée *stick man*, avec le nom de l'acteur sous le dessin. On peut également figurer un acteur sous la forme rectangulaire d'une classe, avec le mot-clé <<actor>>. Une troisième représentation (intermédiaire entre les deux premières) est également possible avec certains outils, comme cela est indiqué ci-après.

Figure 1-1.
Représentations graphiques possibles d'un acteur



Une bonne recommandation consiste à faire prévaloir l'utilisation de la forme graphique du *stick man* pour les acteurs humains et une représentation rectangulaire pour les systèmes connectés.

Cas d'utilisation

Un *cas d'utilisation* (« use case ») représente un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur particulier.

Chaque cas d'utilisation spécifie un comportement attendu du système considéré comme un tout, sans imposer le mode de réalisation de ce comportement. Il permet de décrire *ce que* le futur système devra faire, sans spécifier *comment* il le fera.

Comment les identifier ?

L'objectif est le suivant : l'ensemble des cas d'utilisation doit décrire exhaustivement les exigences fonctionnelles du système. Chaque cas d'utilisation correspond donc à une fonction métier du système, selon le point de vue d'un de ses acteurs.

Pour chaque acteur, il convient de :

- rechercher les différentes intentions métier avec lesquelles il utilise le système,
- déterminer dans le cahier des charges les services fonctionnels attendus du système.

Nommez les cas d'utilisation par un verbe à l'infinitif suivi d'un complément, du point de vue de l'acteur (et non pas du point de vue du système).

Comment les analyser ?

Pour détailler la dynamique du cas d'utilisation, la procédure la plus évidente consiste à recenser de façon textuelle toutes les interactions entre les acteurs et le système. Le cas d'utilisation doit avoir un début et une fin clairement identifiés. Il faut également préciser les variantes possibles, telles que le cas nominal, les différents cas alternatifs et d'erreur, tout en essayant d'ordonner séquentiellement les descriptions, afin d'améliorer leur lisibilité.

Comment les représenter ?

Le diagramme de cas d'utilisation est un schéma qui montre les cas d'utilisation (ovales) reliés par des associations (lignes) à leurs acteurs (icône du « stick man », ou représentation graphique équivalente). Chaque association signifie simplement « participe à ». Un cas d'utilisation doit être relié à au moins un acteur.

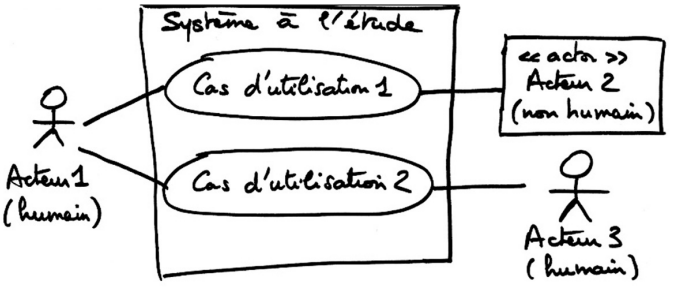


Figure 1-2.
Diagramme de cas d'utilisation

Une fois les cas d'utilisation identifiés, il faut encore les décrire !

Scénario

Un *scénario* représente une succession particulière d'enchaînements, s'exécutant du début à la fin du cas d'utilisation, un *enchaînement* étant l'unité de description de séquences d'actions. Un cas d'utilisation contient en général un scénario nominal et plusieurs scénarios alternatifs (qui se terminent de façon normale) ou d'erreur (qui se terminent en échec).

On peut d'ailleurs proposer une définition différente pour un cas d'utilisation : « ensemble de scénarios d'utilisation d'un système reliés par un but commun du point de vue d'un acteur ».

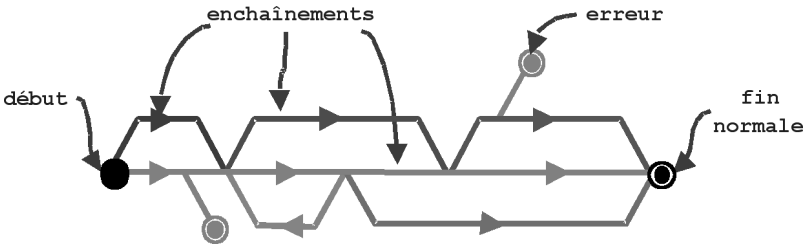


Figure 1-3. *Représentation des scénarios d'un cas d'utilisation*

La fiche de description textuelle d'un cas d'utilisation n'est pas normalisée par UML. Nous préconisons pour notre part la structuration suivante :

Sommaire d'identification (obligatoire)	Inclut titre, résumé, dates de création et de modification, version, responsable, acteurs...
Description des scénarios (obligatoire)	Décrit le scénario nominal, les scénarios (ou enchaînements) alternatifs, les scénarios (ou enchaînements) d'erreur, mais aussi les préconditions et les postconditions.
Exigences non-fonctionnelles (optionnel)	Ajoute, si c'est pertinent, les informations suivantes : fréquence, volumétrie, disponibilité, fiabilité, intégrité, confidentialité, performances, concurrence, etc. Précise également les contraintes d'interface homme-machine comme des règles d'ergonomie, une charte graphique, etc.

