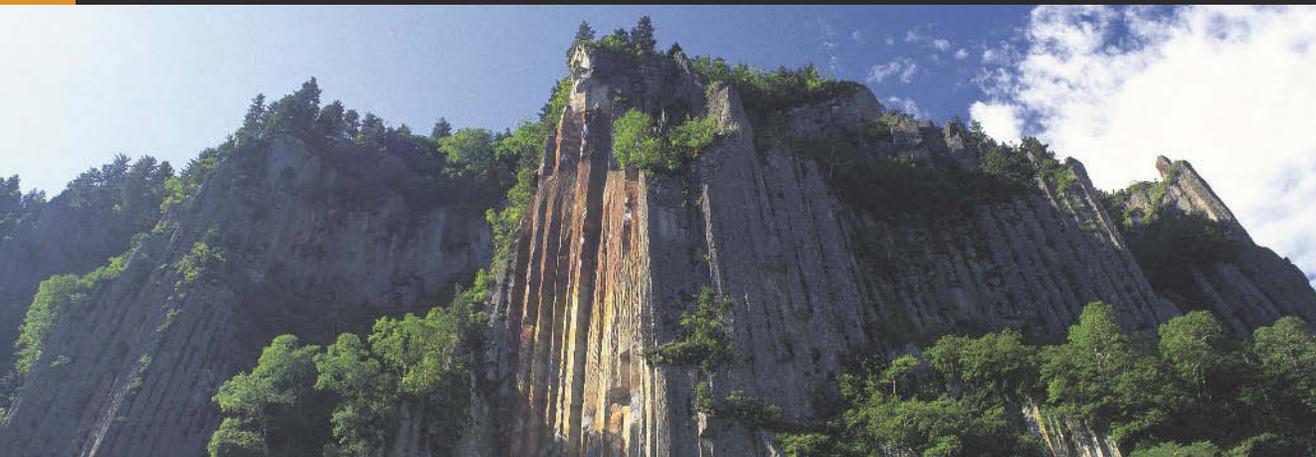


UML 2

ANALYSE ET CONCEPTION

Mise en œuvre guidée
avec études de cas



Joseph Gabay
David Gabay

DUNOD

UML 2

ANALYSE ET CONCEPTION

**Mise en œuvre guidée
avec études de cas**

Joseph Gabay

*Directeur de projet informatique au CNRS
Chargé de cours à l'université de Paris-Dauphine*

David Gabay

Chef de projet chez Cap Gemini

DUNOD

Toutes les marques citées dans cet ouvrage sont des marques déposées par leurs propriétaires respectifs.

Illustration de couverture : Mountain, DAJ, Hokkaido
Source : gettyimages®

Le pictogramme qui figure ci-contre mérite une explication. Son objet est d'alerter le lecteur sur la menace que représente pour l'avenir de l'écrit, particulièrement dans le domaine de l'édition technique et universitaire, le développement massif du photocopillage.

Le Code de la propriété intellectuelle du 1^{er} juillet 1992 interdit en effet expressément la photocopie à usage collectif sans autorisation des ayants droit. Or, cette pratique s'est généralisée dans les établissements

d'enseignement supérieur, provoquant une baisse brutale des achats de livres et de revues, au point que la possibilité même pour

les auteurs de créer des œuvres nouvelles et de les faire éditer correctement est aujourd'hui menacée. Nous rappelons donc que toute reproduction, partielle ou totale, de la présente publication est interdite sans autorisation de l'auteur, de son éditeur ou du

Centre français d'exploitation du droit de copie (CFC, 20, rue des Grands-Augustins, 75006 Paris).



© Dunod, Paris, 2008

ISBN 978-2-10-053567-5

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5, 2° et 3° a), d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite » (art. L. 122-4).

Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

Tables des matières

Avant-propos	IX
Chapitre 1 – Concepts de l’approche objet et présentation d’UML 2	1
1.1 Concepts de l’approche objet	1
1.1.1 <i>Objet et classe</i>	2
1.1.2 <i>Encapsulation et interface</i>	3
1.1.3 <i>Association et agrégation entre les classes</i>	3
1.1.4 <i>Généralisation et spécialisation de classe</i>	4
1.1.5 <i>Polymorphisme</i>	4
1.1.6 <i>Persistance</i>	5
1.1.7 <i>Avantages du développement à l’aide des langages objet</i>	6
1.2 Présentation générale d’UML	6
1.2.1 <i>Historique</i>	6
1.2.2 <i>Structuration de la présentation</i>	7
1.2.3 <i>Règles générales</i>	8
1.2.4 <i>Présentation générale des diagrammes</i>	11
1.2.5 <i>Schéma d’ensemble des treize diagrammes d’UML 2</i>	14
Chapitre 2 – Les diagrammes structurels (ou statiques)	17
2.1 Diagramme de classe (DCL) et diagramme d’objet (DOB)	17
2.1.1 <i>Objet</i>	17
2.1.2 <i>Classe, attribut et opération</i>	18

2.1.3	Association, multiplicité, navigabilité et contraintes	23
2.1.4	Agrégation et composition entre classes	27
2.1.5	Association qualifiée, dépendance et classe d'interface	30
2.1.6	Généralisation et spécialisation	32
2.1.7	Stéréotype de classe	36
2.1.8	Exercices.	36
2.2	Diagramme de composant (DCP).	46
2.2.1	Composant	46
2.2.2	Les deux types de représentation et exemples	46
2.3	Diagramme de déploiement (DPL).	50
2.3.1	Nœud.	50
2.3.2	Artefact	51
2.3.3	Spécification de déploiement.	51
2.3.4	Liens entre un artefact et les autres éléments du diagramme	52
2.3.5	Représentation et exemples	53
2.4	Diagramme de paquetage (DPA)	54
2.4.1	Paquetage	54
2.4.2	Dépendance entre paquetages	56
2.4.3	Représentation et exemples	56
2.5	Diagramme de structure composite (DSC).	58
2.5.1	Collaboration	58
2.5.2	Représentation et exemples	58
Chapitre 3 – Les diagrammes comportementaux		61
3.1	Diagramme des cas d'utilisation (DCU).	61
3.1.1	Présentation générale et concepts de base	61
3.1.2	Représentation du diagramme des cas d'utilisation	63
3.1.3	Relations entre cas d'utilisation	64
3.1.4	Description textuelle d'un cas d'utilisation	66
3.1.5	Exercices	67
3.2	Diagramme d'état-transition (DET).	72
3.2.1	Présentation générale et concepts de base	72
3.2.2	Représentation du diagramme d'état-transition d'un objet	73
3.2.3	Compléments sur le diagramme d'état-transition	75
3.2.4	Exercices.	78

3.3	Diagramme d'activité (DAC).....	80
3.3.1	Présentation générale et concepts de base.....	80
3.3.2	Représentation du diagramme d'activité.....	87
3.3.3	Exercices.....	88
3.4	Diagramme de séquence (DSE).....	90
3.4.1	Présentation générale et concepts de base.....	90
3.4.2	Opérations particulières.....	91
3.4.3	Fragment d'interaction.....	93
3.4.4	Autre utilisation du diagramme de séquence.....	101
3.4.5	Exercices.....	102
3.5	Diagramme de communication (DCO).....	104
3.5.1	Présentation générale et concepts de base.....	104
3.5.2	Formalisme et exemple.....	105
3.5.3	Exercices.....	106
3.6	Diagramme global d'interaction (DGI).....	106
3.6.1	Présentation générale et concepts de base.....	106
3.6.2	Représentation et exemple.....	108
3.7	Diagramme de temps (DTP).....	109
3.7.1	Présentation générale et concepts de base.....	109
3.7.2	Représentation et exemples.....	109
	Chapitre 4 – Démarche de développement.....	111
4.1	Présentation d'UP.....	111
4.2	Les principes d'UP.....	112
4.2.1	Processus guidé par les cas d'utilisation.....	112
4.2.2	Processus itératif et incrémental.....	112
4.2.3	Processus centré sur l'architecture.....	112
4.2.4	Processus orienté par la réduction des risques.....	113
4.3	Les concepts et les deux dimensions du processus UP.....	113
4.3.1	Définition des principaux concepts et schéma d'ensemble.....	113
4.3.2	Phases et itérations du processus (aspect dynamique).....	114
4.3.3	Activités du processus (aspect statique).....	116
4.4	Les principaux apports de RUP.....	117
4.4.1	Les bonnes pratiques.....	118
4.4.2	Les phases et les activités du processus.....	119

4.5	Démarche de développement UP7	123
4.5.1	Présentation générale	123
4.5.2	Description des activités (fiche guide par sous-activité)	128
4.5.3	Compléments sur la conception	146
Chapitre 5	– Étude de cas n° 1 Analyse	149
5.1	Énoncé du cas ALLOC	149
5.2	Modélisation métier	150
5.2.1	Élaboration du schéma de contexte du domaine d'étude (FG1)	150
5.2.2	Élaboration du diagramme d'activité (FG2)	150
5.2.3	Élaboration du diagramme de classe métier (FG3)	151
5.2.4	Extrait des documents de cadrage	152
5.3	Exigences fonctionnelles	153
5.3.1	Élaboration du diagramme des cas d'utilisation système (FG4)	153
5.3.2	Élaboration du diagramme de séquence système (FG5)	155
5.3.3	Élaboration du schéma de navigation générale (FG6)	158
5.4	Analyse des cas d'utilisation	159
5.4.1	Élaboration du diagramme des cas d'utilisation (FG7)	159
5.4.2	Description des cas d'utilisation (FG8, FG9, FG11, FG12)	159
5.5	Synthèse de l'analyse	172
Chapitre 6	– Étude de cas n° 2 Analyse et conception	175
6.1	Énoncé du cas Gestion activité et frais	175
6.2	Modélisation métier	176
6.2.1	Élaboration du schéma de contexte du domaine d'étude (FG1)	176
6.2.2	Élaboration du diagramme d'activité (FG2)	176
6.2.3	Élaboration du diagramme de classe métier (FG3)	177
6.3	Exigences fonctionnelles	181
6.3.1	Élaboration du diagramme des cas d'utilisation système (FG4)	181
6.3.2	Élaboration des diagrammes de séquence système (FG5)	182
6.3.3	Élaboration du schéma de navigation générale (FG6)	184
6.4	Analyse des cas d'utilisation	185
6.4.1	Élaboration du diagramme des cas d'utilisation (FG7)	185
6.4.2	Description des cas d'utilisation (FG8, FG9, FG11, FG12)	186

6.5	Synthèse de l'analyse	202
6.5.1	Élaboration du diagramme de classe récapitulatif (FG13)	202
6.5.2	Élaboration de la matrice de validation (FG14)	204
6.6	Conception	204
6.6.1	Réalisation des choix techniques et élaboration des diagrammes techniques (FG15, FG16, FG17)	204
6.6.2	Élaboration du diagramme de paquetage (FG18)	216
Annexes	219
A.	Récapitulatif des concepts d'UML 2	219
B.	Récapitulatif de la démarche UP7	222
Bibliographie	223
Index	225

Avant-propos

UML, une évolution majeure dans le domaine des méthodes

UML compte déjà une dizaine d'années d'existence. À l'échelle d'un courant méthodologique, c'est encore une durée relativement courte puisque l'on estime qu'un cycle de développement d'une méthode de cette envergure s'étale sur une période de vingt à trente ans, ce qui a été le cas par exemple pour Merise.

Mais l'accélération du renouvellement des technologies conjuguée avec la pression économique et concurrentielle qui s'exerce sur les entreprises, obligent les acteurs du monde informatique à produire des solutions de plus en plus rapidement dans un contexte d'amélioration continue de la qualité et de la performance des systèmes d'information.

Notons aussi qu'Internet a été un vecteur favorisant le développement de très nombreuses applications dont une grande partie utilise des solutions à base de langage de programmation objet comme Java, C++ ou C#.

UML a apporté tout naturellement le support méthodologique qui manquait à tous les concepteurs et développeurs qui voulaient formaliser l'analyse et la conception technique de leur logiciel.

UML s'est donc imposée en tant que langage graphique de modélisation puisque non seulement ce langage répond à un véritable besoin mais en outre il est devenu un standard de fait puisqu'il s'appuie sur une norme très structurante.

Rendons tout de même hommage aux trois pères fondateurs d'UML que sont James Rumbaugh, Ivar Jacobson et Grady Booch qui ont été dès le début des années 90 des références dans le monde des méthodes de développement objets. Les ouvrages qu'ils ont écrits à l'époque sont là pour en témoigner : [Rumbaugh1991], [Booch1994] et [Jacobson1992].

C'est grâce à un premier niveau de travail de fond mené en commun par ces trois compères qu'est née en 1995 la méthode dite unifiée qui a été ensuite consacrée par l'OMG (*Object Management Group*) en 1997 avec la diffusion de la première version de la norme : UML V1.1.

Précisons aussi que les trois auteurs à l'origine d'UML ont poursuivi leur mission d'explication et d'illustration des différentes versions successives d'UML en publiant des ouvrages de référence comme [Jacobson2000b] et [Rumbaugh2004].

Il nous semble important de souligner le rôle majeur joué par l'OMG. En effet, cet organisme international de normalisation a en charge non seulement la norme UML mais aussi d'autres réflexions méthodologiques comme l'approche MDA (*Model Driven Architecture*) qui pousse encore plus loin les limites de l'automatisation de la production du logiciel.

Ainsi nous pouvons imaginer que nous arriverons bien un jour à disposer d'une méthode de conception et de développement standardisée couvrant tout le cycle de fabrication d'un logiciel en permettant une production fortement automatisée de la programmation.

Mais soyons réalistes, cela demandera à notre avis probablement plusieurs décennies étant donné les difficultés qui restent à traiter et la complexité des niveaux d'abstraction qu'il faut arriver à modéliser.

C'est l'occasion pour nous de lancer un petit avertissement aux concepteurs d'UML travaillant à l'OMG pour leur dire que certes l'objectif visé représente un énorme challenge pour toute la profession informatique, mais cela ne doit pas se traduire par une plus grande lourdeur et complexité du contenu de cette norme. En effet, c'est le ressenti que nous commençons à avoir en observant les versions successives de la norme qui se caractérise aujourd'hui par plusieurs centaines de pages à lire et un nombre sans cesse croissant de concepts à assimiler associés à une représentation graphique qui devient aussi de plus en plus dense.

Positionnement de l'ouvrage

Notre étude des nombreux ouvrages déjà publiés sur UML 2, nous a permis de constater qu'il en existait déjà un certain nombre qui s'était attaché à une présentation relativement exhaustive et détaillée de la norme, comme par exemple dans [Muller2000] et [Fowler2004a] ou encore d'autres plus synthétiques comme [Scott2004], [Fowler2004b], [Barbier2005] et [Blaha2002].

Cependant, nous n'avons pas trouvé de livres traitant à la fois l'aspect normatif d'UML 2 et la démarche d'élaboration des diagrammes couvrant l'analyse et la conception des systèmes d'information.

Nous avons donc décidé de répondre à ce besoin en essayant de traiter le plus efficacement possible les treize diagrammes d'UML 2 conformément à la norme et en accompagnant le lecteur dans un apprentissage progressif fondé sur de nombreux exemples, des exercices corrigés et de véritables études de cas se rapprochant de projets réels d'entreprise.

Nous proposons donc dans un même ouvrage d'une part l'aspect théorique des concepts d'UML 2 et leur représentation graphique et d'autre part une démarche de mise en œuvre illustrée par des fiches guides pour les activités d'analyse et de conception à mener dans le cadre des développements des systèmes d'information (SI).

En résumé nous nous sommes fixé trois objectifs en réalisant ce livre :

- présenter les treize diagrammes d'UML 2 en essayant de concilier au mieux le respect strict de la norme avec une application centrée sur les systèmes d'information des entreprises.
- illustrer la modélisation à l'aide des diagrammes d'UML 2 en s'appuyant sur des exemples et des exercices adaptés au contexte professionnel donc aux attentes des concepteurs et développeurs d'application.
- proposer une démarche de mise en œuvre d'UML 2 qui est fondée sur les processus standard du développement itératif et incrémental et qui prenne en compte notre propre expérience de praticiens de la méthode. Cette démarche fait l'objet d'une description précise des activités avec notamment des fiches guides et une mise en application dans deux études de cas conséquentes.

Notre double compétence de professionnel de la conception et du développement des systèmes d'information en entreprise, et d'enseignant universitaire dans le domaine des méthodes des SI nous a permis de faire bénéficier l'ouvrage de notre grande pratique (dix années) d'UML, et de l'expérience tirée de nombreuses années d'enseignements d'UML dispensés en milieu universitaire (MIAGE, MASTER, IUT, BTS...).

En tant qu'enseignant d'UML nous avons pu, au fil des années, affiner une démarche progressive d'apprentissage. C'est ainsi que pour les points délicats, nous avons toujours recherché une approche pragmatique s'appuyant sur des exemples pour accompagner la présentation théorique des concepts.

En tant que professionnel, les enseignements d'une longue expérience de la pratique des méthodes auprès des équipes de développement de projets ont été particulièrement précieux. De plus, le point de vue des utilisateurs a été aussi un indicateur pertinent pour ajuster au mieux la pratique de ces méthodes.

Enfin nous restons intimement convaincus que l'exposé théorique d'une méthode doit être réduit à l'essentiel et qu'au contraire une large place doit être donnée à l'application ; c'est ainsi qu'en matière d'apprentissage d'une méthode, il faut bien entendu apprendre pour pratiquer mais aussi et peut-être plus que dans n'importe quel autre domaine : pratiquer pour mieux apprendre.

Organisation de l'ouvrage

L'ouvrage est structuré en six chapitres :

- Le chapitre 1 décrit les concepts de l'approche objet et propose une première présentation d'UML 2 en mettant l'accent sur les dernières évolutions.

- Le chapitre 2 traite les six diagrammes structurels : diagramme de classe, diagramme d'objet, diagramme de composant, diagramme de déploiement, diagramme de paquetage et diagramme de structure composite.
- Le chapitre 3 est lui consacré aux sept diagrammes comportementaux : diagramme des cas d'utilisation, diagramme d'activité, diagramme d'état-transition, diagramme de séquence, diagramme de communication, diagramme global d'interaction et diagramme de temps.

Des exemples et exercices corrigés sont associés à la présentation de la plupart des 13 diagrammes. Nous avons aussi utilisé, en tant que fil conducteur, un même exercice « Locagite » pour les diagrammes les plus structurants (classe, cas d'utilisation et séquence).

- Le chapitre 4 porte sur la démarche que nous proposons de mettre en œuvre avec UML 2 en vue de traiter l'analyse et la conception de système d'information. Cette démarche prend appui sur le processus unifié UP (*Unified Process*) et intègre le fruit de l'expérience des auteurs dans la conduite de projets réels menés en entreprise.
Nous avons privilégié une présentation sous forme de « fiches guides » pour couvrir la démarche d'analyse et de conception.
- Les chapitres 5 et 6 sont consacrés aux études de cas afin d'illustrer, sur des sujets plus conséquents que de simples exercices, le langage de formalisation d'UML 2 d'une part et l'application de la démarche proposée dans cet ouvrage d'autre part.
La première étude de cas (chapitre 5) est essentiellement dédiée à l'étape d'analyse, tandis que la seconde (chapitre 6) couvre les étapes d'analyse et de conception.

En résumé, le lecteur pourra tout d'abord acquérir les connaissances nécessaires à l'apprentissage d'UML 2 (chapitres 1 à 3) et ensuite s'exercer à la mise en pratique grâce à la démarche proposée et aux deux études de cas couvrant l'ensemble des phases d'analyse et de conception (chapitres 4 à 6).

À qui s'adresse ce livre ?

Cet ouvrage de synthèse sur les concepts, la représentation graphique des diagrammes d'UML 2 et la mise en œuvre guidée en analyse et conception s'adresse :

- aux étudiants de premier et second cycle universitaire qui veulent s'initier à UML 2 et maîtriser tous les concepts ;
- à tous ceux qui connaissent déjà UML et qui désirent comprendre les changements apportés par UML 2 ;
- à tous les professionnels, concepteurs et développeurs, qui souhaitent mieux maîtriser UML 2 et acquérir une démarche pratique de mise en œuvre.

Remerciements

Qu'il nous soit permis de remercier tous ceux qui ont apporté une contribution dans la réalisation de cet ouvrage.

Plus particulièrement, nous voudrions remercier Jacques LAVIELLE pour les échanges fructueux sur les concepts et la démarche et pour toutes les discussions que nous avons eues autour des enseignements d'UML à l'université Paris-Dauphine.

Enfin, nous adressons tous nos remerciements à Nicolas ZENOU pour son aide précieuse et efficace dans la relecture attentive qu'il a bien voulu consacrer à cet ouvrage.

1

Concepts de l'approche objet et présentation d'UML 2

1.1 CONCEPTS DE L'APPROCHE OBJET

Aujourd'hui l'approche objet occupe une place prépondérante dans le génie logiciel. En effet, ces dernières années nous avons assisté tout d'abord à une utilisation plus large des langages de programmation objet de référence comme C++, C# et Java et ensuite à l'introduction des concepts objet dans d'autres langages comme par exemple VB.NET, Perl et même Cobol. Le développement très important d'applications liées à Internet constitue une des principales explications de ce phénomène sans précédent alors que l'on aurait pu s'attendre à un démarrage plus précoce de ce changement compte tenu de l'existence dès le début des années 80 des premiers langages objet tel que C++.

À notre avis les deux événements majeurs qui ont marqué cette évolution se sont produits à la fin des années 90 avec l'arrivée de Java en 1995 et d'UML en 1997.

Notre objectif est donc de présenter l'essentiel des concepts objet qui nous paraissent nécessaires à une bonne compréhension d'UML. Les concepts qui nous semblent importants à bien maîtriser sont les suivants :

- objet et classe,
- encapsulation et interface,
- association et agrégation de classes,
- généralisation et spécialisation de classe,
- polymorphisme,
- persistance.

Nous n'utiliserons pas, par contre dans ce chapitre, de formalisme particulier de représentation puisque nous ne voulons pas ajouter un formalisme de plus à celui déjà défini dans UML.

Nous précisons que nous nous limitons volontairement, dans le cadre de cet ouvrage, à une présentation générale des principaux concepts de l'approche objet. Le lecteur qui désire approfondir ses connaissances dans ce domaine pourra se reporter aux nombreux ouvrages traitant en détail l'approche objet comme [Meyer2000]et [Bersini2004].

1.1.1 Objet et classe

Concept d'objet

Un **objet** représente une entité du monde réel (ou du monde virtuel pour les objets immatériels) qui se caractérise par un ensemble de propriétés (attributs), des états significatifs et un comportement.

L'**état** d'un objet correspond aux valeurs de tous ses attributs à un instant donné. Les propriétés sont définies dans la classe d'appartenance de l'objet. Le comportement d'un objet est caractérisé par l'ensemble des opérations qu'il peut exécuter en réaction aux messages provenant des autres objets. Les opérations sont définies dans la classe d'appartenance de l'objet.

Exemple

Considérons l'employé Durand, n° 1245, embauché en tant qu'ingénieur travaillant sur le site N.

Cet objet est caractérisé par la liste de ses attributs et son état est représenté par les valeurs de ses attributs :

- n° employé : 1245,
- nom : Durand,
- qualification : ingénieur,
- lieu de travail : site N.

Son comportement est caractérisé par les opérations qu'il peut exécuter. Dans notre cas nous pouvons avoir les opérations suivantes :

- entrer dans l'organisme,
- changer de qualification,
- changer de lieu de travail,
- sortir de l'organisme.

Concept de classe

Une **classe** est l'abstraction d'un ensemble d'objets qui possèdent une structure identique (liste des attributs) et un même comportement (liste des opérations).

Un **objet** est une instance d'une et une seule classe. Une **classe abstraite** est une classe qui n'a pas d'instance. Les concepts de classe et d'objet sont interdépendants.

Exemple

Considérons la classe Employé qui représente l'ensemble des employés d'une entreprise. La description de la classe Employé comportera les éléments suivants :

- Nom de classe : Employé.
- Attributs :
 - numéro,
 - nom,
 - qualification,
 - site de travail.
- Opérations :
 - engager un employé,
 - consulter un employé,
 - modifier un employé,
 - départ d'un employé.

1.1.2 Encapsulation et interface

Par rapport à l'approche classique, l'approche objet se caractérise par le regroupement dans une même classe de la description de la structure des attributs et de la description des opérations. Ce regroupement des deux descriptions porte le nom d'**encapsulation** données-traitements.

Plus précisément, les données ne sont accessibles qu'à partir d'opérations définies dans la classe. Le principe d'encapsulation renforce l'autonomie et l'indépendance de chaque classe et donne une potentialité accrue de définition de classe réutilisable. L'ensemble des opérations d'une classe rendu visible aux autres classes porte le nom d'**interface**. Le schéma d'ensemble du principe de l'encapsulation est présenté à la figure 1.1.

1.1.3 Association et agrégation entre les classes

L'**association** représente une relation entre plusieurs classes. Elle correspond à l'abstraction des liens qui existent entre les objets dans le monde réel. Les multiplicités (ou cardinalités) et les rôles des objets participant aux relations complètent la description d'une association. Les exemples d'associations sont donnés directement dans les diagrammes de classe d'UML.

L'**agrégation** est une forme particulière d'association entre plusieurs classes. Elle exprime le fait qu'une classe est composée d'une ou plusieurs autres classes. La relation composant-composé ou la relation structurelle représentant l'organigramme d'une entreprise sont des exemples types de la relation d'agrégation.

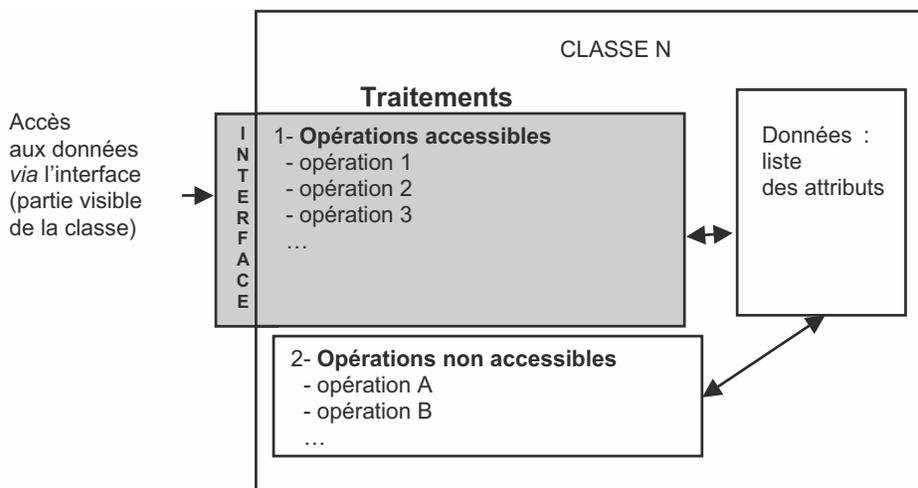


Figure 1.1 — Schéma de principe de l'encapsulation

1.1.4 Généralisation et spécialisation de classe

La **généralisation** de classes consiste à factoriser dans une classe, appelée **super-classe**, les attributs et/ou opérations des classes considérées. Appliquée à l'ensemble des classes, elle permet de réaliser une hiérarchie des classes.

La **spécialisation** représente la démarche inverse de la généralisation puisqu'elle consiste à créer à partir d'une classe, plusieurs classes spécialisées.

Chaque nouvelle classe créée est dite spécialisée puisqu'elle comporte en plus des attributs ou opérations de la super-classe (disponibles par héritage) des attributs ou opérations qui lui sont propres.

Une classe spécialisée porte aussi le nom de **sous-classe**. La spécialisation de classe se construit en deux temps : d'abord par héritage des opérations et des attributs d'une super-classe et ensuite par ajout d'opérations et/ou d'attributs spécifiques à la sous-classe.

La **généralisation-spécialisation** est un des mécanismes les plus importants de l'approche objet qui facilite la réutilisation des classes.

1.1.5 Polymorphisme

Le **polymorphisme** est la capacité donnée à une même opération de s'exécuter différemment suivant le contexte de la classe où elle se trouve.

Ainsi une opération définie dans une super-classe peut s'exécuter de manière différente selon la sous-classe où elle est héritée.

En fait lors de l'exécution, l'appel de l'opération va automatiquement déclencher l'exécution de l'opération de la sous-classe concernée. Dans le déroulement de l'exécution de l'opération de la sous-classe, il est possible de faire appel à l'opération de la super-classe qui contient en général la partie commune applicable aux deux sous-classes.

Exemple

Soit la classe Employé et ses deux sous-classes Cadre et NonCadre.

- **Nom de classe** : Employé.
 - Attributs :
 - numéro,
 - nom,
 - salaire de base.
 - Opérations : calculSalaire().
- **Nom de la sous-classe** : Cadre.
 - Attributs : niveau d'encadrement.
 - Opérations : calculSalaire().
- **Nom de la sous-classe** : NonCadre.
 - Attributs : niveau de spécialisation.
 - Opérations : calculSalaire().

Le principe de calcul du salaire étant de calculer pour chaque type d'employé une prime spécifique en fonction soit du niveau d'encadrement, soit du niveau de spécialisation selon le type d'employé.

Voyons maintenant comment se réalise l'application du polymorphisme lors de l'exécution des opérations.

Dans cet exemple, lorsque l'on appelle l'opération calculSalaire(), c'est l'opération de sous-classe Cadre ou celle de la sous-classe NonCadre qui est en fait activée selon l'objet concerné. L'opération de la sous-classe fait en général appel explicitement à l'opération calculSalaire() de la super-classe pour bénéficier des traitements communs aux cadres et non cadres et ensuite il y aura poursuite du traitement spécifique à la sous-classe.

1.1.6 Persistance

La **persistance** est la propriété donnée à un objet de continuer à exister après la fin de l'exécution du programme qui l'a créé.

Par défaut dans l'approche objet, aucun objet n'est persistant. Les modèles décrivent le système en exécution en mémoire centrale et ne tiennent pas compte *a priori* de l'état du système qui doit être stocké sur disque.

La gestion de la mémoire incombe au programmeur avec notamment le problème de la libération des espaces.

1.1.7 Avantages du développement à l'aide des langages objet

Par rapport à une approche fonctionnelle associée à un développement classique mené à l'aide de langages procéduraux, on est en droit de s'interroger sur les avantages qu'apporte réellement le développement à l'aide d'un langage objet comme par exemple C++, C# ou Java.

En fait, deux avantages prépondérants sont mis en général en avant lorsque l'on choisit une approche objet :

- **La modularité** – Par construction, étant donné que l'on conçoit des classes représentant une entité de taille limitée en données et en opérations, il est plus aisé de construire des systèmes modulables que si l'on élabore une seule base de données d'une part et un seul logiciel d'autre part.
- **La réutilisabilité** – La définition d'un système à l'aide de classe ayant chacune la responsabilité d'un sous-ensemble de données et des opérations associées favorise fortement la potentialité de trouver des classes réutilisables. La réutilisation de classe se réalise soit sur le plan métier à l'intérieur d'une même entreprise dans des applications différentes, soit sur le plan technique à l'échelle de tous les développements réalisés à l'aide d'un même langage. Sur ce dernier aspect, c'est toute l'approche du développement par composant qui est en jeu.

Au-delà de ces deux avantages majeurs et compte tenu de la plus grande modularité dans la construction d'une application à l'aide d'objets, la maintenance élémentaire de chaque classe est en soi plus simple à réaliser que celle d'un logiciel unique traitant toutes les données d'un système. Il importe bien entendu dans l'approche objet de construire son système en veillant à minimiser le nombre de relations entre classes.

1.2 PRÉSENTATION GÉNÉRALE D'UML

1.2.1 Historique

Regardons tout d'abord ce qui s'est passé au début des années 90. Par rapport à la cinquantaine de méthodes d'analyse et de conception objet qui existaient au début des années 90, seulement trois d'entre elles se sont détachées nettement au bout de quelques années. En effet, la volonté de converger vers une méthode unifiée était déjà bien réelle et c'est pour cette raison que les méthodes OMT, BOOCH et OOSE se sont démarquées des autres.

OMT (*Object Modeling Technique*) de James Rumbaugh et BOOCH de Grady Booch ont été les deux méthodes les plus diffusées en France durant les années 90. Par ailleurs, OOSE de Ivar Jacobson s'est aussi imposée dans le monde objet pour la partie formalisation des besoins.

Pour aller plus loin dans le rapprochement, James Rumbaugh et Grady Booch se sont retrouvés au sein de la société Rational Software et ont été ensuite rejoints par Ivar Jacobson en se donnant comme objectif de fusionner leur méthode et créer UML (*Unified Methode Language*).

Il est important de noter que contrairement à ce qui avait été envisagé au départ, le processus de développement a été sorti du champ couvert par le projet de norme.

UML est donc une norme du langage de modélisation objet qui a été publiée, dans sa première version, en novembre 1997 par l'OMG (*Object Management Group*), instance de normalisation internationale du domaine de l'objet.

En quelques années, UML s'est imposée comme standard à utiliser en tant que langage de modélisation objet.

Aujourd'hui, en cette fin de la première décennie des années 2000, nous avons déjà une dizaine d'années de recul sur l'enseignement et la pratique d'UML en entreprise.

Les grandes étapes de la diffusion d'UML peuvent se résumer comme suit :

1994-1996 : rapprochement des méthodes OMT, BOOCH et OOSE et naissance de la première version d'UML.

23 novembre 1997 : version 1.1 d'UML adoptée par l'OMG.

1998-1999 : sortie des versions 1.2 à 1.3 d'UML.

2000-2001 : sortie des dernières versions suivantes 1.x.

2002-2003 : préparation de la V2.

10 octobre 2004 : sortie de la V2.1.

5 février 2007 : sortie de la V2.1.1 (version de référence du présent ouvrage).

1.2.2 Structuration de la présentation

Nous proposons au lecteur une présentation détaillée d'UML 2 en privilégiant notamment dans les exemples et les études de cas le contexte d'utilisation des systèmes d'information. Le lecteur pourra, s'il le souhaite ensuite, approfondir sa connaissance d'UML en consultant directement la norme de référence disponible *via* internet [OMG2007].

La présentation d'UML réalisée dans le présent ouvrage se veut synthétique et pragmatique. Nous n'avons pas voulu couvrir tous les détails de la norme qui représente aujourd'hui un volume de près de 700 pages.

Nous avons, tout d'abord, pris le parti d'illustrer systématiquement les concepts présentés et la notation d'UML par des exemples concrets les plus proches possible du domaine de la gestion des entreprises. Ensuite nous donnons, pour les diagrammes les plus structurants d'UML des exercices d'ensemble corrigés et nous traitons aussi un exercice de synthèse (Locagite) qui nous sert de fil conducteur et d'illustration tout au long de l'ouvrage.

La présentation de la démarche que nous proposons UP7 s'appuie fortement sur le processus UP (*Unified Process*).

Les deux études de cas traités dans cet ouvrage sont l'occasion de voir comment les principaux concepts et la notation d'UML peuvent s'appliquer sur des domaines d'étude plus importants que ceux des exercices et sont l'occasion de concrétiser la démarche d'application d'UML que nous proposons.

Nous sommes convaincus que cette présentation pragmatique d'UML, associée aux deux études de cas, doit constituer une aide efficace à tous ceux qui veulent soit s'initier à UML soit approfondir leur maîtrise d'UML en particulier sur toutes les nouveautés apportées par UML 2.

1.2.3 Règles générales

Afin d'assurer un bon niveau de cohérence et d'homogénéité sur l'ensemble des modèles, UML propose d'une part un certain nombre de règles d'écriture ou de représentations graphiques normalisées et d'autre part des mécanismes ou des concepts communs applicables à l'ensemble des diagrammes. Certains éléments, comme les stéréotypes, sont spécifiquement prévus pour assurer une réelle capacité d'adaptation et d'évolution de la notation notamment pour prendre en compte les particularités des différentes situations à modéliser. Les principaux éléments généraux d'UML que nous présentons sont : le stéréotype, la valeur marquée, la note, la contrainte, et la relation de dépendance.

En outre UML propose un méta-modèle de tous les concepts et notations associées utilisés dans les treize diagrammes du langage de modélisation.

Méta-modèle

Le langage de modélisation UML respecte un certain nombre de règles sur les concepts manipulés (classes, attributs, opérations, paquetages...) ainsi que sur la syntaxe d'écriture et le formalisme de représentation graphique. L'ensemble de ces règles constitue en soi un langage de modélisation qui a fait l'objet d'un **méta-modèle** UML. L'intérêt de disposer d'un méta-modèle UML permet de bien maîtriser la structure d'UML et de faciliter son évolution.

Cette approche a été généralisée par l'OMG en normalisant la représentation des méta-modèles par la définition en 1997 d'un méta-méta-modèle défini dans le MOF (*Meta-Object Facility*). Le MOF représente ainsi un super langage de définition des méta-modèles.

Plus globalement, le MOF se retrouve au sommet d'une architecture de description à quatre niveaux :

- M3, niveau du MOF ;
- M2, niveau des méta-modèles (UML est un des méta-modèles) ;
- M1, constitué par les modèles (les diagrammes d'UML sont des instances de ce niveau) ;

- M0, constitué par les instances (les réalisations de diagrammes pour une situation donnée sont des instances de ce niveau).

Le méta-modèle d'UML est complètement décrit dans la norme.

Stéréotype

Un **stéréotype** constitue un moyen de classer les éléments de la modélisation. Un certain nombre de stéréotypes sont déjà définis dans UML (voir annexe C de la norme), mais d'autres valeurs de stéréotypes peuvent être ajoutées si cela est nécessaire soit à l'évolution générale d'UML, soit à la prise en compte de situations particulières propres aux entreprises.

Les stéréotypes peuvent s'appliquer à n'importe quel concept d'UML. Nous nous intéresserons dans cet ouvrage à un certain nombre d'entre eux que nous présenterons au niveau des diagrammes lorsque leur utilisation nous paraîtra pertinente.

En particulier, dans le diagramme de classe, le stéréotype permet de considérer de nouveaux types de classe. Cette possibilité d'extension pour les classes, se définit donc au niveau méta-classe.

Formalisme et exemple

Le nom du stéréotype est indiqué entre guillemets. Un acteur peut être vu comme un stéréotype particulier d'une classe appelée acteur. L'exemple (fig. 1.2) montre une classe Client stéréotypée comme « acteur ».

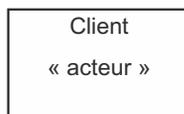


Figure 1.2 — Exemple d'une classe stéréotypée

Valeur marquée

UML permet d'indiquer des valeurs particulières au niveau des éléments de modélisation et en particulier pour les attributs de classe. Une **valeur marquée** se définit au niveau méta-attribut.

Formalisme et exemple

La valeur marquée est mise entre accolades avec indication du nom et de la valeur : {persistance : string} si l'on veut ajouter ce type d'attribut dans une classe.

Profil

Afin de donner la possibilité de spécialiser chaque application d'UML à son propre contexte, UML propose de définir un **profil** d'utilisation caractérisé principalement par la liste des stéréotypes, la liste des valeurs marquées et les contraintes spécifiées pour un projet donné.

Note

Une **note** correspond à un commentaire explicatif d'un élément d'UML.

Formalisme et exemple

La figure 1.3 montre le formalisme et un exemple de la représentation d'une note.

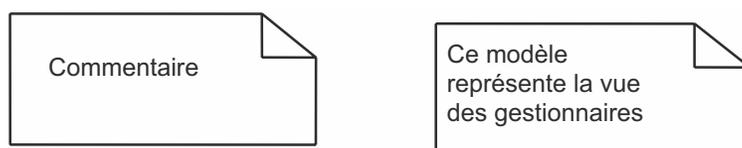


Figure 1.3 — Formalisme et exemple d'utilisation d'une note

Contrainte

Une **contrainte** est une note ayant une valeur sémantique particulière pour un élément de la modélisation. Une contrainte s'écrit entre accolades {}. Dans le cas où la contrainte concerne deux classes ou plus, celle-ci s'inscrit à l'intérieur d'une note.

Formalisme et exemple

- Première forme d'écriture d'une contrainte : {ceci est une contrainte}.
- Deuxième forme d'écriture : à l'intérieur d'une note (fig. 1.4).

Dans UML, un langage spécifique d'expression de contraintes est disponible ; c'est le langage OCL (*Object Constraint Language*). Ce langage n'est pas décrit dans le présent ouvrage.

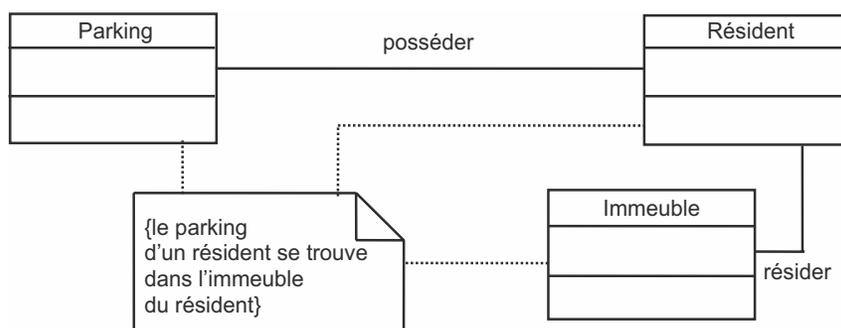


Figure 1.4 — Exemple d'utilisation d'une contrainte (sans représentation des multiplicités)

1.2.4 Présentation générale des diagrammes

UML dans sa version 2 propose treize diagrammes qui peuvent être utilisés dans la description d'un système. Ces diagrammes sont regroupés dans deux grands ensembles.

- **Les diagrammes structurels** – Ces diagrammes, au nombre de six, ont vocation à représenter l'aspect statique d'un système (classes, objets, composants...)
 - *Diagramme de classe* – Ce diagramme représente la description statique du système en intégrant dans chaque classe la partie dédiée aux données et celle consacrée aux traitements. C'est le diagramme pivot de l'ensemble de la modélisation d'un système.
 - *Diagramme d'objet* – Le diagramme d'objet permet la représentation d'instances des classes et des liens entre instances.
 - *Diagramme de composant (modifié dans UML 2)* – Ce diagramme représente les différents constituants du logiciel au niveau de l'implémentation d'un système.
 - *Diagramme de déploiement (modifié dans UML 2)* – Ce diagramme décrit l'architecture technique d'un système avec une vue centrée sur la répartition des composants dans la configuration d'exploitation.
 - *Diagramme de paquetage (nouveau dans UML 2)* – Ce diagramme donne une vue d'ensemble du système structuré en paquetage. Chaque paquetage représente un ensemble homogène d'éléments du système (classes, composants...).
 - *Diagramme de structure composite (nouveau dans UML 2)* – Ce diagramme permet de décrire la structure interne d'un ensemble complexe composé par exemple de classes ou d'objets et de composants techniques. Ce diagramme met aussi l'accent sur les liens entre les sous-ensembles qui collaborent.
- **Les diagrammes de comportement** – Ces diagrammes représentent la partie dynamique d'un système réagissant aux événements et permettant de produire les résultats attendus par les utilisateurs. Sept diagrammes sont proposés par UML :
 - *Diagramme des cas d'utilisation* – Ce diagramme est destiné à représenter les besoins des utilisateurs par rapport au système. Il constitue un des diagrammes les plus structurants dans l'analyse d'un système.
 - *Diagramme d'état-transition (machine d'état)* – Ce diagramme montre les différents états des objets en réaction aux événements.
 - *Diagramme d'activités (modifié dans UML 2)* – Ce diagramme donne une vision des enchaînements des activités propres à une opération ou à un cas d'utilisation. Il permet aussi de représenter les flots de contrôle et les flots de données.
 - *Diagramme de séquence (modifié dans UML 2)* – Ce diagramme permet de décrire les scénarios de chaque cas d'utilisation en mettant l'accent sur la chronologie des opérations en interaction avec les objets.

- *Diagramme de communication (anciennement appelé collaboration)* – Ce diagramme est une autre représentation des scénarios des cas d'utilisation qui met plus l'accent sur les objets et les messages échangés.
- *Diagramme global d'interaction (nouveau dans UML 2)* – Ce diagramme fournit une vue générale des interactions décrites dans le diagramme de séquence et des flots de contrôle décrits dans le diagramme d'activités.
- *Diagramme de temps (nouveau dans UML 2)* – Ce diagramme permet de représenter les états et les interactions d'objets dans un contexte où le temps a une forte influence sur le comportement du système à gérer.

Aujourd'hui UML 2 décrit les concepts et le formalisme de ces treize diagrammes mais ne propose pas de démarche de construction couvrant l'analyse et la conception d'un système. Ce qui a pour conséquence par exemple de ne pas disposer d'une vision des interactions entre les diagrammes.

Formalisme et exemple

Afin de donner un premier aperçu des principaux diagrammes tant sur l'aspect du formalisme que sur leur usage, nous proposons à titre introductif un petit exemple très simple.

Considérons une nouvelle société de formation qui souhaite développer un premier niveau de site web dans lequel elle présente succinctement les formations proposées et enregistre en ligne les demandes de catalogue.

Nous pouvons dès ce stade de l'analyse représenter le diagramme des cas d'utilisation (fig. 1.5).

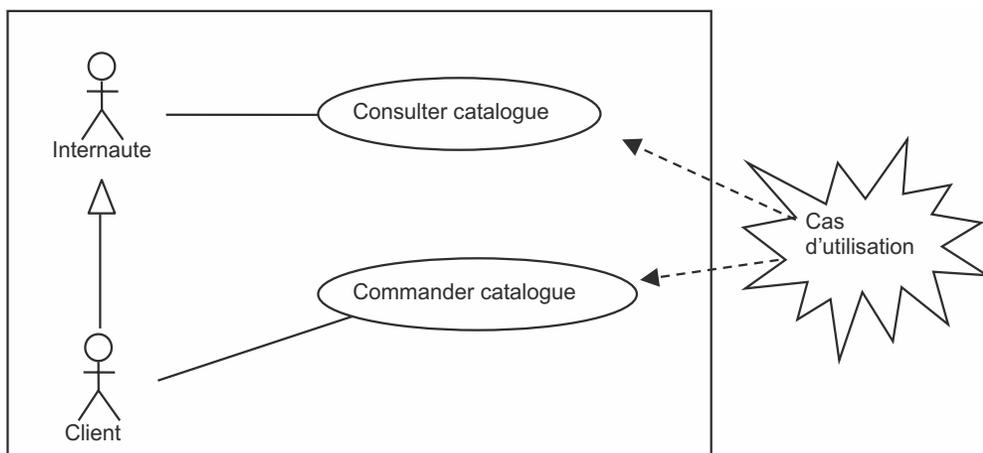


Figure 1.5 – Exemple de diagramme des cas d'utilisation

Le diagramme de classe (fig. 1.6) va nous permettre de décrire les concepts manipulés, à savoir : Client, Catalogue et Formation.