Étude d'un guichet automatique de banque

Cette étude de cas concerne un système simplifié de Guichet Automatique de Banque (GAB). Le GAB offre les services suivants :

1. Distribution d'argent à tout Porteur de carte de crédit, *via* un lecteur de carte et un distributeur de billets.
2. Consultation de solde de compte, dépôt en numéraire et dépôt de chèques pour les clients porteurs d'une carte de crédit de la banque adossée au GAB.

N'oubliez pas non plus que :

3. Toutes les transactions sont sécurisées.
4. Il est parfois nécessaire de recharger le distributeur, etc.

À partir de ces quatre phrases, nous allons progressivement :

- identifier les acteurs ;
- identifier les cas d'utilisation ;

- construire un diagramme de cas d'utilisation ;
- décrire textuellement les cas d'utilisation ;
- compléter les descriptions par des diagrammes dynamiques ;
- organiser et structurer les cas d'utilisation.

ATTENTION

L'énoncé précédent est volontairement incomplet et imprécis, comme il en est dans les projets réels !
Notez également que le problème et sa solution sont basés sur l'utilisation de cartes à puce dans le contexte des systèmes bancaires français. Si le GAB que vous avez l'habitude d'utiliser ne fonctionne pas exactement comme le nôtre, ce n'est pas très important. C'est surtout un prétexte pour vous montrer comment raisonner fonctionnellement avec les cas d'utilisation UML.

Étape 1 – Identification des acteurs du GAB

EXERCICE 1-1. Les acteurs

Identifiez les acteurs du GAB.

Quelles sont les entités externes qui interagissent directement avec le GAB ?

Considérons linéairement les phrases de l'énoncé.

La phrase 1 nous permet d'identifier immédiatement un premier acteur évident : tout « Porteur de carte». Il pourra uniquement utiliser le GAB pour retirer de l'argent avec sa carte.

En revanche, attention : le lecteur de carte et le distributeur de billets font partie du GAB. Ils ne peuvent donc pas être considérés comme des acteurs ! Vous pouvez noter ici que l'identification des acteurs oblige à fixer précisément la frontière entre le système à l'étude et son environnement. Si nous restreignions l'étude au système de contrôle-commande des éléments physiques du GAB, le lecteur de carte et le distributeur de billets deviendraient alors des acteurs.

Autre piège : la carte bancaire elle-même est-elle un acteur ? La carte est bien externe au GAB, et elle interagit avec lui… Pourtant, nous ne recommandons pas de la répertorier en tant qu'acteur, car nous appliquons le principe suivant : éliminer autant que possible les acteurs « physiques » au profit des acteurs « logiques ». L'acteur est celui qui bénéficie de l'utilisation du système. C'est bien le Porteur de carte qui retire de l'argent pour le dépenser ensuite, pas la carte !

La phrase 2 identifie des services supplémentaires qui ne sont proposés qu'aux clients de la banque porteurs d'une carte de crédit de cette dernière. Il s'agit donc d'un profil différent du précédent, que nous matérialisons par un deuxième acteur, appelé *Client banque*[1].

La phrase 3 nous incite à prendre en compte le fait que toutes les transactions sont sécurisées. Mais sécurisées par qui ? Pas par le GAB. Il existe donc d'autres entités externes qui jouent le rôle de Système d'autorisation et avec lesquelles le GAB communique directement. Une interview de l'expert métier est nécessaire, pour nous permettre d'identifier deux acteurs différents :

- le Système d'autorisation global Carte Bancaire, pour les transactions de retrait ;
- le Système d'information de la banque, pour autoriser toutes les transactions effectuées par un client avec sa carte de la banque, mais également pour accéder au solde des comptes.

Enfin, la phrase 4 nous rappelle qu'un GAB nécessite également des actions de maintenance, telles que le rechargement en billets du distributeur, la récupération des cartes avalées, etc. Ces actions de maintenance sont effectuées par un nouvel acteur, que nous appellerons pour simplifier : *Opérateur de maintenance*[2].