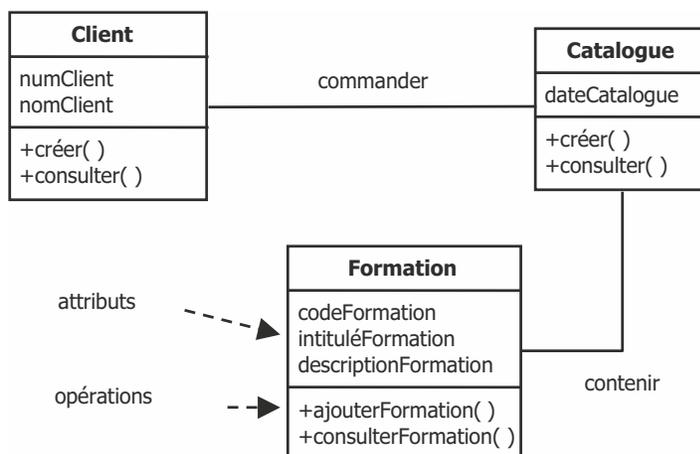


Figure 1.6 — Exemple de diagramme de classe

Le diagramme de séquence va nous permettre de décrire les scénarios des cas d'utilisation du diagramme des cas d'utilisation. À titre d'exemple nous montrons (fig. 1.7) le scénario correspondant à la consultation du catalogue.

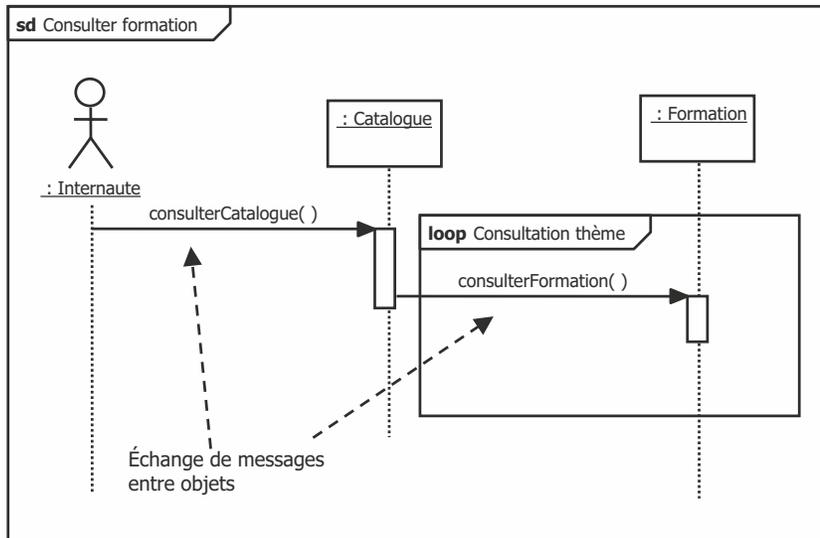


Figure 1.7 — Exemple de diagramme de classe

Cette première illustration de trois diagrammes donne déjà un éclairage sur les concepts importants que sont la classe, le cas d'utilisation et l'objet.

1.2.5 Schéma d'ensemble des treize diagrammes d'UML 2

Afin de donner quelques points de repères sur le positionnement et les liens entre tous les diagrammes d'UML, nous donnons ici notre propre vision en proposant un regroupement des diagrammes en quatre ensembles suivant leur finalité :

- description du système : huit diagrammes ;
- architecture technique : deux diagrammes ;
- vues globales ou spécialisées : deux diagrammes ;
- partition d'éléments de la modélisation : un diagramme.

Le schéma proposé reprend les treize diagrammes en les répartissant sur les quatre ensembles définis (fig. 1.8).

Nous avons adopté, dans cet ouvrage, les abréviations suivantes pour les treize diagrammes :

DAC : Diagramme d'activité
DCL : Diagramme de classe
DOB : Diagramme d'objet
DCP : Diagramme de composant
DCU : Diagramme des cas d'utilisation
DCO : Diagramme de communication
DET : Diagramme d'état-transition
DGI : Diagramme global d'interaction
DPA : Diagramme de paquetage
DPL : Diagramme de déploiement
DSC : Diagramme de structure composite
DSE : Diagramme de séquence
DTP : Diagramme de temps

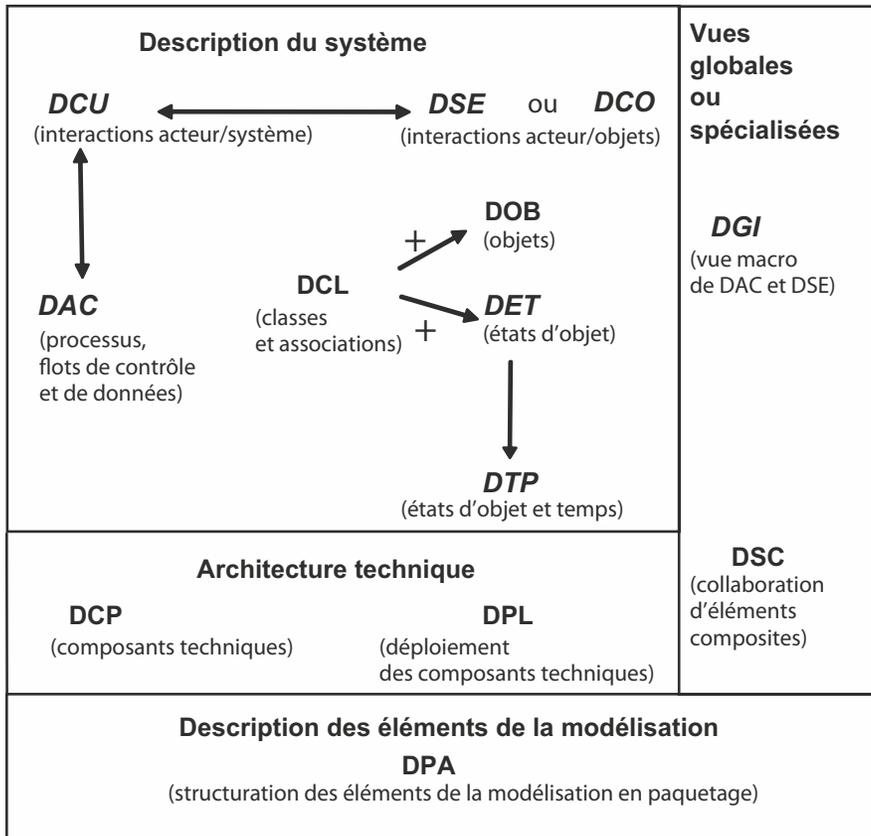


Figure 1.8 — Schéma d'ensemble des treize diagrammes d'UML 2
Les noms en italiques représentent les diagrammes de comportement

2

Les diagrammes structurels (ou statiques)

2.1 DIAGRAMME DE CLASSE (DCL) ET DIAGRAMME D'OBJET (DOB)

Le diagramme de classe constitue l'un des pivots essentiels de la modélisation avec UML. En effet, ce diagramme permet de donner la représentation statique du système à développer. Cette représentation est centrée sur les concepts de classe et d'association. Chaque classe se décrit par les données et les traitements dont elle est responsable pour elle-même et vis-à-vis des autres classes. Les traitements sont matérialisés par des opérations. Le détail des traitements n'est pas représenté directement dans le diagramme de classe ; seul l'algorithme général et le pseudo-code correspondant peuvent être associés à la modélisation.

La description du diagramme de classe est fondée sur :

- le concept d'objet,
- le concept de classe comprenant les attributs et les opérations,
- les différents types d'association entre classes.

2.1.1 Objet

Nous allons donner une première définition du concept d'objet avant de traiter le concept de classe. La description d'un objet sera complétée simultanément à la présentation du concept de classe.

Un **objet** est un concept, une abstraction ou une chose qui a un sens dans le contexte du système à modéliser. Chaque objet a une identité et peut être distingué des autres sans considérer a priori les valeurs de ses propriétés.

Exemple

La figure 2.1 montre des exemples d'objets physiques (une chaise, une voiture, une personne, un vélo) et d'objets de gestion (la Commande n° 12, le Client Durand).

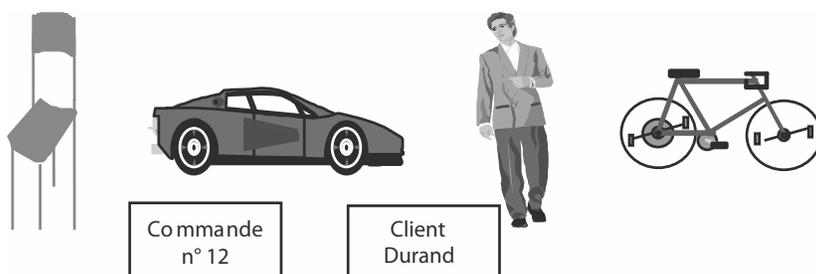


Figure 2.1 — Exemples d'objets physiques et d'objets de gestion

Autres caractéristiques

Un objet est caractérisé par les valeurs de ses propriétés qui lui confèrent des états significatifs suivant les instants considérés. Le formalisme de représentation d'un objet est donné après celui d'une classe.

2.1.2 Classe, attribut et opération

Classe

Une **classe** décrit un groupe d'objets ayant les mêmes propriétés (attributs), un même comportement (opérations), et une sémantique commune (domaine de définition).

Un objet est une **instance** d'une classe. La classe représente l'abstraction de ses objets. Au niveau de l'implémentation, c'est-à-dire au cours de l'exécution d'un programme, l'identificateur d'un objet correspond une adresse mémoire.

Formalisme général et exemple

Une classe se représente à l'aide d'un rectangle comportant plusieurs compartiments. Les trois compartiments de base sont :

- la désignation de la classe,
- la description des attributs,
- la description des opérations.

Deux autres compartiments peuvent être aussi indiqués :

- la description des responsabilités de la classe,
- la description des exceptions traitées par la classe.

Il est possible de manipuler les classes en limitant le niveau de description à un nombre réduit de compartiments selon les objectifs poursuivis par le modélisateur. Ainsi les situations suivantes sont possibles pour la manipulation d'une description restreinte de classe :

- description uniquement du nom et des caractéristiques générales de la classe,
- description du nom de la classe et de la liste d'attributs.

La figure 2.2 montre le formalisme général des compartiments d'une classe et des premiers exemples.

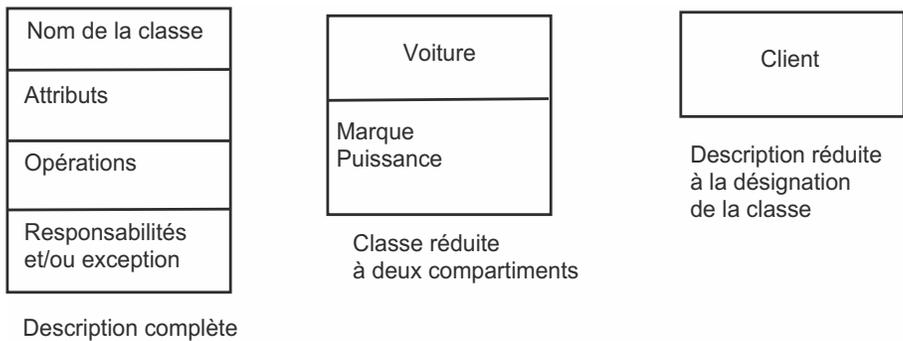


Figure 2.2 — Formalisme général d'une classe et exemples

Attribut

Un **attribut** est une propriété élémentaire d'une classe. Pour chaque objet d'une classe, l'attribut prend une valeur (sauf cas d'attributs multivalués).

Formalisme et exemple

La figure 2.3 montre le formalisme et un exemple de représentation des attributs de classe.

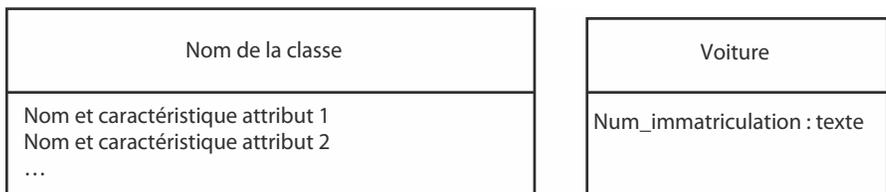


Figure 2.3 — Formalisme d'attributs de classe et exemple

Caractéristiques

Le nom de la classe peut être qualifié par un « stéréotype ». La description complète des attributs d'une classe comporte un certain nombre de caractéristiques qui doivent respecter le formalisme suivant :

- **Visibilité/Nom attribut : type [= valeur initiale {propriétés}]**
 - *Visibilité* : se reporter aux explications données plus loin sur ce point.
 - *Nom d'attribut* : nom unique dans sa classe.
 - *Type* : type primitif (entier, chaîne de caractères...) dépendant des types disponibles dans le langage d'implémentation ou type classe matérialisant un lien avec une autre classe.
 - *Valeur initiale* : valeur facultative donnée à l'initialisation d'un objet de la classe.
 - *{propriétés}* : valeurs marquées facultatives (ex. : « interdit » pour mise à jour interdite).

Un attribut peut avoir des valeurs multiples. Dans ce cas, cette caractéristique est indiquée après le nom de l'attribut (ex. : prénom [3] pour une personne qui peut avoir trois prénoms).

Un attribut dont la valeur peut être calculée à partir d'autres attributs de la classe est un attribut dérivé qui se note « /nom de l'attribut dérivé ». Un exemple d'attribut dérivé est donné à la figure 2.5.

Opération

Une **opération** est une fonction applicable aux objets d'une classe. Une opération permet de décrire le comportement d'un objet. Une **méthode** est l'implémentation d'une opération.

Formalisme et exemple

Chaque opération est désignée soit seulement par son nom soit par son nom, sa liste de paramètres et son type de résultat. La signature d'une méthode correspond au nom de la méthode et la liste des paramètres en entrée. La figure 2.4 montre le formalisme et un exemple de représentation d'opérations de classe.

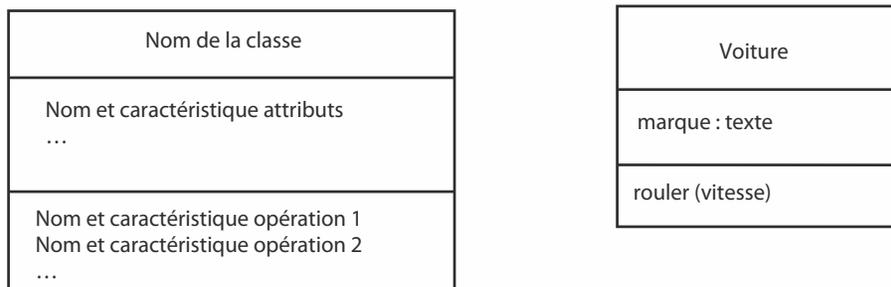


Figure 2.4 — Formalisme et exemple d'opérations de classe

Caractéristiques

La description complète des opérations d'une classe comporte un certain nombre de caractéristiques qui doivent respecter le formalisme suivant :

- **Visibilité Nom d'opération (paramètres) [:[type résultat] {propriétés}]**
 - *Visibilité* : se reporter aux explications données plus loin sur ce point.
 - *Nom d'opération* : utiliser un verbe représentant l'action à réaliser.
 - *Paramètres* : liste de paramètres (chaque paramètre peut être décrit, en plus de son nom, par son type et sa valeur par défaut). L'absence de paramètre est indiquée par ().
 - *Type résultat* : type de (s) valeur(s) retourné(s) dépendant des types disponibles dans le langage d'implémentation. Par défaut, une opération ne retourne pas de valeur, ceci est indiqué par exemple par le mot réservé « void » dans le langage C++ ou Java.
 - *{propriétés}* : valeurs facultatives applicables (ex. : {query} pour un comportement sans influence sur l'état du système).

Exemples de classes et représentation d'objets

La figure 2.5 présente l'exemple d'une classe « Voiture ». La figure 2.6 donne le formalisme d'un objet.

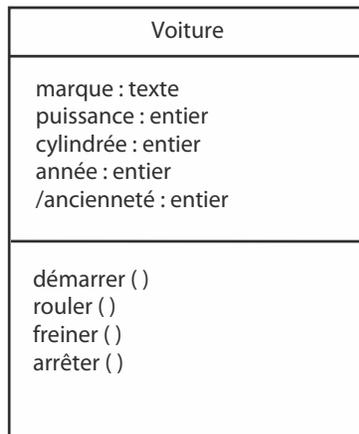


Figure 2.5 — Exemple de représentation d'une classe

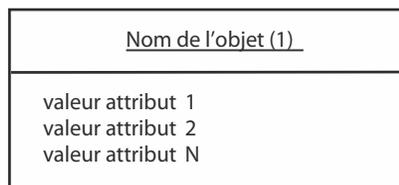


Figure 2.6 — Formalisme de représentation d'un objet

(1) Le nom d'un objet peut être désigné sous trois formes : *nom de l'objet*, désignation directe et explicite d'un objet ; *nom de l'objet : nom de la classe*, désignation incluant le nom de la classe ; *: nom de la classe*, désignation anonyme d'un objet d'une classe donnée.

Il est utile de préciser que la représentation des objets sera utilisée dans plusieurs autres diagrammes importants d'UML. C'est le cas notamment du diagramme de séquence ou encore du diagramme d'état-transition.

La figure 2.7 présente des exemples d'objets.

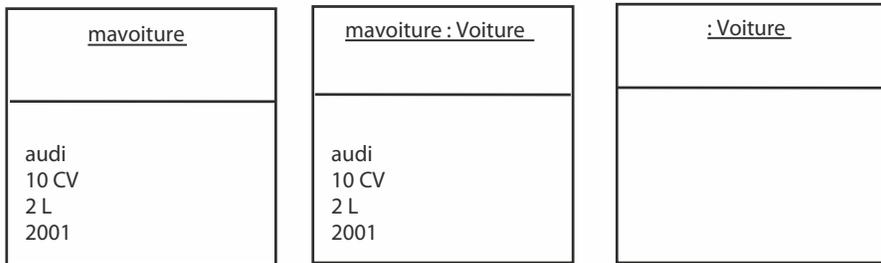


Figure 2.7 – Exemples de représentation d'objets

Visibilité des attributs et opérations

Chaque attribut ou opération d'une classe peut être de type public, protégé, privé ou paquetage. Les symboles + (public), # (protégé), - (privé) et ~ (paquetage) sont indiqués devant chaque attribut ou opération pour signifier le type de visibilité autorisé pour les autres classes.

Les droits associés à chaque niveau de confidentialité sont :

- **Public (+)** – Attribut ou opération visible par tous.
- **Protégé (#)** – Attribut ou opération visible seulement à l'intérieur de la classe et pour toutes les sous-classes de la classe.
- **Privé (-)** – Attribut ou opération seulement visible à l'intérieur de la classe.
- **Paquetage (~)** – Attribut ou opération ou classe seulement visible à l'intérieur du paquetage où se trouve la classe.

Exemple

La figure 2.8 montre un exemple d'utilisation des symboles de la visibilité des éléments d'une classe.

Dans cet exemple, tous les attributs sont déclarés de type privé, les opérations « démarrer » et « freiner » sont de type public, l'opération « rouler » est de type privé et l'opération « arrêter » est de type protégé.

Attribut ou opération de niveau classe

Caractéristiques

Un **attribut** ou une **opération** peut être défini non pas au niveau des instances d'une classe, mais au niveau de la classe. Il s'agit soit d'un attribut qui est une constante pour toutes les instances d'une classe soit d'une opération d'une classe abstraite (voir § 2.1.6) ou soit par exemple d'une opération « créer » qui peut être définie au niveau de la classe et applicable à la classe elle-même.

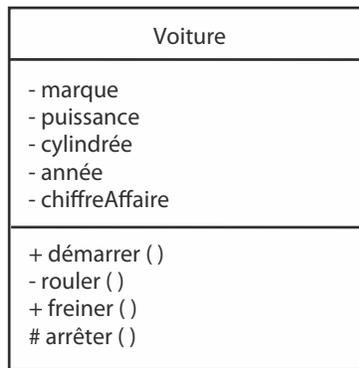


Figure 2.8 — Exemple de représentation des symboles de visibilité

Formalisme et exemple

C'est le soulignement de l'attribut ou de l'opération qui caractérise cette propriété. Dans l'exemple de la figure 2.9, l'attribut « ristourne » est de type classe et l'opération « créer » est une opération exécutable au niveau de la classe.

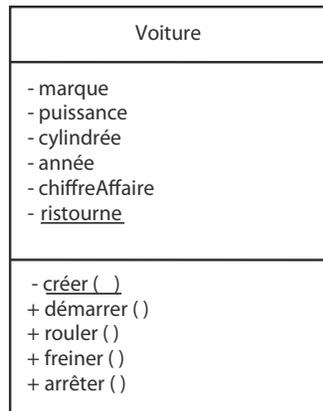


Figure 2.9 — Exemple d'attribut ou d'opération de niveau classe

2.1.3 Association, multiplicité, navigabilité et contraintes

Lien et association

Un **lien** est une connexion physique ou conceptuelle entre instances de classes donc entre objets. Une **association** décrit un groupe de liens ayant une même structure et une même sémantique. Un lien est une instance d'une association. Chaque association peut être identifiée par son nom.

Une **association entre classes** représente les liens qui existent entre les instances de ces classes.

Formalisme et exemple

La figure 2.10 donne le formalisme de l'association. Le symbole (facultatif) indique le sens de lecture de l'association. Dans cette figure est donné aussi un exemple de représentation d'une association.

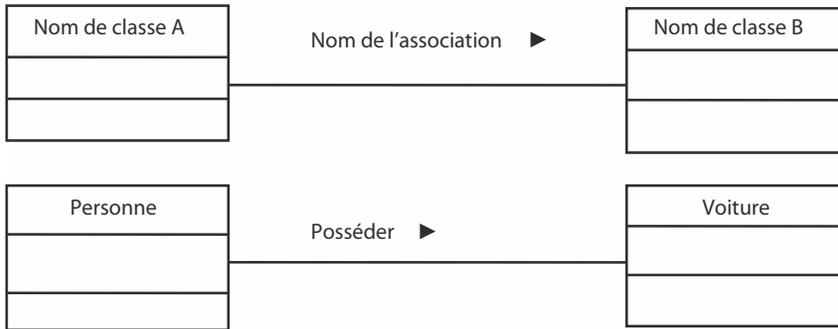


Figure 2.10 – Formalisme et exemple d'association

Rôle d'association

Le rôle tenu par une classe vis-à-vis d'une association peut être précisé sur l'association.