Plutôt que de répertorier simplement les acteurs textuellement, on peut réaliser un premier diagramme que nous appelons *diagramme de contexte statique*. Il suffit pour cela d'utiliser un diagramme de classes dans lequel chaque acteur est relié par une association à une classe centrale unique représentant le système, ce qui permet en outre de spécifier le nombre d'instances d'acteurs connectées au système à un moment donné.

Bien que ce diagramme ne fasse pas partie des diagrammes UML « officiels », nous l'avons très souvent trouvé utile dans notre expérience des projets réels.

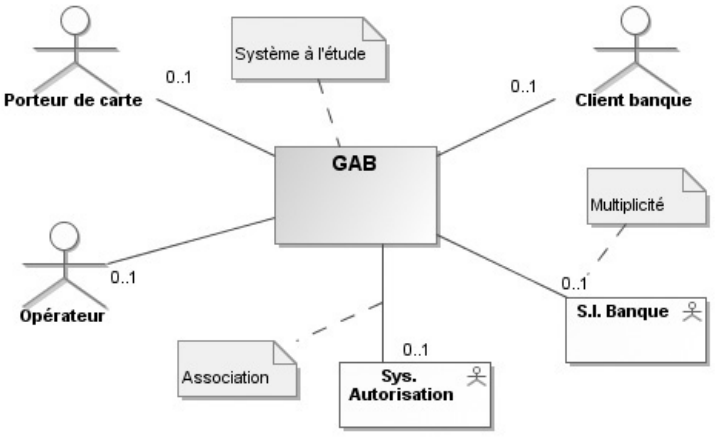
EXERCICE 1-2. Diagramme de contexte statique

Élaborez le diagramme de contexte statique du GAB.

Le GAB est un système fondamentalement mono-utilisateur : à tout instant, il n'y a qu'une instance de chaque acteur (au maximum) connectée au système.

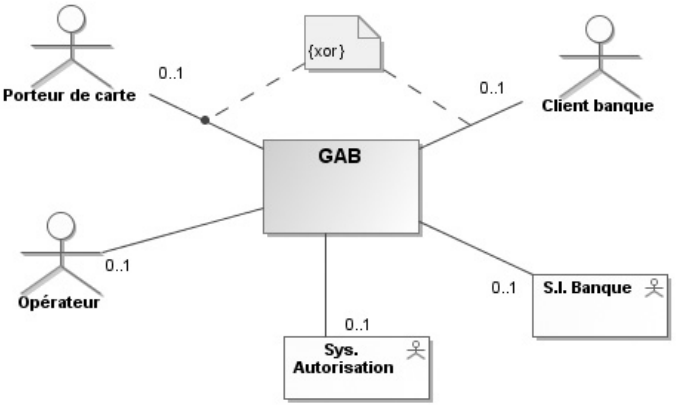
1. Il faudrait parler en toute rigueur de « Porteur de carte client de la banque », mais c'est un peu long, d'où la nécessité absolue de documenter tout élément UML, y compris les acteurs.
2. Nous pourrions par exemple identifier un acteur supplémentaire appelé « *Convoyeur* » ou « *Gabiste* », chargé spécifiquement de remplir la caisse du GAB.

Figure 1-4.
Diagramme de contexte statique du GAB



Il faudrait en toute rigueur ajouter une note graphique pour indiquer qu'en outre les acteurs humains *Client banque* et *Porteur de carte* sont mutuellement exclusifs, ce qui n'est pas implicite d'après les multiplicités des associations.

Figure 1-5.
Diagramme de contexte statique du GAB complété



Une autre solution, un peu plus élaborée, consiste à considérer que *Client banque* est une spécialisation de *Porteur de carte*, comme cela est illustré sur la figure suivante. Le problème précité d'exclusivité est ainsi résolu par construction, grâce à l'héritage entre acteurs.

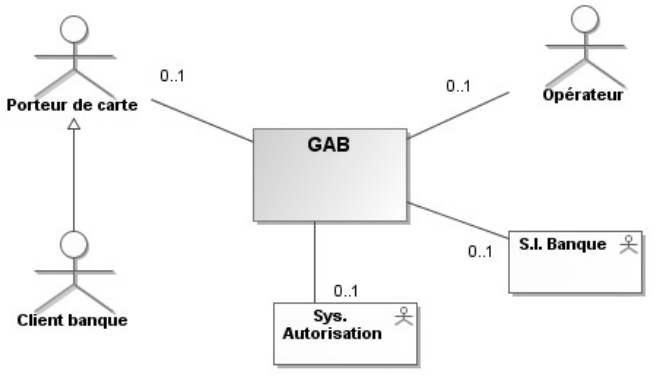


Figure 1-6.
Version plus élaborée du diagramme de contexte statique du GAB

Étape 2 – Identification des cas d'utilisation

EXERCICE 1-3. Cas d'utilisation

Préparez une liste préliminaire des cas d'utilisation du GAB, par acteur.

Reprenons un à un les cinq acteurs et listons les différentes façons qu'ils ont d'utiliser le GAB :

Porteur de carte :

- Retirer de l'argent.

Client banque :

- Retirer de l'argent (à ne pas oublier !).
- Consulter le solde de son compte courant.
- Déposer du numéraire.
- Déposer de l'argent (du numéraire ou des chèques)[3].

Opérateur de maintenance :

- Recharger le distributeur.
- Maintenir l'état opérationnel (récupérer les cartes avalées, récupérer les chèques déposés, remplacer le ruban de papier, etc.).

3. Il n'est pas nécessaire de répertorier deux cas d'utilisation distincts appelés *Déposer du numéraire* et *Déposer des chèques*. En effet, ils auraient le même acteur principal et le même objectif global. Cependant, cette décision pourra être remise en cause lors de la description détaillée des cas d'utilisation si l'on s'aperçoit alors que les scénarios sont trop différents. N'oubliez pas qu'une décision de modélisation ne doit jamais être irréversible !

Système d'autorisation (Sys. Auto.) :

- Néant.

Système d'information (SI) banque :

- Néant.

À RETENIR

Acteur principal ou secondaire

Contrairement à ce que l'on pourrait croire, tous les acteurs n'utilisent pas forcément le système ! Nous appelons acteur principal celui pour qui le cas d'utilisation produit un résultat observable. Par opposition, nous qualifions d'acteurs secondaires les autres participants du cas d'utilisation. Les acteurs secondaires sont souvent sollicités pour des informations complémentaires ; ils peuvent uniquement consulter ou informer le système lors de l'exécution du cas d'utilisation.

C'est exactement le cas des deux acteurs « non humains » de notre exemple : le Sys. Auto. et le SI banque sont uniquement sollicités par le GAB dans le cadre de la réalisation de certains cas d'utilisation. Mais ils n'ont pas eux-mêmes de façon propre d'utiliser le GAB, d'objectif à part entière.

Étape 3 – Réalisation de diagrammes de cas d'utilisation

On obtient sans difficulté un diagramme préliminaire en transcrivant la réponse précédente sur un schéma qui montre les cas d'utilisation (ovales) reliés par des associations (lignes) à leurs acteurs principaux (icône du « stick man »).

À RETENIR

Cadre de diagramme : tag et nom

Notez que depuis UML 2.0, un diagramme peut être inclus dans un cadre accueillant tout le contenu graphique. Le cadre a pour intitulé le nom du diagramme et établit sa portée. C'est un rectangle avec un petit pentagone (appelé tag de nom) placé dans l'angle supérieur gauche, qui contient le type du diagramme et son nom. Le cadre n'est cependant pas obligatoire lorsque le contexte est clair. La spécification UML définit les tags de chaque type de diagramme, mais cela n'a pas de caractère obligatoire et chaque outil a fait ses propres choix. Celui que nous avons utilisé majoritairement pour les nouveaux diagrammes UML 2 du livre, Enterprise Architect de la société Sparx Systems[a], a pris le parti de définir tous les tags avec deux lettres : `ud`, `cd`, `sd`, `td`, `ad`, `sm`, `id`, `dd`. L'autre outil que nous avons également utilisé, MagicDraw, prend des conventions parfois légèrement différentes. Dans le diagramme suivant, uc signifie *use case diagram.*

a. Le lecteur peut visiter le site web suivant http://www.sparxsystems.com.au/ où il trouvera une version d'évaluation du produit *Enterprise Architect*. Nous recommandons également l'introduction à UML 2 disponible sur le même site.