Exemple

La figure 2.11 donne un exemple de rôle d'association.



Figure 2.11 – Exemple de rôles d'une association

Multiplicité

La **multiplicité** indique un domaine de valeurs pour préciser le nombre d'instance d'une classe vis-à-vis d'une autre classe pour une association donnée. La multiplicité peut aussi être utilisée pour d'autres usages comme par exemple un attribut multi-valué. Le domaine de valeurs est décrit selon plusieurs formes :

- **Intervalle fermé** – Exemple : 2, 3 ..15.
- **Valeurs exactes** – Exemple : 3, 5, 8.

- **Valeur indéterminée notée *** – Exemple : 1..*.
 – Dans le cas où l'on utilise seulement *, cela traduit une multiplicité 0..*.
 – Dans le cas de multiplicité d'associations, il faut indiquer les valeurs minimale et maximale d'instances d'une classe vis-à-vis d'une instance d'une autre classe.

Formalisme et exemple

Nous donnons, à la figure 2.12, quelques exemples des principales multiplicités définies dans UML.

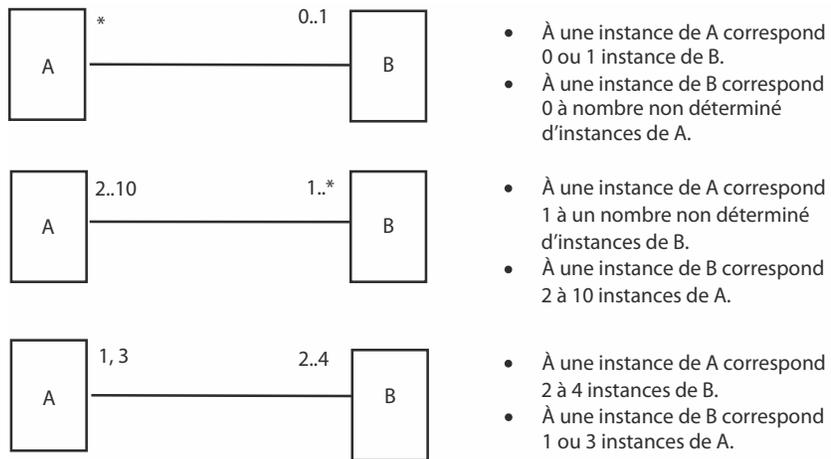


Figure 2.12 – Exemple de multiplicités

Navigabilité

La **navigabilité** indique si l'association fonctionne de manière unidirectionnelle ou bidirectionnelle, elle est matérialisée par une ou deux extrémités fléchées. La non-navigabilité se représente par un « X »

Les situations possibles de navigabilité sont représentées à la figure 2.13.

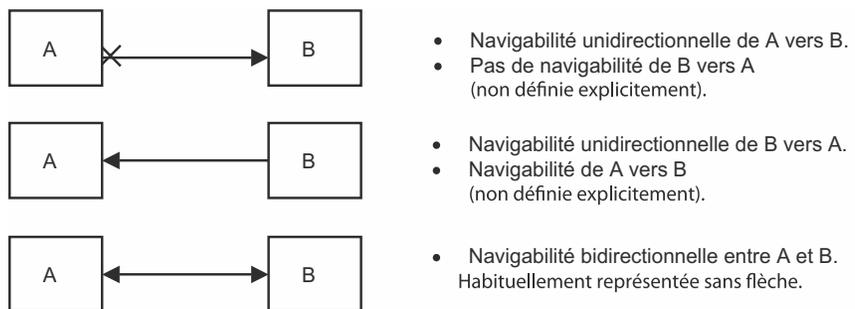


Figure 2.13 – Représentation de la navigabilité d'association

Par défaut, on admet qu'une navigabilité non définie correspond à une navigabilité implicite.

Dans l'exemple donné à la figure 2.14, à une personne sont associées ses copies d'examen mais l'inverse n'est pas possible (retrouver directement l'auteur de la copie d'examen, notamment avant la correction de la copie).

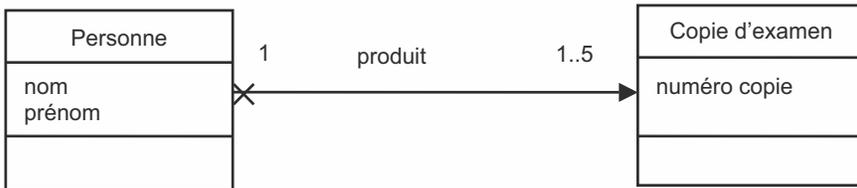


Figure 2.14 – Exemple de navigabilité d'une association

Contraintes

D'autres propriétés particulières (contraintes) sont proposées dans UML pour préciser la sémantique d'une association.

Ordre de tri

Pour une association de multiplicité supérieure à 1, les liens peuvent être :

- non ordonnés (valeur par défaut),
- ordonnés ou triés lorsque l'on est au niveau de l'implémentation (tri sur une valeur interne).

Un exemple est donné à la figure 2.15. Dans cet exemple, pour une entreprise donnée, les personnes seront enregistrées suivant un ordre qui correspondra à un des attributs de Personne.

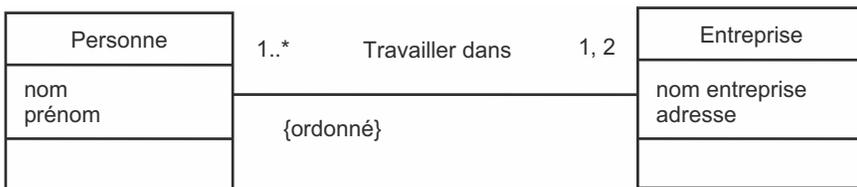


Figure 2.15 – Exemple de contrainte d'ordre d'une association

Propriétés de mise à jour de liens

Il est possible d'indiquer des contraintes particulières relatives aux conditions de mise à jour des liens.

- {interdit} : interdit l'ajout, la suppression ou la mise à jour des liens.
- {ajout seul} : n'autorise que l'ajout de liens.

Association de dimension supérieure à 2 et classe-association

Une association de dimension supérieure à 2 se représente en utilisant un losange permettant de relier toutes les classes concernées.

Une **classe-association** permet de décrire soit des attributs soit des opérations propres à l'association. Cette classe-association est elle-même reliée par un trait en pointillé au losange de connexion. Une classe-association peut être reliée à d'autres classes d'un diagramme de classes.

Exemple

Un exemple d'une association de dimension 3 comprenant une classe-association « Affectation » est donné à la figure 2.16. La classe-association Affectation permet de décrire les attributs propres à l'association de dimension 3 représentée.

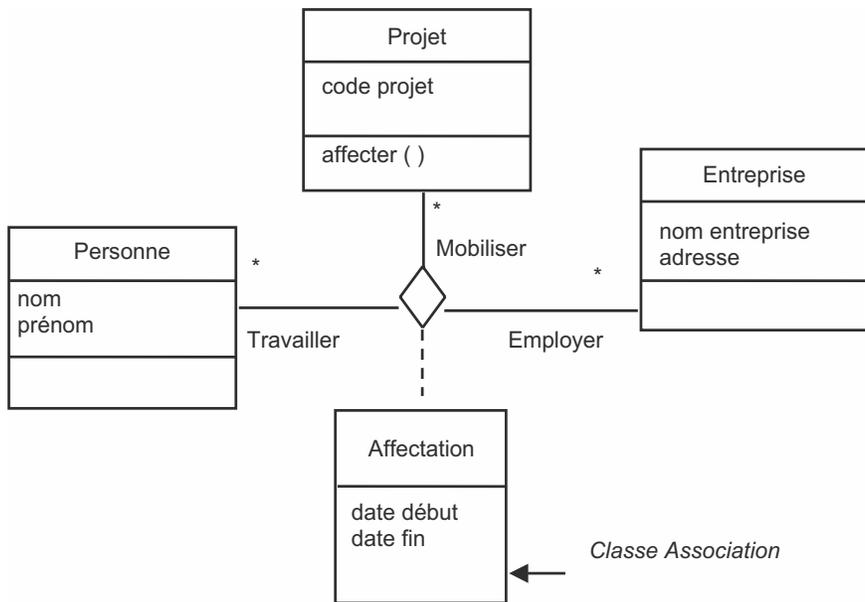


Figure 2.16 — Exemple d'une association de dimension 3 et d'une classe-association

2.1.4 Agrégation et composition entre classes

Agrégation

L'**agrégation** est une association qui permet de représenter un lien de type « ensemble » comprenant des « éléments ». Il s'agit d'une relation entre une classe représentant le niveau « ensemble » et 1 à n classes de niveau « éléments ». L'agrégation représente un lien structurel entre une classe et une ou plusieurs autres classes.

Formalisme et exemple

La figure 2.17 donne le formalisme général de l'agrégation.

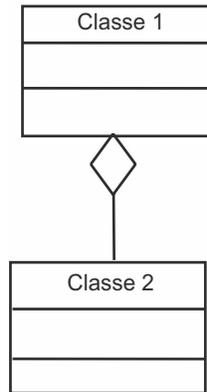


Figure 2.17 – Formalisme de l'agrégation

La figure 2.18 montre un exemple de relation d'agrégation. Dans cet exemple, nous avons modélisé le fait qu'un ordinateur comprend une UC, un clavier et un écran.

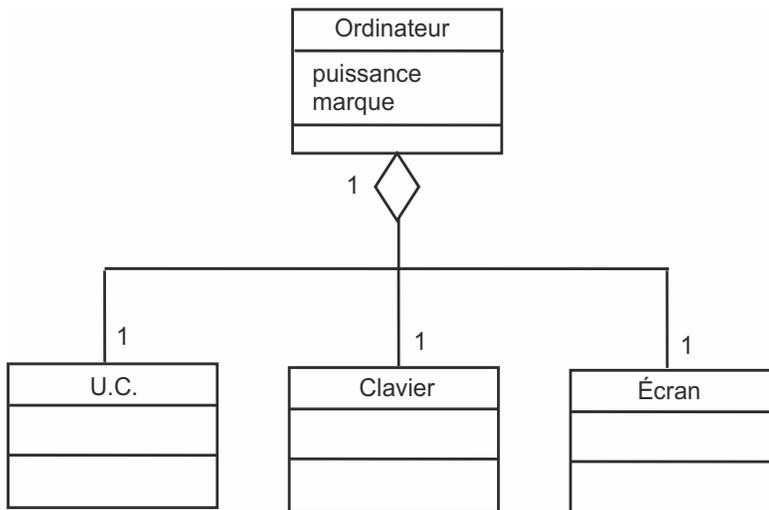


Figure 2.18 – Exemple d'agrégation

Composition

La **composition** est une relation d'agrégation dans laquelle il existe une contrainte de durée de vie entre la classe « composant » et la ou les classes « composé ». Autrement dit la suppression de la classe « composé » implique la suppression de la ou des classes « composant ».

Formalisme et exemple

La figure 2.19 donne le formalisme général de la composition. La figure 2.20 montre un exemple de relation de composition. Une seconde forme de présentation peut être aussi utilisée, elle est illustrée à la figure 2.21.

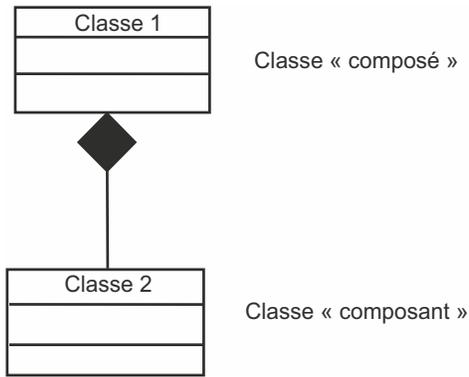


Figure 2.19 – Formalisme de la composition

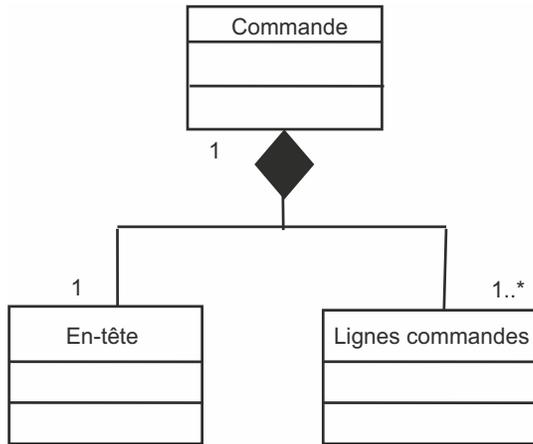


Figure 2.20 – Exemple d'une relation de composition

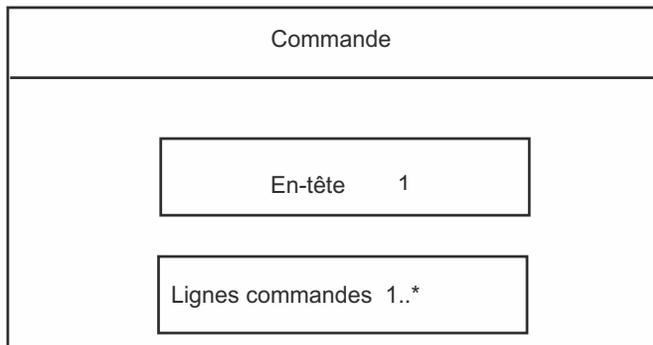


Figure 2.21 – Exemple de la seconde forme de représentation de la relation de composition

2.1.5 Association qualifiée, dépendance et classe d'interface

Qualification

La **qualification** d'une relation entre deux classes permet de préciser la sémantique de l'association et de qualifier de manière restrictive les liens entre les instances.

Seules les instances possédant l'attribut indiqué dans la qualification sont concernées par l'association. Cet attribut ne fait pas partie de l'association.

Formalisme et exemple

Soit la relation entre les répertoires et les fichiers appartenant à ces répertoires. À un répertoire est associé 0 à n fichiers. Si l'on veut restreindre cette association pour ne considérer qu'un fichier associé à son répertoire, la relation qualifiée est alors utilisée pour cela. La figure 2.22 montre la représentation de ces deux situations.

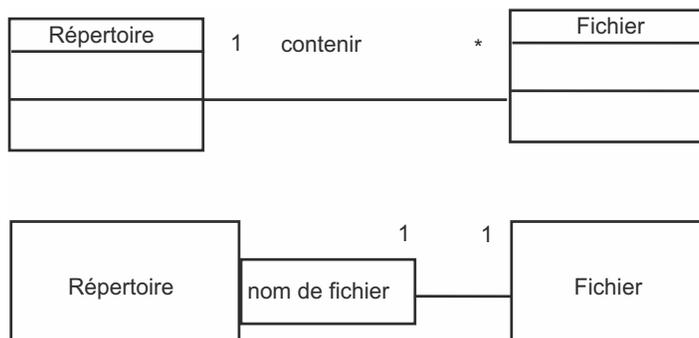


Figure 2.22 — Formalisme et exemple d'association qualifiée

Dépendance

La **dépendance** entre deux classes permet de représenter l'existence d'un lien sémantique. Une classe B est en dépendance de la classe A si des éléments de la classe A sont nécessaires pour construire la classe B.

Formalisme et exemple

La relation de dépendance se représente par une flèche en pointillé (fig. 2.23) entre deux classes.



Figure 2.23 — Formalisme de représentation d'un lien de dépendance

Interface

Une classe d'**interface** permet de décrire la vue externe d'une classe. La classe d'interface, identifiée par un nom, comporte la liste des opérations accessibles par les autres classes. Le compartiment des attributs ne fait pas partie de la description d'une interface.

L'interface peut être aussi matérialisée plus globalement par un petit cercle associé à la classe source.

La classe utilisatrice de l'interface est reliée au symbole de l'interface par une flèche en pointillé. La classe d'interface est une spécification et non une classe réelle. Une classe d'interface peut s'assimiler à une classe abstraite.

Formalisme et exemple

La figure 2.24 donne le formalisme, sur un exemple, des deux types de représentation d'une interface.

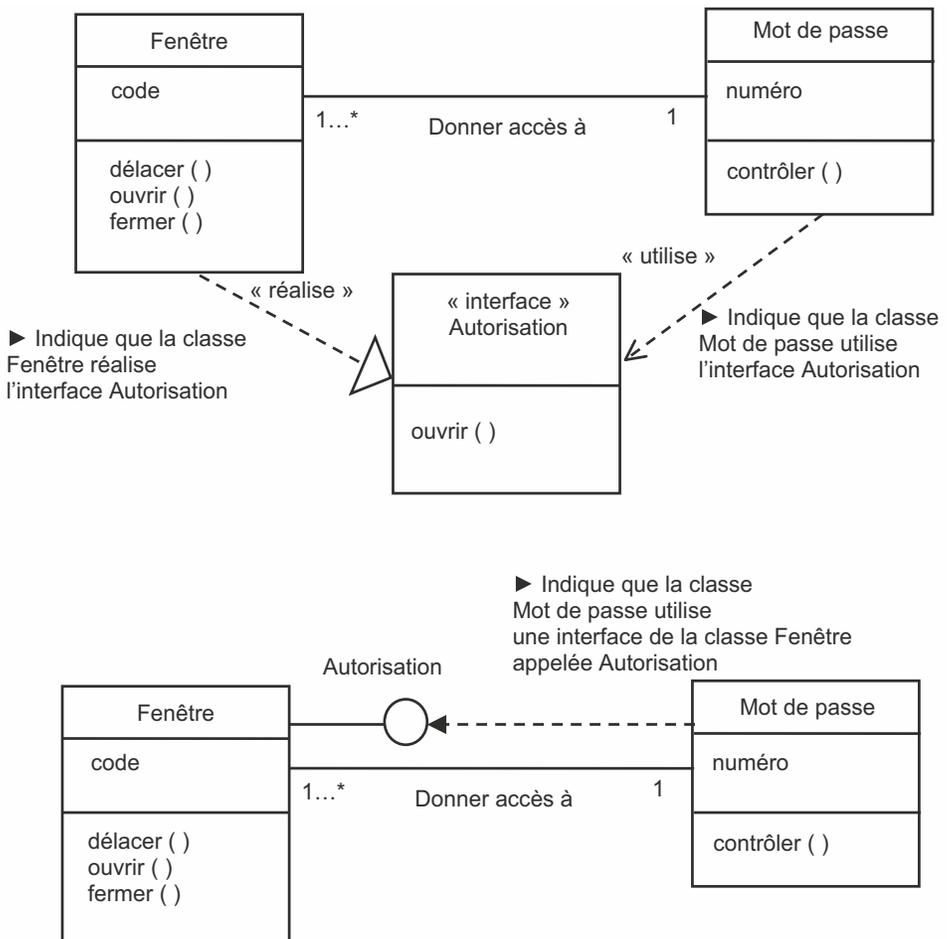


Figure 2.24 — Exemple de description d'une classe d'interface

2.1.6 Généralisation et spécialisation

La généralisation/spécialisation et l'héritage simple

La **généralisation** est la relation entre une classe et deux autres classes ou plus partageant un sous-ensemble commun d'attributs et/ou d'opérations.

La classe qui est affinée s'appelle **super-classe**, les classes affinées s'appellent **sous-classes**. L'opération qui consiste à créer une super-classe à partir de classes s'appelle la généralisation. Inversement la **spécialisation** consiste à créer des sous-classes à partir d'une classe.

Formalisme et exemple

La figure 2.25 montre le formalisme de la généralisation-spécialisation sous forme d'exemple général. Dans cet exemple :

- la sous-classe A1 hérite de A, c'est une spécialisation de A ;
- la sous-classe A2 hérite de A, c'est une spécialisation de A.

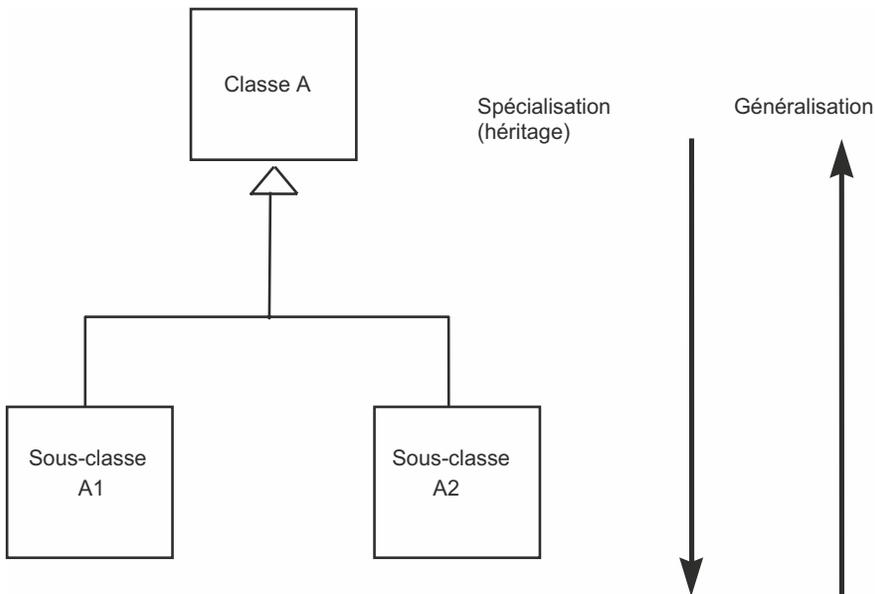


Figure 2.25 — Formalisme de la relation de généralisation

L'héritage permet à une sous-classe de disposer des attributs et opérations de la classe dont elle dépend. Un discriminant peut être utilisé pour exploiter le critère de spécialisation entre une classe et ses sous-classes. Le discriminant est simplement indiqué sur le schéma, puisque les valeurs prises par ce discriminant correspondent à chaque sous-classe.

La figure 2.26 montre un exemple de relation de spécialisation. Dans cet exemple, les attributs nom, prénom et date de naissance et l'opération « calculer âge » de « Employé » sont hérités par les trois sous-classes : Employé horaire, Employé salarié, Vacataire.

Classe abstraite

Une **classe abstraite** est une classe qui n'a pas d'instance directe mais dont les classes descendantes ont des instances. Dans une relation d'héritage, la super-classe est par définition une classe abstraite. C'est le cas de la classe Employé dans l'exemple présenté à la figure 2.26.