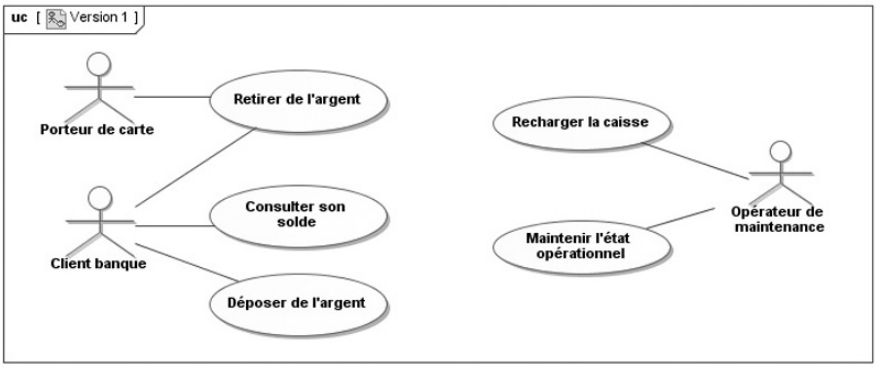


Figure 1-7. *Diagramme de cas d'utilisation préliminaire du GAB*

EXERCICE 1-4. Généralisation entre acteurs

Proposez une *autre* version, plus sophistiquée, de ce diagramme de cas d'utilisation préliminaire.

Le cas d'utilisation *Retirer de l'argent* a deux acteurs principaux possibles (mais exclusifs du point de vue de la simultanéité). Une autre façon de l'exprimer consiste à considérer l'acteur *Client de la banque* comme une spécialisation (au sens de la relation d'héritage) de l'acteur plus général *Porteur de carte*, comme nous l'avons déjà indiqué sur la figure 1-6. Un client de la banque est en effet un Porteur de carte particulier qui a toutes les prérogatives de ce dernier, ainsi que d'autres qui lui sont propres en tant que client.

UML permet de décrire une relation de généralisation/spécialisation entre acteurs, comme cela est indiqué sur le diagramme de cas d'utilisation suivant.

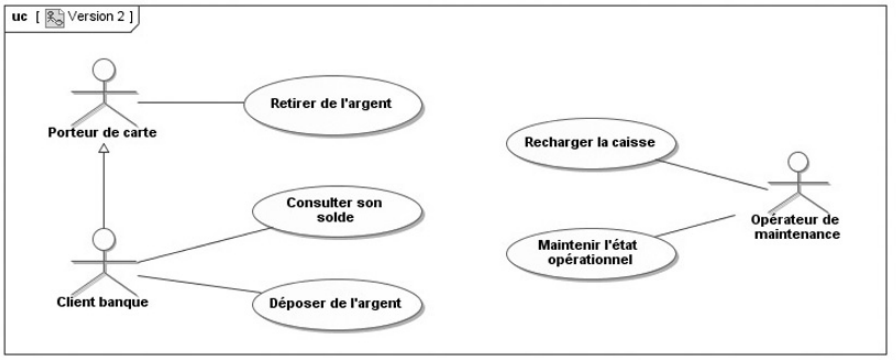


Figure 1-8. *Version plus sophistiquée du diagramme de cas d'utilisation préliminaire*

Cependant, dans notre exemple, l'intérêt de cette relation de généralisation n'est pas évident. Elle permet certes de supprimer l'association entre l'acteur *Client banque* et le cas d'utilisation *Retirer de l'argent*, qui est maintenant héritée de l'acteur *Porteur de carte*, mais ajoute en revanche le symbole de généralisation entre les deux acteurs… De plus, nous verrons au paragraphe suivant que les acteurs secondaires sollicités ne sont pas les mêmes dans le cas du Porteur de carte non client et dans celui du client de la banque.

Nous ne retiendrons donc pas cette solution et nous considérerons dans la suite du chapitre que les dénominations *Porteur de carte non client* et *Porteur de carte* sont synonymes.

Il nous reste maintenant à ajouter les acteurs secondaires pour compléter le diagramme de cas d'utilisation. Pour cela, UML propose simplement en standard de faire apparaître ces acteurs avec des associations supplémentaires vers les cas d'utilisation existants.

À RETENIR

Précisions graphiques au diagramme de cas d'utilisation

Pour notre part, afin d'améliorer le contenu informatif de ces diagrammes, nous recommandons d'adopter les conventions suivantes :
– par défaut, le rôle d'un acteur est « principal » ; si ce n'est pas le cas, indiquez explicitement que le rôle est « secondaire » sur l'association, du côté de l'acteur ;
– dans la mesure du possible, disposez les acteurs principaux à gauche des cas d'utilisation et les acteurs secondaires à droite.

EXERCICE 1-5. Acteurs secondaires

Complétez le diagramme de cas d'utilisation préliminaire en ajoutant les acteurs secondaires. Pour simplifier, ne tenez plus compte pour l'instant de l'opérateur de maintenance.

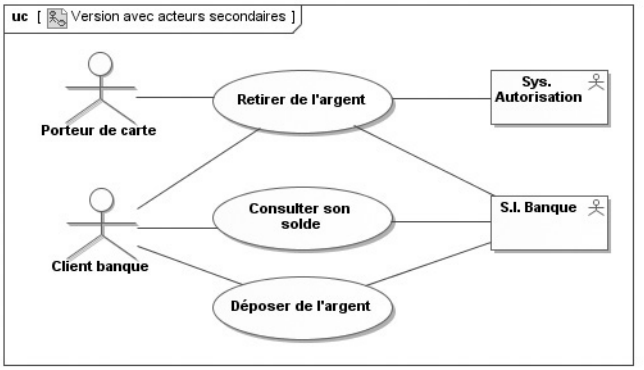
Pour tous les cas d'utilisation propres au client de la banque, il faut clairement faire intervenir comme acteur secondaire *SI banque*.

Mais un problème se pose pour le cas d'utilisation partagé *Retirer de l'argent*. En effet, si l'acteur principal est un Porteur de carte non client, il faudra faire appel au *Sys. Auto.* (qui se chargera ensuite de contacter le SI de la banque du porteur), alors que, s'il s'agit d'un client de la banque, le GAB contactera directement le *SI banque*.

Une première solution consiste à ajouter une association avec chacun des deux acteurs non-humains. Cette modélisation simpliste ne permet pas au lecteur du diagramme de

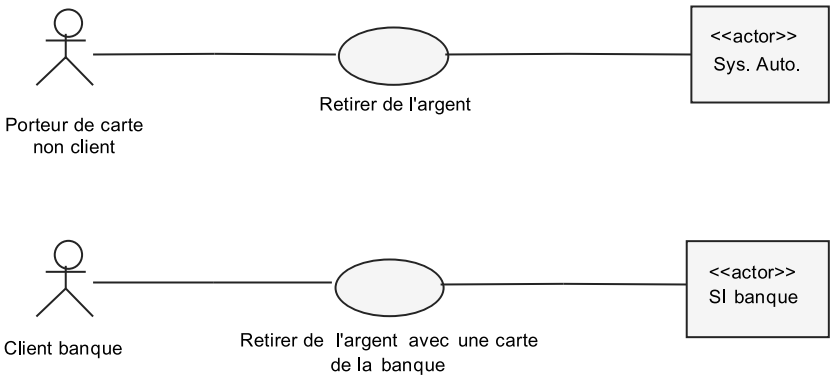
comprendre que les acteurs participent au cas d'utilisation *Retirer de l'argent* sélectivement deux par deux et non pas tous ensemble (figure 1-9).

Figure 1-9.
Version simple du diagramme de cas d'utilisation complété



Une autre solution consiste à distinguer deux cas d'utilisation pour le retrait d'argent : *Retirer de l'argent* et *Retirer de l'argent avec une carte de la banque*. Cette modélisation plus précise, mais plus lourde, est plus parlante pour l'expert métier. Elle milite d'ailleurs clairement contre l'utilisation de la généralisation entre acteurs évoquée précédemment. En effet, la distinction entre les deux cas d'utilisation est contradictoire avec la tentative d'héritage par l'acteur *Client banque* du cas unique *Retirer de l'argent*, qui avait été envisagée plus haut, alors que les acteurs secondaires n'avaient pas encore été ajoutés. Nous garderons cette seconde solution pour la suite de l'exercice[4] (figure 1-10).

Figure 1-10.
Fragment de la version plus précise du diagramme de cas d'utilisation complété



On notera que le *SI banque* n'est pas un acteur direct du cas d'utilisation *Retirer de l'argent*, car nous considérons que le *Sys. Auto.* se charge de le contacter, en dehors de la portée du GAB.

4. Il s'agit ici d'un choix de modélisation arbitraire ! Nous ne disons pas que toute autre solution serait mauvaise, mais nous expliquons avec des arguments concrets pourquoi nous préférons la nôtre.

Étape 4 – Description textuelle des cas d'utilisation

EXERCICE 1-6. Partie obligatoire du cas d'utilisation

Décrivez la partie obligatoire du cas d'utilisation RETIRER DE L'ARGENT (pour l'acteur non client de la banque).

Sommaire d'identification

Titre : Retirer de l'argent

Résumé : ce cas d'utilisation permet à un Porteur de carte, qui n'est pas client de la banque, de retirer de l'argent, si son crédit hebdomadaire le permet.