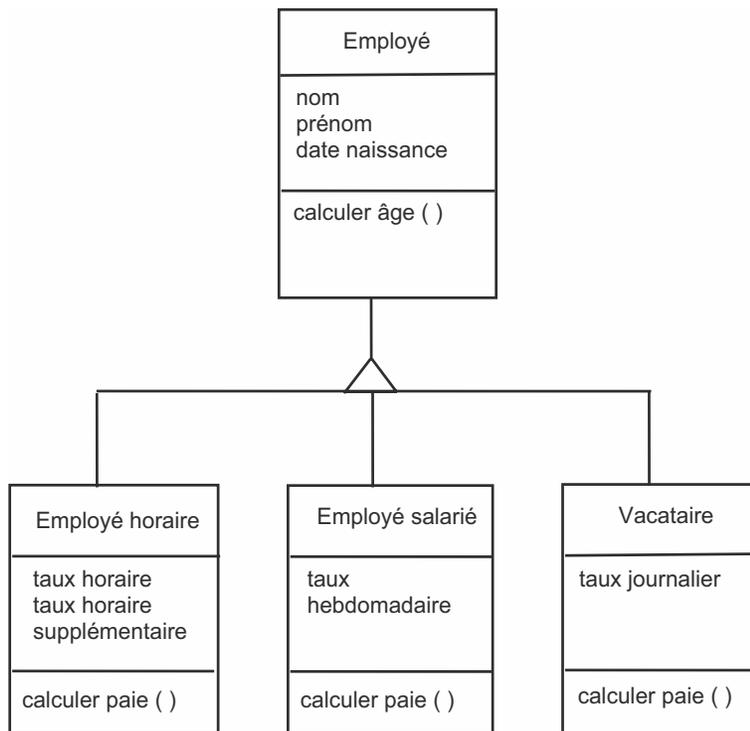


Figure 2.26 — Exemple de relation de spécialisation

L'héritage avec recouvrement

Par défaut, les sous-classes ont des instances disjointes les unes par rapport aux autres. Dans certains cas, il existe un recouvrement d'instances entre les sous-classes. D'une manière générale, quatre situations peuvent se rencontrer et se représentent sous forme de contraintes :

- **{chevauchement}** : deux sous-classes peuvent avoir, parmi leurs instances, des instances identiques ;
- **{disjoint}** : les instances d'une sous-classe ne peuvent être incluses dans une autre sous-classe de la même classe ;

- **{complète}** : la généralisation ne peut pas être étendue ;
- **{incomplète}** : la généralisation peut être étendue.

Dans certains cas, il est possible de ne pas citer toutes les sous-classes mais d'indiquer seulement des points de suspension (...).

Formalisme et exemple

La figure 2.27 montre un exemple d'héritage avec recouvrement d'instances entre les classes Étudiant et Employé. En effet, une même personne peut être à la fois étudiante dans une université et employée dans une entreprise.

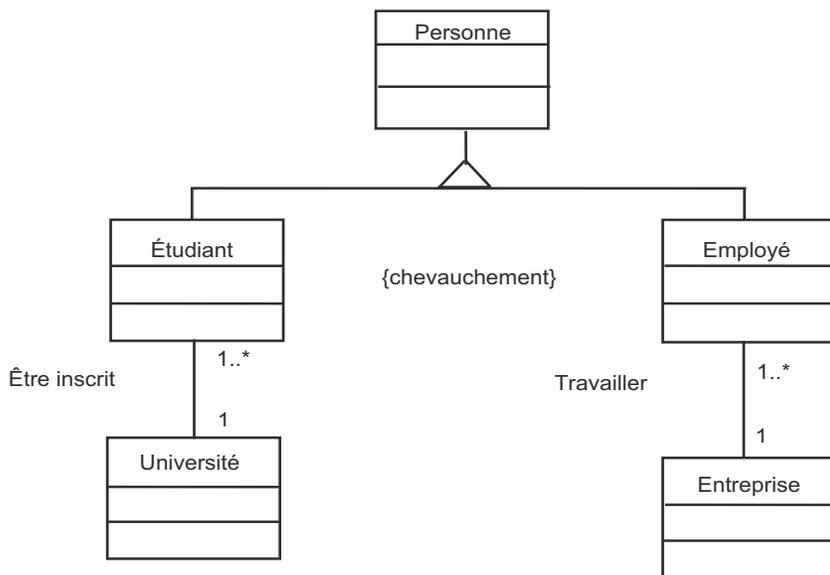


Figure 2.27 — Exemple d'héritage avec recouvrement d'instances

Extension et restriction de classe

L'ajout de propriétés dans une sous-classe correspond à une **extension de classe**. Le masquage de propriétés dans une sous-classe correspond à une **restriction de classe**.

Formalisme et exemple

La figure 2.28 montre un exemple d'héritage avec restriction et extension.

L'héritage multiple

Dans certains cas, il est nécessaire de faire hériter une même classe de deux classes « parentes » distinctes. Ce cas correspond à un **héritage multiple**.

Exemple

La figure 2.29 montre un exemple classique d'héritage multiple où la classe « Véhicule amphibie » hérite des classes « Véhicule terrestre » et « Véhicule marin ».

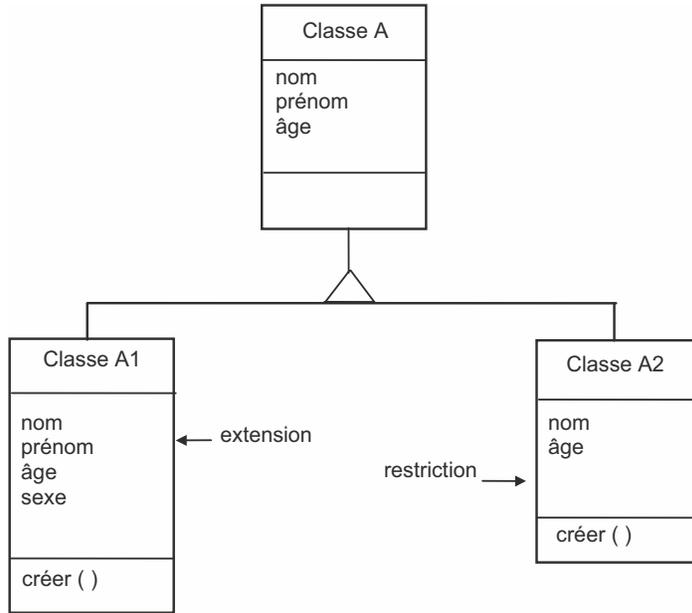


Figure 2.28 — Exemple d'héritage avec extension et restriction de propriétés

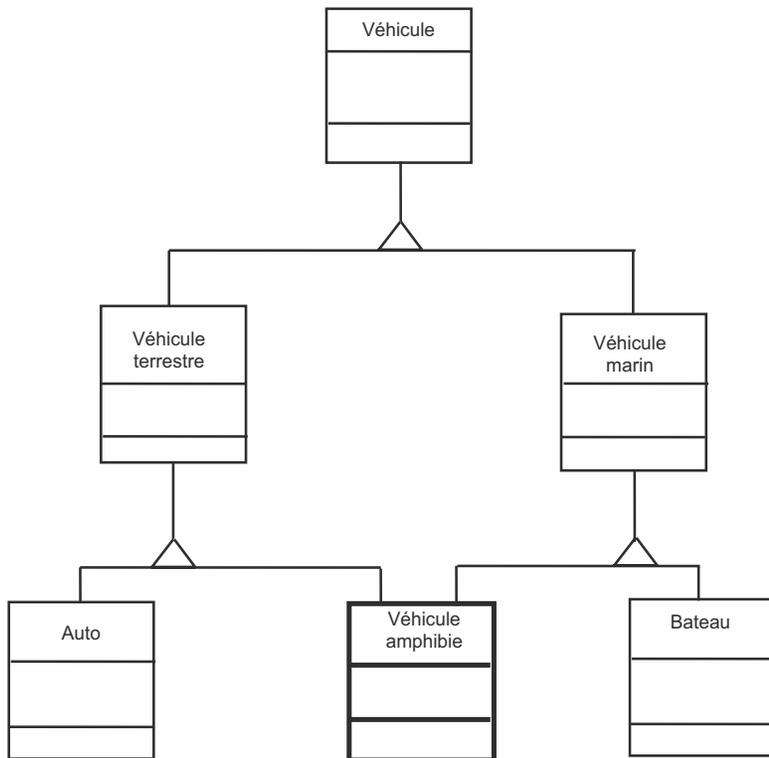


Figure 2.29 — Exemple de relation d'héritage multiple

2.1.7 Stéréotype de classe

UML propose un certain nombre de **stéréotypes** qui permettent de qualifier les profils d'utilisation.

Parmi ces stéréotypes, nous présentons ci-après quatre d'entre eux :

- « **Classe d'implémentation** » – Ce stéréotype est utilisé pour décrire des classes de niveau physique.
- « **Type** » – Ce stéréotype permet de spécifier des opérations applicables à un domaine d'objets. Exemple : Type Integer d'un langage de programmation.
- « **Utilitaire** » – Ce stéréotype qualifie toutes les fonctions utilitaires de base utilisées par les objets.
- « **MétaClasse** » – Ce stéréotype permet de regrouper des classes dans une famille de classe.

2.1.8 Exercices

Nous proposons au lecteur une série de quatre exercices sur le diagramme de classe ainsi qu'un exercice de synthèse (Locagite) pour lequel nous fournirons au fur et à mesure du parcours sur UML les principaux diagrammes.

Exercice 1

Énoncé

Il est demandé de représenter le diagramme de classe d'une gestion technique de documents. Chaque document est composé d'un ou plusieurs feuillets. Un feuillet comporte du texte et des objets géométriques qui constituent deux types d'objets graphiques supportant des opérations de type : sélectionner, copier, couper, coller et déplacer.

Nous considérons les quatre objets géométriques suivants : cercle, ellipse, carré, rectangle. Il est demandé d'utiliser les propriétés de la généralisation et la spécialisation afin de représenter au mieux ces objets géométriques.

Corrigé

La figure 2.30 propose au lecteur un corrigé type. D'autres variantes peuvent être envisagées notamment dans les choix de généralisation et spécialisation.

Exercice 2

Énoncé

Une entreprise nationale de vente d'appareil électroménager souhaite réaliser une première expérience d'analyse objet avec la méthode UML sur un petit sous-ensemble de son SI. Ce sous-ensemble concerne le suivi des personnels des agences locales implantées dans les régions. Chaque région est pilotée par une direction régionale qui a en charge un certain nombre d'agences locales. Une direction régionale est caractérisée par un code et un libellé.

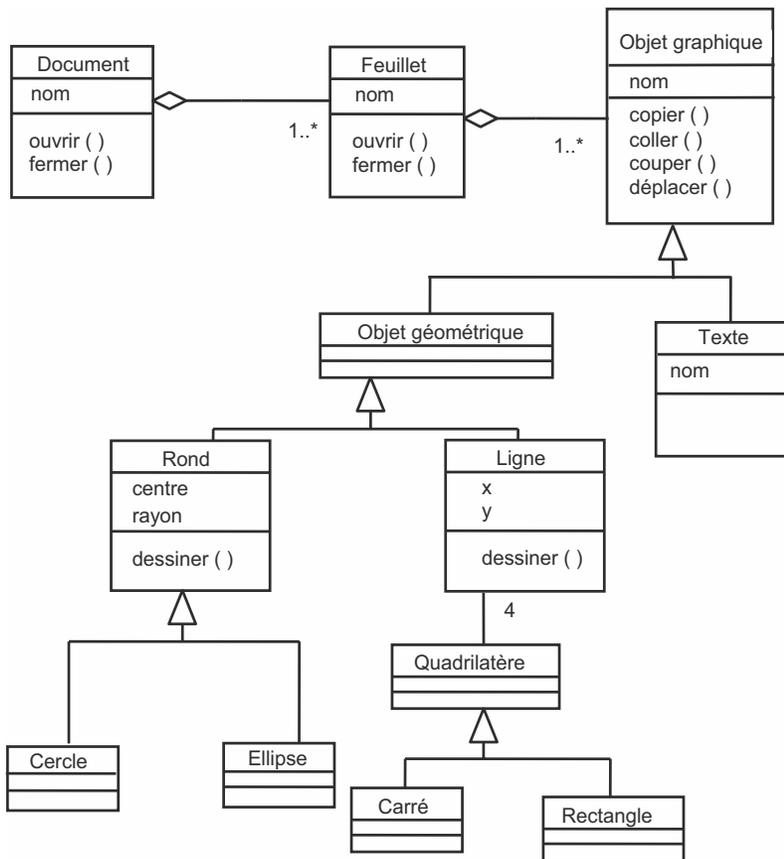


Figure 2.30 – Diagramme de classe de l'exercice 1

Chaque agence est caractérisée par un code, un intitulé, une date de création et une date de fermeture.

À une agence sont rattachées une à plusieurs personnes. Chaque personne est caractérisée par les données : numéro, qualité (M., Mme, Mlle), nom, prénom, date de naissance, date prévisionnelle d'arrivée, date d'arrivée et date de départ.

Il est demandé d'élaborer le diagramme de classe de ce premier sous-ensemble du SI de cette entreprise.

Corrigé

Les trois classes constituant ce système sont évidentes puisque déjà bien identifiées dans l'énoncé : Direction régionale, Agence et Personnel.

L'association entre Direction régionale et Agence est une agrégation qui matérialise une relation structurante entre ces classes. La relation entre Agence et Personnel est une association de un à plusieurs.

Les opérations mentionnées dans chaque classe correspondent aux opérations élémentaires nécessaires à la gestion du personnel des agences.

Le corrigé type est donné à la figure 2.31. Nous donnons aussi, figure 2.32, un exemple de diagramme d'objet associé à ce diagramme de classe.

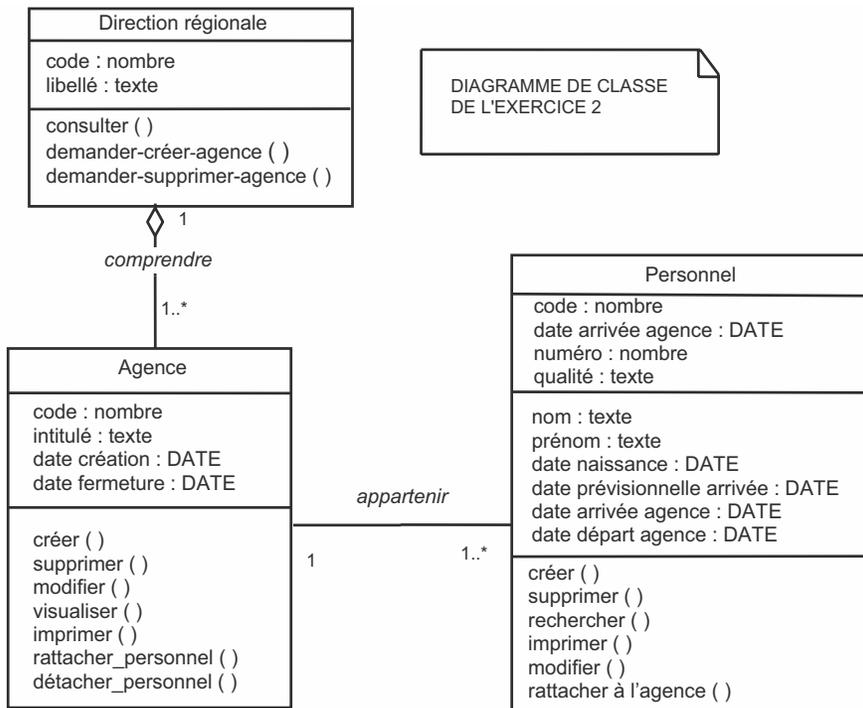


Figure 2.31 — Diagramme de classe de l'exercice 2

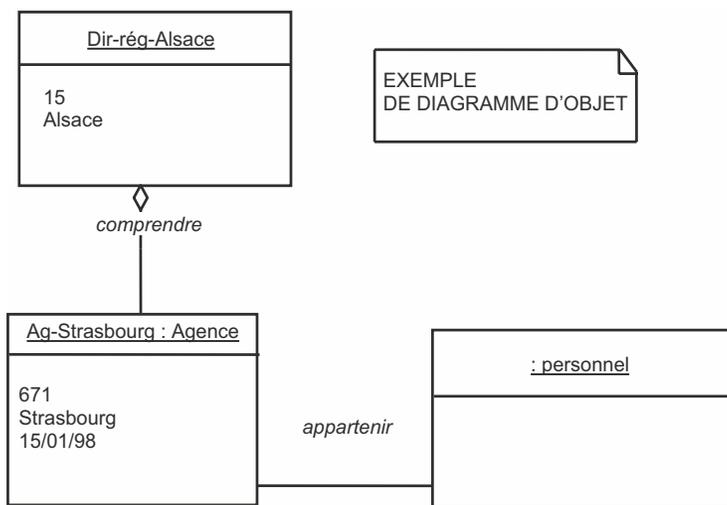


Figure 2.32 — Exemple de diagramme d'objet associé au diagramme de classe de l'exercice 2

Exercice 3

Énoncé

La société Forma possède un service qui gère la formation interne. Sa mission comporte plusieurs fonctions :

- Élaborer les catalogues qui décrivent les cours et donnent les dates prévisionnelles des sessions.
- Inscrire les personnes qui désirent participer aux sessions et leur envoyer leur convocation.
- Déterminer les formateurs qui vont animer les sessions et leur envoyer leur convocation (ces personnes sont choisies parmi celles qui peuvent enseigner un cours). Certaines sessions peuvent être animées par une personne d'un organisme extérieur.
- Faire le bilan des participations réelles aux formations.

Les cours sont déterminés afin de répondre aux besoins de formation internes. Certains cours sont organisés en filières, c'est-à-dire qu'ils doivent être suivis dans un certain ordre. Exemple : le cours ITE 16 (la démarche ITEOR OO) ne peut être suivi avant ITE 03 (UML). Les cours utilisent des documents référencés (tab. 2.1).

Tableau 2.1 – Documents référencés

Liste des attributs	
Code cours	N° catalogue
Date catalogue	N° document
Date session	N° session
Durée cours	Nom
État de la session (prévue, annulée, en cours, close)	Organisme extérieur
Intitulé du cours	Prénom
Lieu session	Service
Matricule	Titre document

Corrigé

La lecture du sujet et en particulier l'analyse des attributs indiqués conduisent à identifier rapidement les classes suivantes : Session, Cours, Catalogue, Document, Personne et Organisme.

Une réflexion complémentaire menée sur la classe Personne permet de distinguer en fait trois sous-classes spécialisées : PersonneExterne, PersonneIntNe (non enseignante) et PersonneIntEn (enseignante).

Le diagramme de classes peut être élaboré ensuite sans difficulté. La figure 2.33 donne le corrigé type de cet exercice.

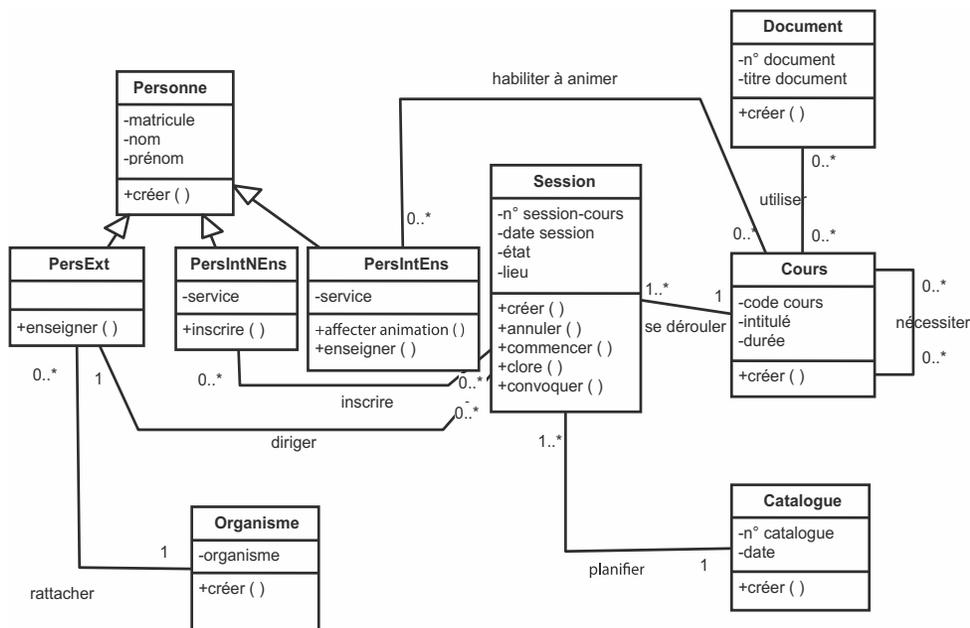


Figure 2.33 – Diagramme de classe de l'exercice 3

Exercice 4

Énoncé

Il vous est demandé de gérer les apports de raisin de la coopérative viticole de champagne. Le processus de traitements des apports de raisin correspond à un déroulement en plusieurs étapes.

- **Départ des camions pour le ramassage des raisins** – Le ramassage est organisé par le responsable du transport. Tous les matins, durant la campagne de ramassage, chaque chauffeur-livreur charge dans son camion un certain nombre de palettes vides et passe à la pesée. Le responsable du transport enregistre le départ du camion en lui affectant un n° d'apport ainsi que son poids à vide (la gestion des itinéraires ne fait pas partie de la présente étude).

Les camions ont été préalablement répertoriés par la coopérative avec leur capacité exprimée en nombre de palettes. Il est fréquent qu'un même camion soit utilisé pour plusieurs apports.

- **Retour des camions et pesée des apports** – L'arrivée d'un camion de palettes de caisses de raisins correspond à un apport de raisin. Les date et heure d'arrivée de cet apport sont soigneusement notées dans un registre avec le numéro d'immatriculation du camion.

Dès l'arrivée d'un apport, les palettes sont déchargées puis pesées. Le poids et le nombre de caisses par palettes sont soigneusement contrôlés par le responsable de la pesée puis enregistrés.

- **Constitution des lots** – Après la pesée des palettes reçues, le responsable de la coopérative répartit l'apport en lots. Chaque lot correspond aux palettes contenant les raisins de même cépage et de même cru. Un exemple de lot est fourni ci-après (tab.2.2) : il s'agit du troisième lot de l'apport numéro 3101 qui regroupe les palettes du Chardonnay en provenance de Cormigny.

Le raisin de Champagne se divise en trois cépages : le Chardonnay qui est un raisin blanc, le Pinot noir et le Pinot meunier qui sont des raisins noirs. Le cru est la provenance du raisin : il correspond à un nom de commune et est identifié par le numéro Insee de cette commune. Un cru possède un classement de 80 à 100 qui est remis en cause chaque année. Les classements annuels successifs d'un cru doivent être mémorisés.

Les palettes appartiennent toutes à la coopérative. Elles possèdent un numéro qui est peint sur le bois et qui permet de les identifier sans ambiguïté. Elles sont à la libre disposition des livreurs. Une même palette peut servir plusieurs fois au cours d'une même vendange.

Un lot concerne un livreur. Tous les livreurs sont obligatoirement connus de la coopérative. Un bulletin d'information professionnel leur est adressé régulièrement.

Les livreurs sont des coopérateurs ou des particuliers indépendants.

Les coopérateurs exploitent une ou plusieurs parcelles. Pour chaque livreur coopérateur, le pourcentage de sa production qu'il apporte à la coopérative est enregistré. En aucun cas un coopérateur ne peut être un indépendant et vice versa.

En ce qui concerne les livreurs indépendants, il est important de connaître le statut juridique qu'ils ont choisi. Celui-ci est identifié par le code du statut et se caractérise par un libellé (entreprise individuelle, société à responsabilité limitée, société coopérative d'exploitation agricole, etc.).

Tableau 2.2 – Exemple de lot

COOPÉRATIVE NOUVELLE DE CHAMPAGNE		
Apport n°: 3101		
Lot n°: 3		
Cépage : Chardonnay		
Cru : Cormigny		
Code Insee : 51231		
Nom du livreur : Dupond Laurent		
Numéro de palette	Nombre de caisses	Poids net en kg
428	8	459
102	14	642
14	10	670
123	12	578

Corrigé

La lecture du sujet et en particulier l'analyse des attributs indiqués conduisent à identifier rapidement les classes suivantes : Apport, Camion, Lot, Palette, Cépage, Commune, Livreur et Parcelle.