Acteurs : Porteur de carte (principal), *Système d'autorisation (secondaire).*

Date de création : 02/03/09 **Date de mise à jour** : 05/05/09

Version : 1.7 **Responsable** : Pascal Roques

Description des scénarios

Préconditions

- La caisse du GAB est alimentée (il reste au moins un billet !).
- Aucune carte ne se trouve déjà coincée dans le lecteur.
- La connexion avec le Système d'autorisation est opérationnelle.

Scénario nominal

1. Le Porteur de carte[5] introduit sa carte dans le lecteur de cartes du GAB.
2. Le GAB vérifie que la carte introduite est bien une carte bancaire.
3. Le GAB demande au Porteur de carte de saisir son code d'identification.
4. Le Porteur de carte saisit son code d'identification.
5. Le GAB compare le code d'identification avec celui qui est codé sur la puce de la carte.
6. Le GAB demande une autorisation au Système d'autorisation.
7. Le Système d'autorisation donne son accord et indique le crédit hebdomadaire.
8. Le GAB demande au Porteur de carte de saisir le montant désiré du retrait.
9. Le Porteur de carte saisit le montant désiré du retrait.

5. Nous préconisons de mettre systématiquement une majuscule au début du nom des acteurs pour améliorer la lisibilité du scénario nominal.

10. Le GAB contrôle le montant demandé par rapport au crédit hebdomadaire.
11. Le GAB demande au Porteur de carte s'il veut un ticket.
12. Le Porteur de carte demande un ticket.
13. Le GAB rend sa carte au Porteur de carte.
14. Le Porteur de carte reprend sa carte.
15. Le GAB délivre les billets et un ticket.
16. Le Porteur de carte prend les billets et le ticket.

Une autre présentation intéressante[6] consiste à séparer les actions des acteurs et du système en deux colonnes comme suit :

1. Le Porteur de carte introduit sa carte dans le lecteur de cartes du GAB.	2. Le GAB vérifie que la carte introduite est bien une carte bancaire. 3. Le GAB demande au Porteur de carte de saisir son code d'indentification.
4. Le Porteur de carte saisit son code d'identification.	5. Le GAB compare le code d'identification avec celui qui est codé sur la puce de la carte. 6. Le GAB demande une autorisation au Système d'autorisation.
7. Le Système d'autorisation donne son accord et indique le crédit hebdomadaire.	8. Le GAB demande au Porteur de carte de saisir le montant désiré du retrait.
9. Le Porteur de carte saisit le montant désiré du retrait.	10. Le GAB contrôle le montant demandé par rapport au crédit hebdomadaire. 11. Le GAB demande au Porteur de carte s'il veut un ticket.
12. Le Porteur de carte demande un ticket.	13. Le GAB rend sa carte au Porteur de carte.
14. Le Porteur de carte reprend sa carte.	15. Le GAB délivre les billets et un ticket.
16. Le Porteur de carte prend les billets et le ticket.	

6. Cette présentation a été recommandée par C. Larman dans la première version de *Applying UML and Patterns [Larman 97]*.

Enchaînements alternatifs[7]

A1 : code d'identification provisoirement erroné

L'enchaînement A1 démarre au point 5 du scénario nominal.

6. Le GAB indique au Porteur de carte que le code est erroné, pour la première ou deuxième fois.
7. Le GAB enregistre l'échec sur la carte.

Le scénario nominal reprend au point 3.

A2 : montant demandé supérieur au crédit hebdomadaire

L'enchaînement A2 démarre au point 10 du scénario nominal.

11. Le GAB indique au Porteur de carte que le montant demandé est supérieur au crédit hebdomadaire.

Le scénario nominal reprend au point 8.

A3 : ticket refusé

L'enchaînement A3 démarre au point 11 du scénario nominal.

12. Le Porteur de carte refuse le ticket.
13. Le GAB rend sa carte au Porteur de carte.
14. Le Porteur de carte reprend sa carte.
15. Le GAB délivre les billets.
16. Le Porteur de carte prend les billets.

Enchaînements d'erreur

E1 : carte non-valide

L'enchaînement E1 démarre au point 2 du scénario nominal.

3. Le GAB indique au Porteur que la carte n'est pas valide (illisible, périmée, etc.), la confisque ; le cas d'utilisation se termine en échec.

E2 : code d'identification définitivement erroné

L'enchaînement E2 démarre au point 5 du scénario nominal.