Le diagramme de classes peut être élaboré ensuite sans difficulté. La figure 2.34 donne le corrigé type de cet exercice.

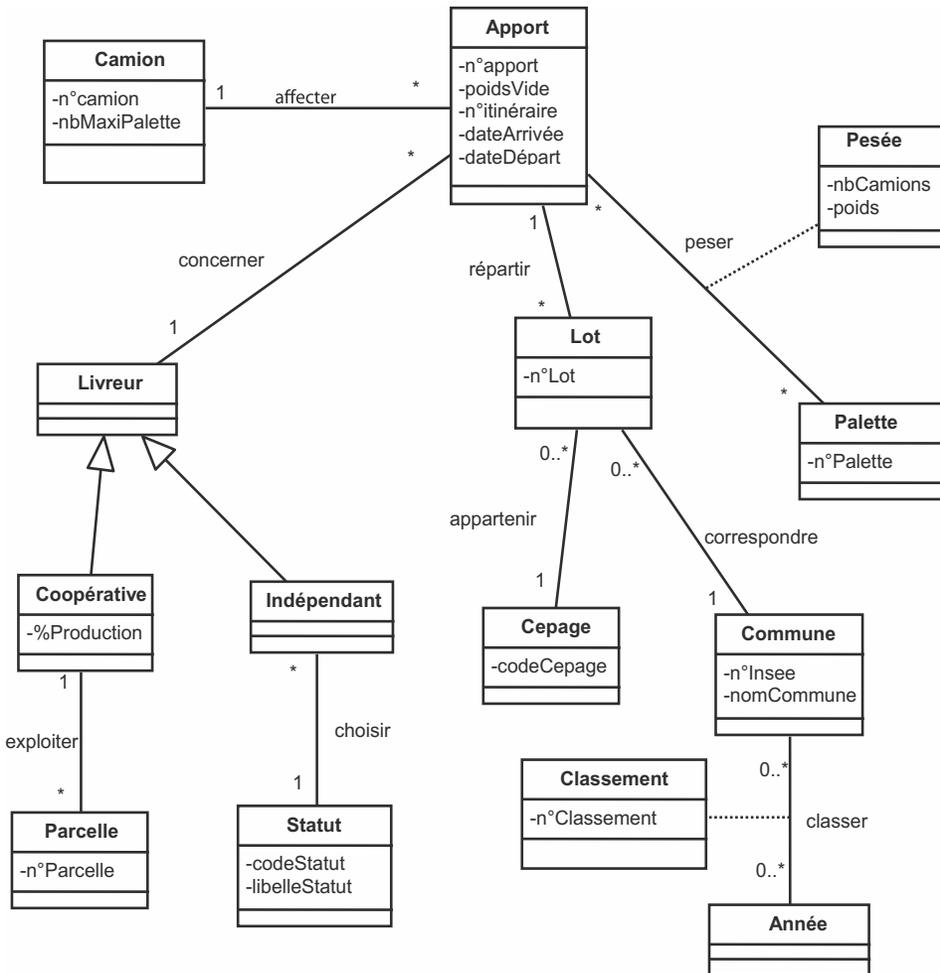


Figure 2.34 — Diagramme de classe de l'exercice 4

Exercice de synthèse : Locagite

Énoncé

« Locagite » est une association qui permet à divers propriétaires ruraux de mettre en location, à la semaine, des gîtes meublés. Elle publie annuellement un catalogue contenant les gîtes proposés par les propriétaires. Les gîtes doivent répondre à un certain nombre de critères qualité, correspondant à un nombre d'étoiles, qui sont vérifiées lors de l'adhésion du gîte et une fois tous les trois ans lors d'une visite de contrôle. Le propriétaire reçoit tous les ans un catalogue des gîtes, et peut modifier les informations qui le concernent (prix par saison, photo du gîte, nombre de personnes, de chambres, terrain...).

« Locagite » regroupe 450 gîtes en France, pour une moyenne de 12 semaines de réservation par gîte et par an.

« Locagite » propose aux propriétaires qui le souhaitent, un service central de réservation. Tous les ans, les propriétaires qui veulent utiliser ce service signent un contrat avec « Locagite », qui spécifie les périodes ouvertes à la location et la rémunération de la centrale de réservation en pourcentage de chaque location, ce dernier taux étant valable pour l'année et pour l'ensemble des gîtes. Le propriétaire, en signant le contrat, joint un relevé d'identité bancaire.

Le propriétaire ayant signé le contrat de la réservation centrale reçoit chaque mois un état des réservations fermes. Il reçoit aussi tous les mois un état des sommes encaissées par la centrale de réservation. Le virement bancaire des sommes dues, correspondant à l'état précédent, est envoyé en milieu du mois suivant.

Un client potentiel (que l'on peut appeler client réservataire) téléphone à la centrale de réservation pour réserver un gîte sur la base du catalogue. La centrale de réservation prend en compte la demande, et lui envoie un contrat de location ainsi qu'une demande d'acompte si un accord a été trouvé sur les dates de réservation. Le client réservataire renvoie le contrat signé accompagné de l'acompte : la réservation devient ferme. Un mois avant le séjour, le client locataire envoie le solde du paiement ; il reçoit alors une confirmation de séjour lui donnant les coordonnées de la personne à contacter pour convenir de son arrivée. Le client peut à tout moment annuler son séjour, 30 % des sommes versées ne sont pas remboursées. En cas de non-retour du contrat signé après 15 jours, la pré-réservation est automatiquement annulée.

Corrigé

Élaboration du diagramme de classe : l'analyse de l'ensemble des données disponibles nous permet de mettre en évidence les classes suivantes : Propriétaire, Gîte, Gîtes gérés, Catalogue, Activité, Période Location, Réservation, Client et Contrat Location. Le diagramme de classe correspondant est donné à la figure 2.35.

Tableau 2.3 — Principales informations portées par les documents échangés

Catalogue	
Année du catalogue N° du gîte Nom de la commune Adresse du gîte Animaux acceptés (O, N) NB d'étoiles NB de personnes acceptées	Capacité (nb. de chambres) Description du gîte (texte) Adresse et tél. du propriétaire (pour la réservation) ou tél. du service de réservation Tarifs semaine (HS, juin/sept/Vac Scol) Activités disponibles et distance (ex. : piscine à 3 km)
Contrat propriétaire	État mensuel des locations
N° de contrat propriétaire N° propriétaire Nom du propriétaire Adresse et tél. du propriétaire Référence du gîte Description du gîte (voir ci-dessus) Tarifs semaine (HS, juin/sept/Vac Scol) Périodes de location	N° propriétaire, nom du propriétaire, adresse et tél. du propriétaire Par contrat, par gîte et par réservation : N° de réservation Date d'arrivée Date de départ NB de nuits Nom et adresse du locataire NB d'adultes NB d'enfants Animaux (O, N) Montant reçu, Montant à recevoir
Contrat de location	
N° du contrat Référence du gîte Nom du propriétaire Ville ou village où se situe le gîte Dates d'arrivée et de départ Prix du séjour : Tarif de location = prix de la semaine * nombre de semaines Frais de dossiers (fixe) Assurance annulation (1,5 % de la location) Prix total	Composition de la famille Nom Adresse NB adultes NB d'enfants Animaux domestiques (O, N) Téléphone
Conditions de réservation	
Acompte à recevoir : date et montant Solde à recevoir : date et montant	

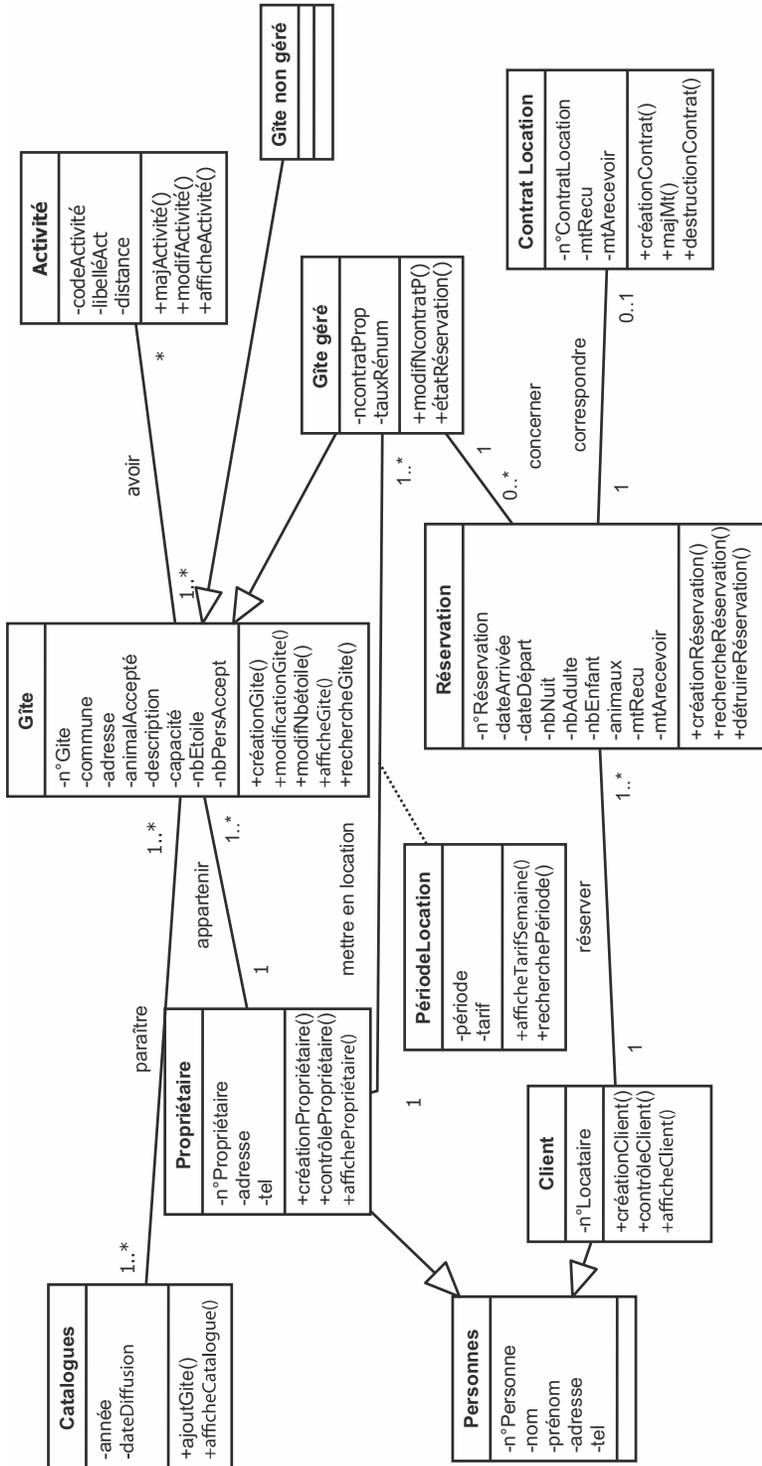


Figure 2.35 — Diagramme de classe de Locagite

2.2 DIAGRAMME DE COMPOSANT (DCP)

Le **diagramme de composant** permet de représenter les composants logiciels d'un système ainsi que les liens existant entre ces composants.

Les composants logiciels peuvent être de deux origines : soit des composants métiers propres à une entreprise soit des composants disponibles sur le marché comme par exemple les composants EJB, CORBA, .NET, WSDL.

2.2.1 Composant

Chaque **composant** est assimilé à un élément exécutable du système. Il est caractérisé par :

- un nom ;
- une spécification externe sous forme soit d'une ou plusieurs interfaces requises¹, soit d'une ou plusieurs interfaces fournies² ;
- un port de connexion.

Le port d'un composant représente le point de connexion entre le composant et une interface. L'identification d'un port permet d'assurer une certaine indépendance entre le composant et son environnement extérieur.

Formalisme général

Un composant est représenté (fig. 2.36) par un classeur avec le mot-clé « composant » ou bien par un classeur comportant une icône représentant un module.



Figure 2.36 — Formalisme général d'un composant

2.2.2 Les deux types de représentation et exemples

Deux types de représentation sont disponibles pour modéliser les composants : une représentation « boîte noire » et une représentation « boîte blanche ». Pour chaque représentation, plusieurs modélisations des composants sont proposées.

1. Une interface requise est une interface nécessaire au bon fonctionnement du composant.
2. Une interface fournie est une interface proposée par le composant aux autres composants.

Représentation « boîte noire »

C'est une vue externe du composant qui présente ses interfaces fournies et requises sans entrer dans le détail de l'implémentation du composant. Une boîte noire peut se représenter de différentes manières.

Connecteur d'assemblage

Une interface fournie se représente à l'aide d'un trait et d'un petit cercle et une interface requise à l'aide d'un trait et d'un demi-cercle. Ce sont les **connecteurs d'assemblage**.

Un exemple de modélisation avec les connecteurs d'assemblage est donné à la figure 2.37, le composant *Commande* possède deux interfaces fournies et deux interfaces requises.



Figure 2.37 — Représentation d'un connecteur d'assemblage

Connecteur d'interfaces

Une autre représentation peut être aussi utilisée en ayant recours aux **dépendances d'interfaces** *utilise* et *réalise* :

- pour une interface fournie, c'est une relation de réalisation partant du composant et allant vers l'interface ;
- pour une interface requise, c'est une dépendance avec le mot-clé « utilise » partant du composant et allant vers l'interface.

Un exemple de modélisation avec connecteurs d'interfaces est donné à la figure 2.38. Le composant *Commande* possède une interface fournie « *GestionCommande* » et une interface requise « *Produit* ».

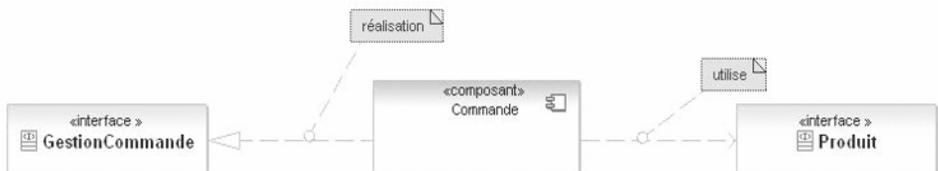


Figure 2.38 — Représentation d'un composant avec connecteur d'interfaces

Compartment

Une dernière manière de modéliser un composant avec une représentation boîte noire est de décrire sous forme textuelle les interfaces fournies et requises à l'intérieur d'un second **compartment**.

Un exemple de modélisation avec les compartiments est donné à la figure 2.39.

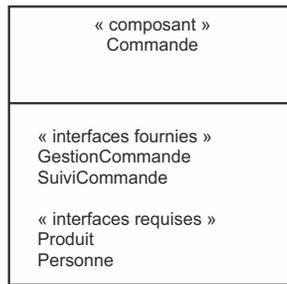


Figure 2.39 — Représentation d'un composant avec compartiments

Représentation « boîte blanche »

C'est une vue interne du composant qui décrit son implémentation à l'aide de classificateurs (classes, autres composants) qui le composent. Plusieurs modélisations sont possibles pour la représentation boîte blanche.

Compartment

Une manière de modéliser un composant avec une représentation boîte blanche est de décrire sous forme textuelle les interfaces fournies et requises à l'intérieur d'un compartiment, les classificateurs (classes, autres composants) dans un autre compartiment, les artefacts (élément logiciel : jar, war, ear, dll) qui représentent physiquement le composant dans un dernier **compartment**.

Un exemple de modélisation avec les compartiments est donné à la figure 2.40.

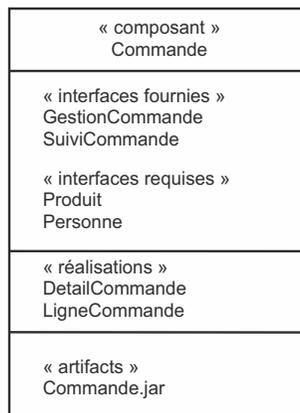


Figure 2.40 — Représentation boîte blanche avec compartiments

Dépendance

Une autre représentation interne du composant peut être aussi utilisée en ayant recours aux **dépendances**. Ainsi, les classificateurs qui composent le composant sont reliés à celui-ci par une relation de dépendance.

Les relations entre les classificateurs (association, composition, agrégation) sont aussi modélisées. Néanmoins, si elles sont trop complexes, elles peuvent être représentées sur un diagramme de classe relié au composant par une note.

Un exemple de modélisation boîte blanche avec les dépendances est donné à la figure 2.41.

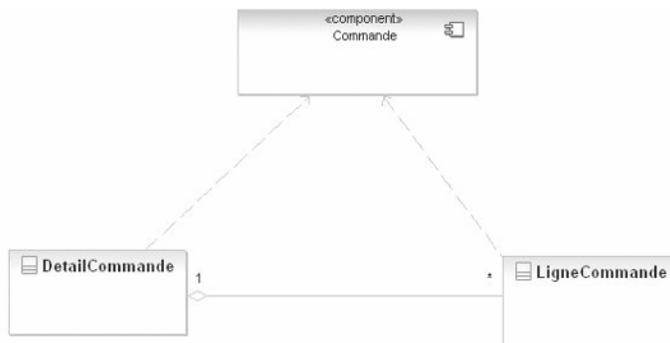


Figure 2.41 — Représentation boîte blanche avec dépendances

Ports et connecteurs

Le **port** est représenté par un petit carré sur le composant. Les connecteurs permettent de relier les ports aux classificateurs. Ils sont représentés par une association navigable et indiquent que toute information arrivée au port est transmise au classificateur.

Un exemple de représentation avec port et connecteur du composant *Commande* est donné à la figure 2.42.

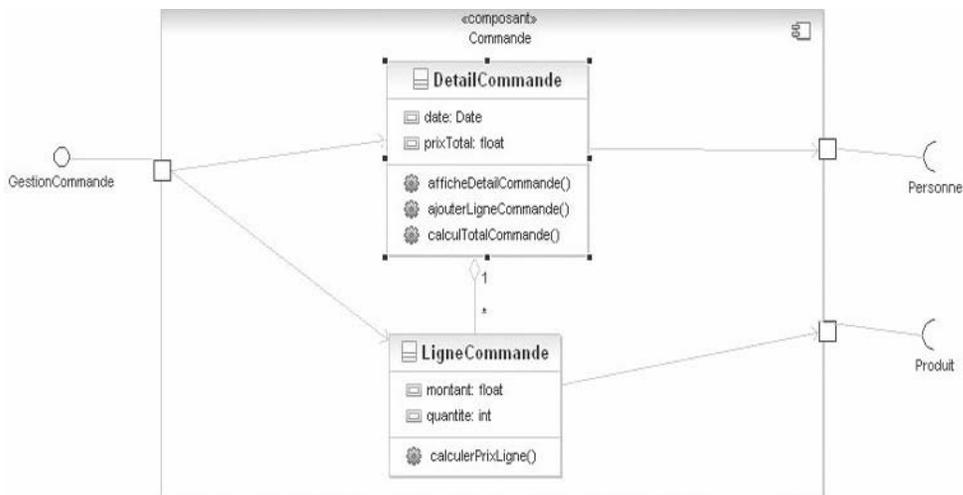


Figure 2.42 — Représentation boîte blanche avec connecteurs

Dans l'exemple de la figure 2.42, le composant `Commande` est constitué de deux classes (classificateur) reliées par une agrégation : `DetailCommande` et `LigneCommande`.

L'interface fournie `GestionCommande` est accessible de l'extérieur *via* un port et permet d'accéder *via* les connecteurs aux opérations des deux classes `DetailCommande` et `LigneCommande` (`ajouterLigneCommande`, `calculTotalCommande`, `calculPrixLigne`).

L'interface requise `Personne` est nécessaire pour l'affichage du détail de la commande et est accessible *via* un port du composant `Commande`.

L'interface requise `Produit` est nécessaire pour le calcul du prix de la ligne de commande et est accessible *via* un port du composant `Commande`.

2.3 DIAGRAMME DE DÉPLOIEMENT (DPL)

Le **diagramme de déploiement** permet de représenter l'architecture physique supportant l'exploitation du système. Cette architecture comprend des nœuds correspondant aux supports physiques (serveurs, routeurs...) ainsi que la répartition des artefacts logiciels (bibliothèques, exécutables...) sur ces nœuds. C'est un véritable réseau constitué de nœuds et de connexions entre ces nœuds qui modélise cette architecture.

2.3.1 Nœud

Un **nœud** correspond à une ressource matérielle de traitement sur laquelle des artefacts seront mis en œuvre pour l'exploitation du système. Les nœuds peuvent être interconnectés pour former un réseau d'éléments physiques.

Formalisme et exemple

Un nœud ou une instance de nœud se représente par un cube ou parallélépipède (fig. 2.43).

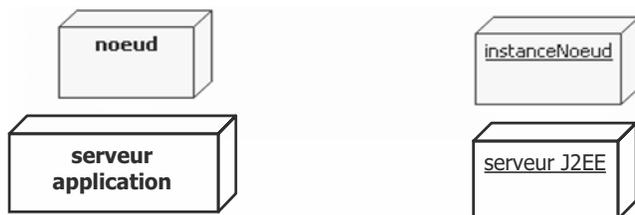


Figure 2.43 — Représentation de nœuds

Compléments sur la description d'un nœud

Il est possible de représenter des nœuds spécialisés. UML propose en standard les deux types de nœuds suivants :

- **Unité de traitement** – Ce nœud est une unité physique disposant de capacité de traitement sur laquelle des artefacts peuvent être déployés.
Une unité de traitement est un nœud spécialisé caractérisé par le mot-clé « device ».
- **Environnement d'exécution** – Ce nœud représente un environnement d'exécution particulier sur lequel certains artefacts peuvent être exécutés.
Un environnement d'exécution est un nœud spécialisé caractérisé par le mot-clé « executionEnvironment ».

2.3.2 Artefact

Un **artefact** est la spécification d'un élément physique qui est utilisé ou produit par le processus de développement du logiciel ou par le déploiement du système. C'est donc un élément concret comme par exemple : un fichier, un exécutable ou une table d'une base de données.

Un artefact peut être relié à d'autres artefacts par notamment des liens de dépendance.

Formalisme et exemple

Un artefact se représente par un rectangle (fig. 2.44) caractérisé par le mot-clé « artifact » et/ou une icône particulière dans le coin droit du rectangle.



Figure 2.44 – Représentation d'un artefact

Deux compléments de description sont proposés par UML pour la représentation des artefacts.

2.3.3 Spécification de déploiement

Une **spécification de déploiement** peut être associée à chaque artefact. Elle permet de préciser les conditions de déploiement de l'artefact sur le nœud sur lequel il va être implanté.

Formalisme et exemple

Une spécification de déploiement se représente par un rectangle avec le mot-clé « deployment spec ». Un exemple d'un artefact avec une spécification de déploiement est donné à la figure 2.45.

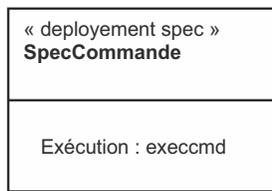


Figure 2.45 – Exemple d'un artefact avec spécification de déploiement

2.3.4 Liens entre un artefact et les autres éléments du diagramme

Il existe deux manières de représenter le lien entre un artefact et son nœud d'appartenance :

- **Représentation inclusive** – Dans cette représentation, un artefact est représenté à l'intérieur du nœud auquel il se situe physiquement. Un exemple est donné à la figure 2.46.

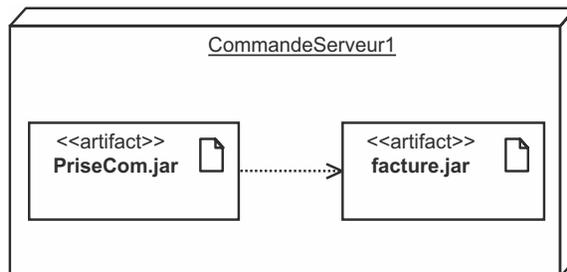


Figure 2.46 – Exemple de représentation inclusive d'artefact

- **Représentation avec un lien de dépendance typé « deploy »** – Dans ce cas l'artefact est représenté à l'extérieur du nœud auquel il appartient avec un lien de dépendance entre l'artefact et le nœud typé avec le mot-clé « deploy ». Un exemple est donné à la figure 2.47.

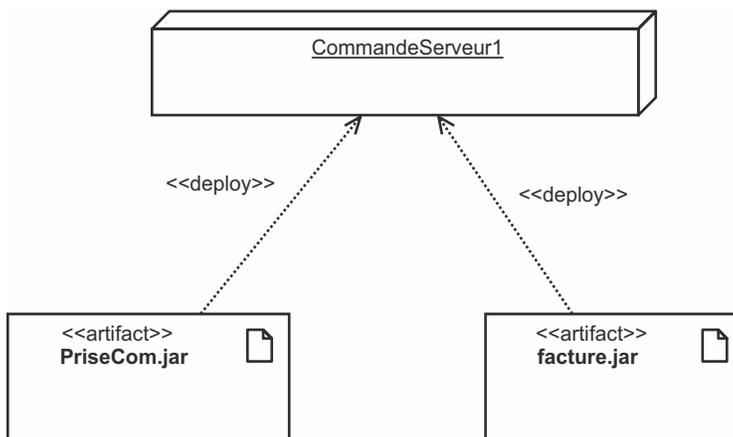


Figure 2.47 – Exemple de représentation d'artefact avec lien de dépendance

Un artefact peut représenter un ou plusieurs éléments d'un modèle. Le qualificatif « manifest » permet d'indiquer ce type de dépendance.

2.3.5 Représentation et exemples

Le diagramme de déploiement représente les nœuds de l'architecture physique ainsi que l'affectation des artefacts sur les nœuds conformément aux règles de déploiement définies. Un premier exemple est donné à la figure 2.48.

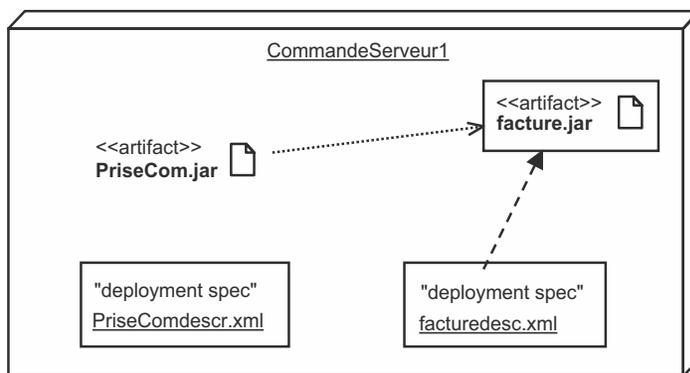


Figure 2.48 — Exemple de représentation d'un diagramme de déploiement

Un second exemple relatif à une implémentation d'une architecture J2EE avec quatre nœuds est donné à la figure 2.49.

Dans l'exemple de la figure 2.49, plusieurs composants sont déployés.

- Un serveur web où se trouvent les éléments statiques du site dans une archive : images, feuilles de style, pages html (static.zip).
- Un serveur d'application « front » sur le quel est déployée l'archive « front.ear » composée de l'application web « front.war » et d'autres composants nécessaires au fonctionnement de cette archive web comme « client-ejb.jar » (classes permettant l'appel aux EJB) et « commun.jar » (classes communes aux deux serveurs d'application).
- Un serveur d'application métier sur lequel sont déployés les composants : «.ejb.jar ». Ils sont packagés dans l'archive « metier.ear ». Deux autres archives sont nécessaires au fonctionnement des EJB : « dao.jar » (classes qui permettent l'accès à la base de données) et « commun.jar » (classes communes aux deux serveurs d'application).
- Un serveur BDD (base de données) sur lequel sont stockées des procédures stockées PL/SQL : « scripts.sql ».

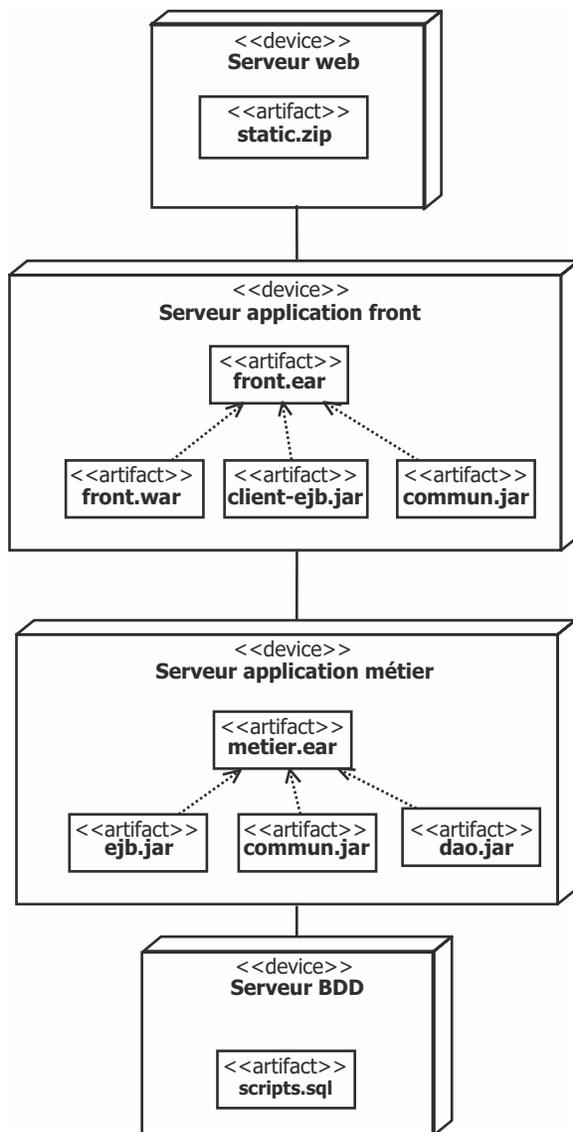


Figure 2.49 — Exemple de diagramme de déploiement comportant quatre nœuds

2.4 DIAGRAMME DE PAQUETAGE (DPA)

2.4.1 Paquetage

Un **paquetage** regroupe des éléments de la modélisation appelés aussi membres, portant sur un sous-ensemble du système. Le découpage en paquetage doit traduire un découpage logique du système à construire qui corresponde à des espaces de nommage homogènes.

Les éléments d'un paquetage peuvent avoir une visibilité déclarée soit de type public (+) soit privé (-).

Un paquetage peut importer des éléments d'un autre paquetage. Un paquetage peut être fusionné avec un autre paquetage.

Formalisme et exemple

La figure 2.50 montre le formalisme général d'un paquetage et les trois manières de présenter un paquetage.

- **Représentation globale** – Le nom du paquetage se trouve à l'intérieur du grand rectangle.
- **Représentation détaillée** – Les membres du paquetage sont représentés et le nom du paquetage d'ensemble s'inscrit dans le petit rectangle.
- **Représentation éclatée** – Les membres du paquetage sont reliés par un lien connecté au paquetage par le symbole \oplus .

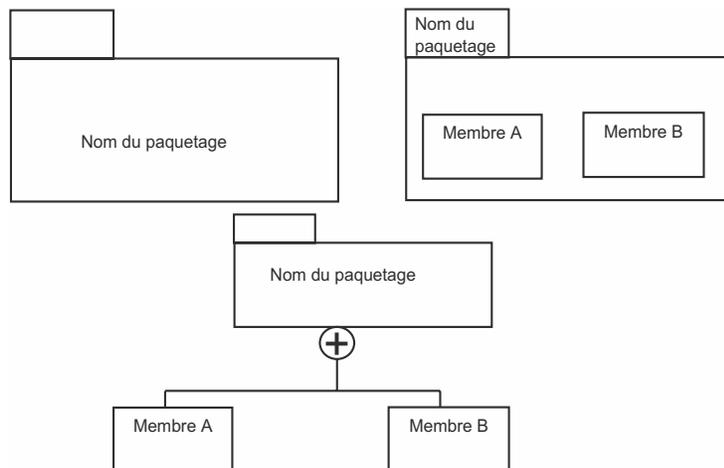


Figure 2.50 – Formalisme de représentation de paquetages

La figure 2.51 donne un exemple de représentation éclatée.

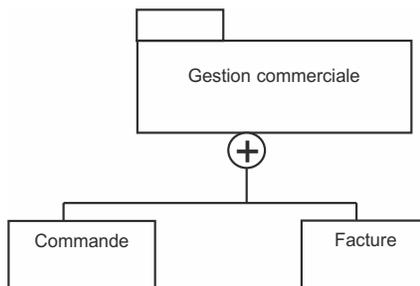


Figure 2.51 – Exemple de représentation éclatée d'un paquetage

2.4.2 Dépendance entre paquetages

La **dépendance entre paquetages** peut être qualifiée par un niveau de visibilité qui est soit public soit privé. Par défaut le type de visibilité est public.

À chaque type de visibilité est associé un lien de dépendance. Les deux types de dépendances entre paquetages sont :

- « **import** » – Ce type de dépendance permet, pour un paquetage donné, d'importer l'espace de nommage d'un autre paquetage. Ainsi tous les membres du paquetage donné ont accès à tous les noms des membres du paquetage importé sans avoir à utiliser explicitement le nom du paquetage concerné. Ce type de dépendance correspond à un lien ayant une visibilité « public ».
- « **access** » – Ce type de dépendance permet, pour un paquetage donné, d'avoir accès à l'espace de nommage d'un paquetage cible. L'espace de nommage n'est donc pas importé et ne peut être transmis à d'autres paquetages par transitivité. Ce type de dépendance correspond à un lien ayant une visibilité « privé ».

Un exemple de dépendance entre paquetages mettant en jeu les niveaux de visibilité est donné à la figure 2.52.

Dans cet exemple, les éléments de Clients externes sont importés dans Domaine client et ensuite dans Domaine tiers. Cependant, les éléments de Clients internes sont seulement accessibles par le paquetage Domaine client et donc pas à partir du paquetage Domaine tiers.

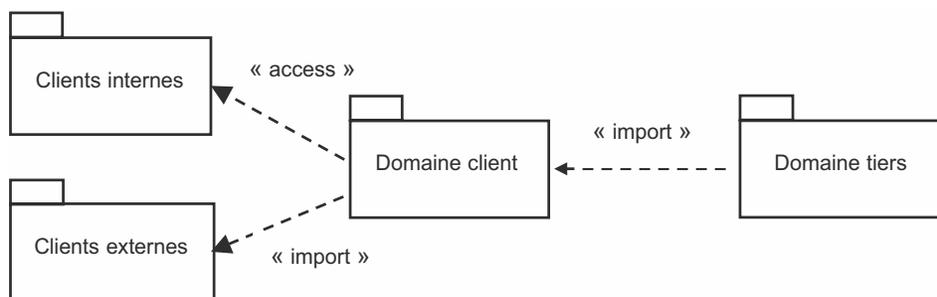


Figure 2.52 — Exemple de liens de dépendance entre paquetages

2.4.3 Représentation et exemples

En reprenant l'exemple type de l'exercice de synthèse Locagite, nous donnons (fig. 2.53) une représentation des liens de dépendance entre paquetages.

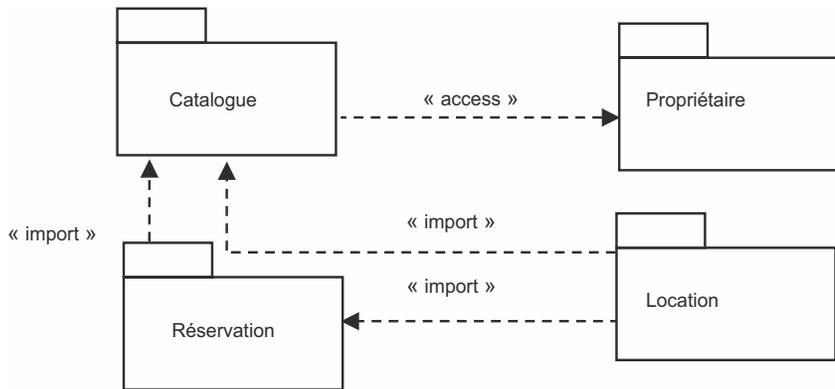


Figure 2.53 — Exemple de liens de dépendance entre paquets de Locagite

Enfin, UML propose aussi une opération de fusion entre deux paquets. Le lien de dépendance comporte dans ce cas le mot-clé « merge ».

Ce type de lien permet de fusionner deux paquets et d'obtenir ainsi un paquetage contenant la fusion des deux paquetages d'origine. Pour bien clarifier cette opération, il est important de qualifier par des rôles les paquetages dans cette fusion. Ainsi trois rôles sont à distinguer :

- le paquetage à fusionner (entrant dans la fusion, mais préservé après la fusion) ;
- le paquetage recevant (paquetage d'origine avant la fusion, mais non conservé après la fusion) ;
- le paquetage résultat (paquetage contenant le résultat de la fusion et écrasant le contenu du paquetage d'origine).

Un exemple type de fusion entre deux paquetages est donné à la figure 2.54.

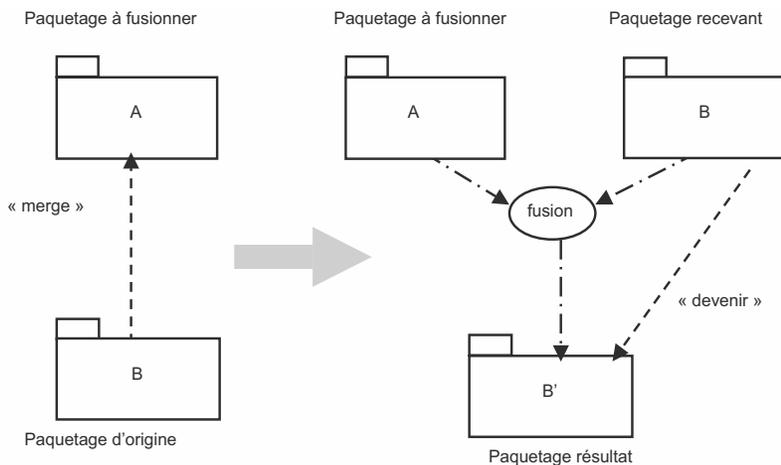


Figure 2.54 — Exemple de liens de fusion entre paquetages

2.5 DIAGRAMME DE STRUCTURE COMPOSITE (DSC)

Le **diagramme de structure composite** permet de décrire des collaborations d'instances (de classes, de composants...) constituant des fonctions particulières du système à développer.

2.5.1 Collaboration

Une **collaboration** représente un assemblage de rôles d'éléments qui interagissent en vue de réaliser une fonction donnée. Il existe deux manières de représenter une collaboration :

- représentation par une collaboration de rôles,
- représentation par une structure composite : le diagramme de structure composite.

2.5.2 Représentation et exemples

Représentation par une collaboration de rôles

Dans ce cas, une collaboration est formalisée par une ellipse en pointillé dans laquelle on fait figurer les rôles des éléments qui interagissent en vue de réaliser la fonction souhaitée (fig. 2.55). Dans cet exemple, la fonction *Persistence objets métier* résulte d'une collaboration entre deux rôles d'éléments :

- mapping : classeMétier,
- stockage : tableBDD.

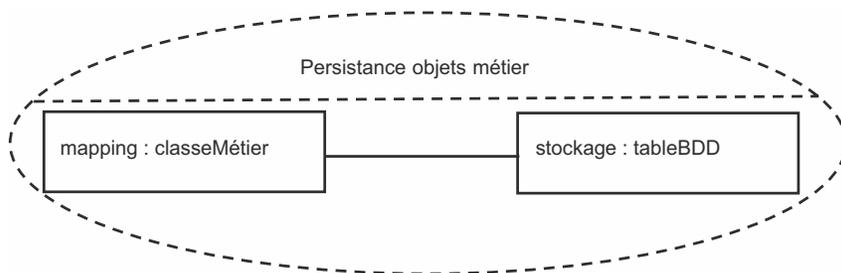


Figure 2.55 — Exemple de représentation d'une structure composite à l'aide d'une collaboration de rôles

Représentation par un diagramme de structure composite

Cette nouvelle représentation (fig. 2.56) permet de montrer plus explicitement les éléments de la collaboration :

- la collaboration représentée par une ellipse en pointillé ;
- les éléments participant à la collaboration (classe, composant...) représentés à l'extérieur de la collaboration ;
- les rôles considérés dans chaque participation représentés sur les liens entre les éléments participants et la collaboration.

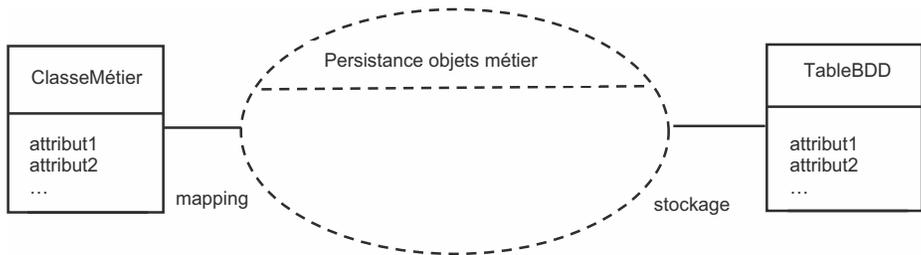


Figure 2.56 — Exemple de représentation de collaboration d'instances par un diagramme de structure composite

Dans cet exemple, la fonction Persistence objets métier résulte d'une collaboration entre la classe ClasseMétier considérée suivant le rôle mapping et la classe TableBDD considérée suivant le rôle stockage.

Cette représentation permet aussi de préciser les seuls attributs des classes participantes qui sont considérés suivant les rôles pris en compte.

3

Les diagrammes comportementaux

Les diagrammes comportementaux sont focalisés sur la description de la partie dynamique du système à modéliser. Sept diagrammes sont proposés par UML 2 pour assurer cette description :

- le diagramme des cas d'utilisation (DCU),
- le diagramme d'état-transition (machine d'état, DET),
- le diagramme d'activité (DAC),
- le diagramme de séquence (DSE),
- le diagramme de communication (DCO),
- le diagramme global d'interaction (DGI),
- le diagramme de temps (DTP).

3.1 DIAGRAMME DES CAS D'UTILISATION (DCU)

3.1.1 Présentation générale et concepts de base

Les cas d'utilisation ont été définis initialement par Ivar Jacobson en 1992 dans sa méthode OOSE. Les cas d'utilisation constituent un moyen de recueillir et de décrire les besoins des acteurs du système. Ils peuvent être aussi utilisés ensuite comme moyen d'organisation du développement du logiciel, notamment pour la structuration et le déroulement des tests du logiciel.

Un cas d'utilisation permet de décrire l'interaction entre les acteurs (utilisateurs du cas) et le système. La description de l'interaction est réalisée suivant le point de vue de l'utilisateur.

La représentation d'un cas d'utilisation met en jeu trois concepts : l'acteur, le cas d'utilisation et l'interaction entre l'acteur et le cas d'utilisation.

Acteur

Un **acteur** est un utilisateur type qui a toujours le même comportement vis-à-vis d'un cas d'utilisation. Ainsi les utilisateurs d'un système appartiennent à une ou plusieurs classes d'acteurs selon les rôles qu'ils tiennent par rapport au système.

Une même personne physique peut se comporter en autant d'acteurs différents que le nombre de rôles qu'elle joue vis-à-vis du système.

Ainsi par exemple, l'administrateur d'un système de messagerie peut être aussi utilisateur de cette même messagerie. Il sera considéré, en tant qu'acteur du système, dans le rôle d'administrateur d'une part et dans celui d'utilisateur d'autre part.

Un acteur peut aussi être un système externe avec lequel le cas d'utilisation va interagir.

Formalisme et exemple

Un acteur peut se représenter symboliquement par un « bonhomme » et être identifié par son nom. Il peut aussi être formalisé par une classe stéréotypée « acteur » (fig. 3.1).



Figure 3.1 — Représentations d'un acteur

Cas d'utilisation et interaction

Un **cas d'utilisation** correspond à un certain nombre d'actions que le système devra exécuter en réponse à un besoin d'un acteur. Un cas d'utilisation doit produire un résultat observable pour un ou plusieurs acteurs ou parties prenantes du système.

Une **interaction** permet de décrire les échanges entre un acteur et un cas d'utilisation.

Formalisme et exemple

Un cas d'utilisation se représente par un ovale dans lequel figure son intitulé.

L'interaction entre un acteur et un cas d'utilisation se représente comme une association. Elle peut comporter des multiplicités comme toute association entre classes (voir § 2.1 Diagramme de classe).

Le formalisme de base de représentation d'un cas d'utilisation est donné à la figure 3.2.

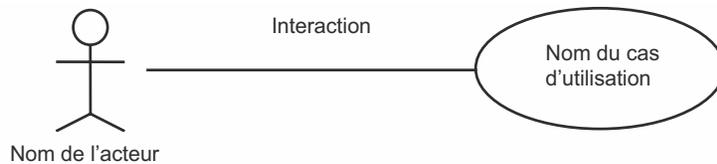


Figure 3.2 — Formalisme de base de représentation d'un cas d'utilisation

Chaque cas d'utilisation doit être décrit sous forme textuelle afin de bien identifier les traitements à réaliser par le système en vue de la satisfaction du besoin exprimé par l'acteur.

3.1.2 Représentation du diagramme des cas d'utilisation

Tout système peut être décrit par un certain nombre de cas d'utilisation correspondant aux besoins exprimés par l'ensemble des utilisateurs. À chaque utilisateur, vu comme acteur, correspondra un certain nombre de cas d'utilisation du système. L'ensemble de ces cas d'utilisation se représente sous forme d'un diagramme.

Exemple

La figure 3.3 montre un exemple d'un système de messagerie comportant quatre cas d'utilisation.

Nous verrons, dans la suite de la présentation d'UML, qu'un cas d'utilisation peut avoir une ou plusieurs instances représentées par des scénarios. Chaque scénario fait l'objet lui-même d'un diagramme de séquence ou de communication.

En conclusion, nous dirons qu'un système est caractérisé par son comportement vis-à-vis de ses utilisateurs. Ce comportement se représente sous forme d'un ensemble de cas d'utilisation qui correspond aux besoins des acteurs de ce système.