6. Le GAB indique au Porteur de carte que le code est erroné, pour la troisième fois.
7. Le GAB confisque la carte.
8. Le Système d'autorisation est informé ; le cas d'utilisation se termine en échec.

7. Nous distinguons les enchaînements alternatifs (Ax) qui reprennent ensuite à une étape du scénario nominal des enchaînements d'erreur (Ey) qui terminent brutalement le cas d'utilisation en échec. L'objectif de l'acteur principal est donc atteint par les scénarios nominaux et alternatifs mais pas par ceux d'erreur.

E3 : retrait non autorisé

L'enchaînement E3 démarre au point 6 du scénario nominal.

7. Le Système d'autorisation interdit tout retrait.
8. Le GAB éjecte la carte ; le cas d'utilisation se termine en échec.

E4 : carte non reprise

L'enchaînement E4 démarre au point 13 du scénario nominal.

14. Au bout de 10 secondes, le GAB confisque la carte.
15. Le Système d'autorisation est informé ; le cas d'utilisation se termine en échec.

E5 : billets non pris

L'enchaînement E5 démarre au point 15 du scénario nominal.

16. Au bout de 10 secondes, le GAB reprend les billets.
17. Le cas d'utilisation se termine en échec.

E6 : annulation de la transaction

L'enchaînement E6 peut démarrer entre les points 4 et 12 du scénario nominal.

4 à 12. Le Porteur de carte demande l'annulation de la transaction en cours.
Le GAB éjecte la carte ; le cas d'utilisation se termine en échec.

Une autre présentation intéressante des enchaînements alternatifs et d'erreur consiste à utiliser les conventions préconisées par A. Cockburn[8]. Celui-ci propose d'indiquer les différentes alternatives par des lettres collées au chiffre du numéro de l'étape du scénario nominal concernée. Une version alternative de la solution précédente pourrait être alors :

2a. Carte illisible ou non valable :

Le GAB avertit le Porteur et éjecte la carte ; le cas d'utilisation se termine en échec.

2b. Carte périmée :

Le GAB avertit le Porteur et confisque la carte ; le cas d'utilisation se termine en échec.

4a. Délai de saisie du code expiré :

Le GAB avertit le porteur et éjecte la carte ; le cas d'utilisation se termine en échec.

4-12a.[9] Le Porteur annule la transaction :

Le GAB éjecte la carte ; le cas d'utilisation se termine en échec.

8. Le lecteur se référera avec profit à l'excellent ouvrage d'Alistair Cockburn : *Rédiger des cas d'utilisation efficaces*, éditions Eyrolles 2001 [Cockburn 01].
9. La notation 4-12 signifie : « de l'étape 4 à l'étape 12 ».

5a. Code d'identification erroné pour la première ou deuxième fois :

5a1. Le GAB enregistre l'échec sur la carte.

5a2. Le GAB avertit le Porteur et le scénario nominal reprend à l'étape 3.

5b. Code d'identification erroné pour la troisième fois :

Le GAB avertit le Porteur et confisque la carte ; le cas d'utilisation se termine en échec.

7a. Transaction refusée par le Système d'autorisation :

Le GAB avertit le Porteur et éjecte la carte ; le cas d'utilisation se termine en échec.

7b. Délai de réponse du Système d'autorisation expiré :

Le GAB avertit le Porteur et éjecte la carte ; le cas d'utilisation se termine en échec.

9a. Délai de saisie du montant expiré :

Le GAB avertit le Porteur et éjecte la carte ; le cas d'utilisation se termine en échec.

10a. Montant demandé supérieur au crédit hebdomadaire :

Le GAB avertit le Porteur et le scénario nominal reprend à l'étape 8.

10b. Crédit hebdomadaire insuffisant :

Le GAB avertit le Porteur et éjecte la carte ; le cas d'utilisation se termine en échec.

12a. Le Porteur ne demande pas de ticket :
Le cas d'utilisation continue à l'identique, sauf l'impression du ticket.

14a. Délai de retrait de la carte expiré :

14a1. Le GAB confisque la carte et annule la transaction ;

14a2. Le GAB avertit le Système d'autorisation et le cas d'utilisation se termine en échec.

16a. Délai de retrait des billets expiré :

Le GAB confisque les billets et annule la transaction ; le cas d'utilisation se termine en échec.

1-7a. Coupure réseau avec le Système d'autorisation :

Le GAB avertit le Porteur et éjecte la carte ; le cas d'utilisation se termine en échec.

Postconditions

La caisse du GAB contient moins de billets qu'au début du cas d'utilisation (le nombre de billets manquants est fonction du montant du retrait).