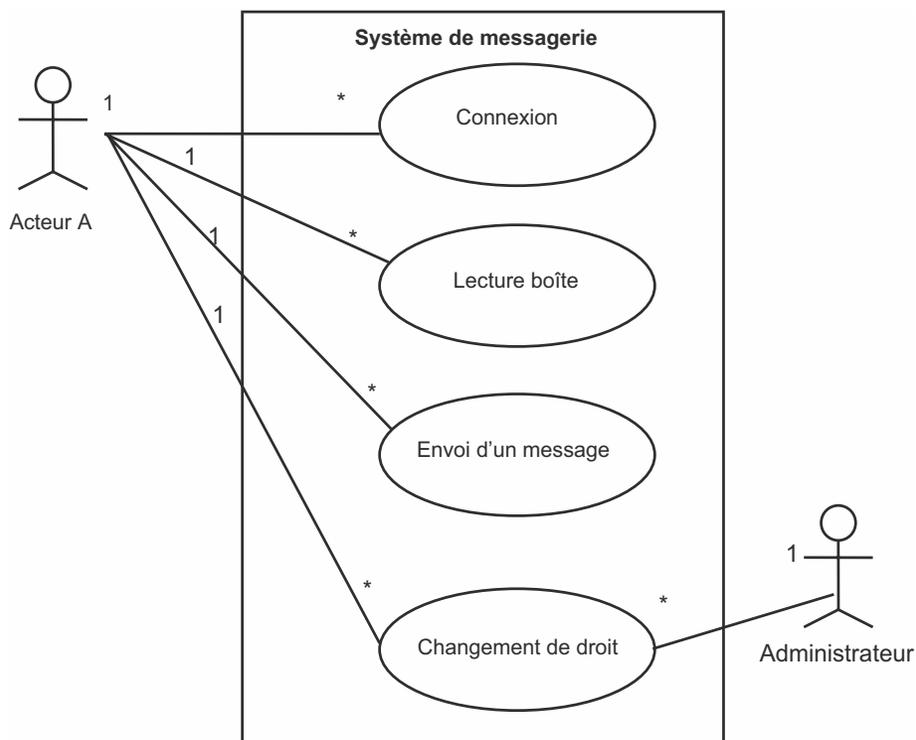


Figure 3.3 — Exemple d'un système de messagerie comportant quatre cas d'utilisation

3.1.3 Relations entre cas d'utilisation

Afin d'optimiser la formalisation des besoins en ayant recours notamment à la réutilisation de cas d'utilisation, trois relations peuvent être décrites entre cas d'utilisation : une relation d'inclusion (« include »), une relation d'extension (« extend ») et une relation de généralisation.

Relation d'inclusion

Une **relation d'inclusion** d'un cas d'utilisation A par rapport à un cas d'utilisation B signifie qu'une instance de A contient le comportement décrit dans B.

Formalisme et exemple

La figure 3.4 donne le formalisme et un exemple d'une relation d'inclusion entre cas d'utilisation.

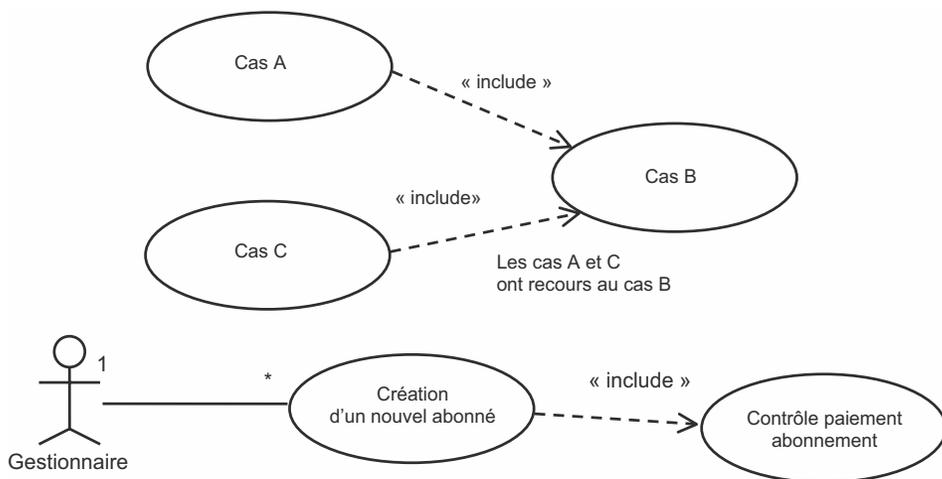


Figure 3.4 — Formalisme et exemple de la relation d'inclusion entre cas d'utilisation

Relation d'extension

Une **relation d'extension** d'un cas d'utilisation A par un cas d'utilisation B signifie qu'une instance de A peut être étendue par le comportement décrit dans B. Deux caractéristiques sont à noter :

- le caractère optionnel de l'extension dans le déroulement du cas d'utilisation standard (A) ;
- la mention explicite du point d'extension dans le cas d'utilisation standard.

Formalisme et exemple

La figure 3.5 donne un exemple d'une relation d'extension entre cas d'utilisation.

Une note peut être ajoutée à la représentation du cas d'utilisation permettant d'explicitier la condition.

Relation de généralisation

Une **relation de généralisation** de cas d'utilisation peut être définie conformément au principe de la spécialisation-généralisation déjà présentée pour les classes.

Formalisme et exemple

La figure 3.6 donne un exemple d'une relation de généralisation de cas d'utilisation.

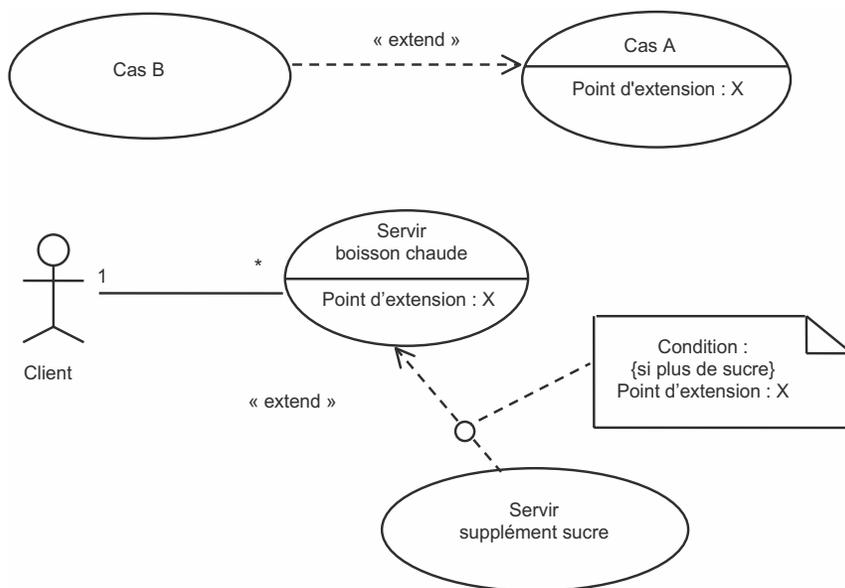


Figure 3.5 – Formalisme et exemple d'une relation d'extension entre cas d'utilisation

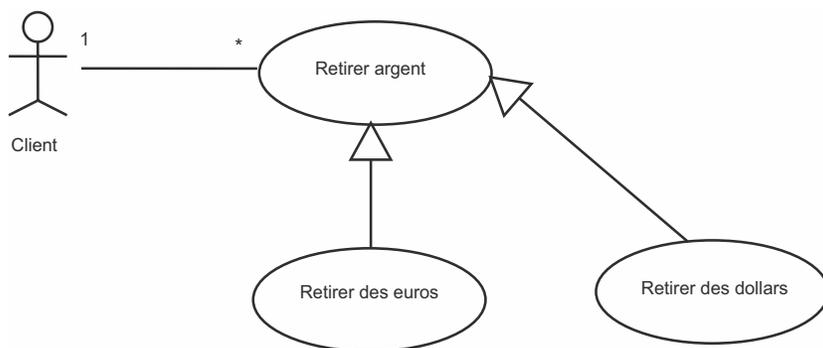


Figure 3.6 – Exemple d'une relation de généralisation de cas d'utilisation

3.1.4 Description textuelle d'un cas d'utilisation

À chaque cas d'utilisation doit être associée une description textuelle des interactions entre l'acteur et le système et les actions que le système doit réaliser en vue de produire les résultats attendus par les acteurs.

UML ne propose pas de présentation type de cette description textuelle. Cependant, les travaux menés par Alistair Cockburn [Cockburn2001] sur ce sujet constituent une référence en la matière et tout naturellement nous reprenons ici l'essentiel de cette présentation.

La description textuelle d'un cas d'utilisation est articulée en six points :

- **Objectif** – Décrire succinctement le contexte et les résultats attendus du cas d'utilisation.
- **Acteurs concernés** – Le ou les acteurs concernés par le cas doivent être identifiés en précisant globalement leur rôle.
- **Pré conditions** – Si certaines conditions particulières sont requises avant l'exécution du cas, elles sont à exprimer à ce niveau.
- **Post conditions** – Par symétrie, si certaines conditions particulières doivent être réunies après l'exécution du cas, elles sont à exprimer à ce niveau. Pour notre part, par souci de simplification nous n'avons pas traité ce point dans les exercices et études de cas présentés.
- **Scénario nominal** – Il s'agit là du scénario principal qui doit se dérouler sans incident et qui permet d'aboutir au résultat souhaité.
- **Scénarios alternatifs** – Les autres scénarios, secondaires ou correspondant à la résolution d'anomalies, sont à décrire à ce niveau. Le lien avec le scénario principal se fait à l'aide d'une numérotation hiérarchisée (1.1a, 1.1b...) rappelant le numéro de l'action concernée.

3.1.5 Exercices

Exercice 1

Énoncé

Une bibliothèque universitaire souhaite automatiser sa gestion. Cette bibliothèque est gérée par un gestionnaire chargé des inscriptions et des relances des lecteurs quand ceux-ci n'ont pas rendu leurs ouvrages au-delà du délai autorisé. Les bibliothécaires sont chargés de gérer les emprunts et la restitution des ouvrages ainsi que l'acquisition de nouveaux ouvrages.

Il existe trois catégories d'abonné. Tout d'abord les étudiants qui doivent seulement s'acquitter d'une somme forfaitaire pour une année afin d'avoir droit à tous les services de la bibliothèque. L'accès à la bibliothèque est libre pour tous les enseignants. Enfin, il est possible d'autoriser des étudiants d'une autre université à s'inscrire exceptionnellement comme abonné moyennant le versement d'une cotisation. Le nombre d'abonné externe est limité chaque année à environ 10 % des inscrits.

Un nouveau service de consultation du catalogue général des ouvrages doit être mis en place.

Les ouvrages, souvent acquis en plusieurs exemplaires, sont rangés dans des rayons de la bibliothèque. Chaque exemplaire est repéré par une référence gérée dans le catalogue et le code du rayon où il est rangé.

Chaque abonné ne peut emprunter plus de trois ouvrages. Le délai d'emprunt d'un ouvrage est de trois semaines, il peut cependant être prolongé exceptionnellement à cinq semaines.

Il est demandé d'élaborer le diagramme des cas d'utilisation (DCU).

Corrigé

Représentation du DCU – La figure 3.7 propose un corrigé-type de cet exercice. Six cas d'utilisation peuvent être identifiés :

- inscription à la bibliothèque,
- consultation du catalogue,
- emprunt d'ouvrages,
- restitution d'ouvrages,
- approvisionnement d'ouvrages,
- relance emprunteur.

Comme le montre la figure 3.7, cinq types d'acteurs peuvent être identifiés :

- étudiant,
- externe,
- emprunteur,
- gestionnaire,
- bibliothécaire.

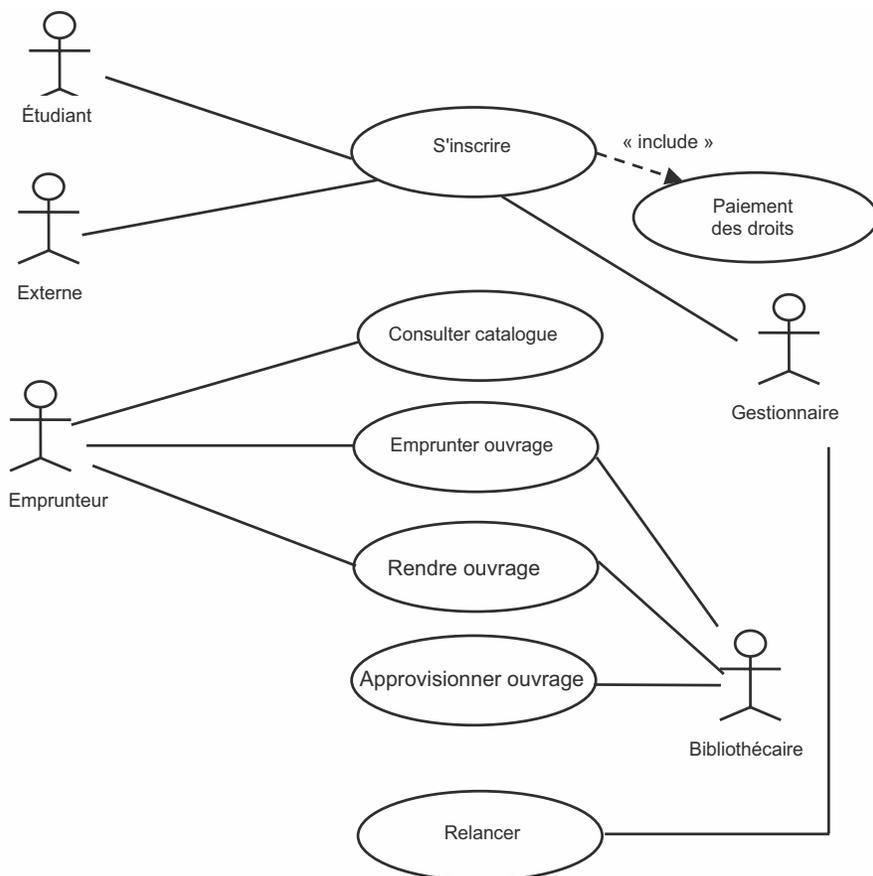


Figure 3.7 – Corrigé type de l'exercice 1 sur les cas d'utilisation

Exercice de synthèse

En reprenant l'énoncé de l'exercice de synthèse Locagite, nous allons élaborer le diagramme des cas d'utilisation des quatre activités décrites :

- **Catalogue**, cette activité se décompose en trois cas d'utilisation :
 - cas d'utilisation 1.1 : Gestion annuelle du catalogue
 - cas d'utilisation 1.2 : Publication du catalogue
 - cas d'utilisation 1.3 : Contrôle annuel de l'état du gîte
- **Propriétaire**, cette activité se décompose en deux cas d'utilisation :
 - cas d'utilisation 2.1 : Gestion propriétaire
 - cas d'utilisation 2.2 : Authentification propriétaire
- **Réservation**, cette activité se décompose en deux cas d'utilisation :
 - cas d'utilisation 3.1 : Gestion des réservations
 - cas d'utilisation 3.2 : Authentification client
- **Location**, cette activité se décompose en deux cas d'utilisation :
 - cas d'utilisation 4.1 : Gestion des locations
 - cas d'utilisation 4.2 : Gestion des annulations

Pour ces cas d'utilisation considérés, quatre acteurs types externes peuvent être identifiés :

- le propriétaire,
- le propriétaire adhérent (qui met en location son gîte),
- le client réservataire,
- le client locataire.

Deux acteurs internes peuvent être identifiés :

- le gestionnaire Catalogue-Propriétaire,
- le gestionnaire Réservation-Location.

Le diagramme de ces cas d'utilisation est donné à la figure 3.8.

Dans le cadre de cet exercice de synthèse, nous nous limiterons à donner la description textuelle des scénarios des cas d'utilisation de l'activité catalogue (cas 1.1, 1.2 et 1.3).

Cas d'utilisation 1.1 : Gestion annuelle du catalogue

Deux scénarios peuvent être considérés : la création du gîte et la modification du gîte.

Scénario 1.1.1 « Création gîte »

- *Objectif* – Permettre l'ajout d'un gîte dans le catalogue.
- *Acteurs concernés* – Gestionnaire catalogue.
- *Pré conditions* – Aucune.

- *Scénario nominal*
 - 1. Créer un nouveau propriétaire s'il n'existe pas.
 - 2. Créer un gîte.
 - 3. Ajouter le gîte créé au catalogue.
- *Scénarios alternatifs*
 - 1-a : Erreurs détectées dans la saisie du propriétaire :
 - Le système réaffiche le formulaire de saisie en indiquant les erreurs détectées.
 - Le coordonnateur corrige les erreurs.
 - Le cas d'utilisation reprend à l'action 1 du scénario nominal.
 - 2-a : Erreurs détectées dans la saisie du gîte :
 - Le système réaffiche le formulaire de saisie en indiquant les erreurs détectées.
 - Le coordonnateur corrige les erreurs.
 - Le cas d'utilisation reprend à l'action 2 du scénario nominal.

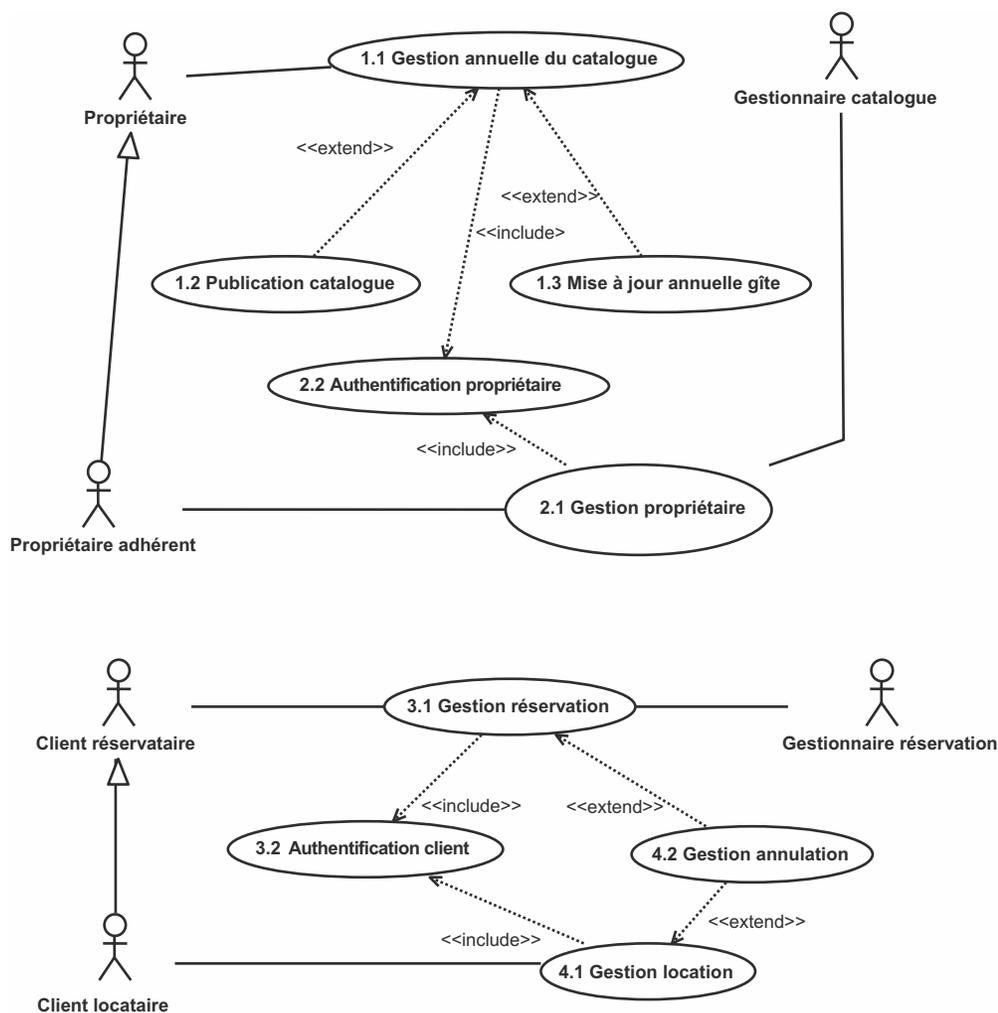


Figure 3.8 — Diagramme des cas d'utilisation de Locagite

Scénario 1.1.2 « Modification gîte »

- *Objectif* – Permettre la modification d'un gîte déjà présent dans le catalogue.
- *Acteurs concernés* – Gestionnaire catalogue.
- *Pré conditions* – Aucune.
- *Scénario nominal*
 - 1. Saisie et contrôle d'existence du gîte.
 - 2. Saisie et contrôle d'existence du propriétaire.
 - 3. Modification des données du gîte.
 - 4. Modification éventuelle des activités du gîte,
- *Scénarios alternatifs*
 - 1-a : Erreur de saisie du gîte :
 - Le système réaffiche le formulaire de saisie en indiquant l'erreur détectée.
 - Le coordonnateur corrige les erreurs.
 - Le cas d'utilisation reprend à l'action 1 du scénario nominal.
 - 2-a : Erreurs de saisie du propriétaire :
 - Le système réaffiche le formulaire de saisie en indiquant les erreurs détectées.
 - Le coordonnateur corrige les erreurs.
 - Le cas d'utilisation reprend à l'action 2 du scénario nominal.

Cas d'utilisation 1.2 : Publication du catalogue

Ce cas d'utilisation ne comporte qu'un seul scénario.

Scénario « Publication du catalogue »

- *Objectif* – Permettre l'édition du catalogue.
- *Acteurs concernés* – Gestionnaire catalogue.
- *Pré conditions* – Aucune.
- *Scénario nominal* – Pour chaque gîte :
 - 1. Rechercher les informations sur le propriétaire.
 - 2. Afficher les tarifs de location à la semaine.
 - 3. Afficher les activités disponibles.
- *Scénarios alternatifs* – Aucun.

Cas d'utilisation 1.3 : Mise à jour annuelle du gîte

Ce cas d'utilisation ne comporte qu'un seul scénario.

Scénario « Mise à jour annuelle du gîte »

- *Objectif* – Permettre la mise à jour du nombre d'étoile d'un gîte donné.
- *Acteurs concernés* – Gestionnaire catalogue.
- *Pré conditions* – Aucune.
- *Scénario nominal* :
 - 1. Saisir le code propriétaire, le code du gîte et le nombre d'étoile.
 - 2. Mettre à jour le nombre d'étoile.

- *Scénarios alternatifs* :
 - 1-a : le propriétaire ou le gîte n'existe pas :
 - Le système réaffiche le formulaire de saisie en indiquant l'erreur détectée.
 - Le gestionnaire corrige les erreurs.
 - Le cas d'utilisation reprend à l'action 1 du scénario nominal.

3.2 DIAGRAMME D'ÉTAT-TRANSITION (DET)

3.2.1 Présentation générale et concepts de base

État-transition et événement

L'état d'un objet est défini, à un instant donné, par l'ensemble des valeurs de ses propriétés. Seuls certains états caractéristiques du domaine étudié sont considérés.

Le passage d'un état à un autre état s'appelle **transition**. Un événement est un fait survenu qui déclenche une transition.

Il existe quatre types d'événements :

- **Type appel de méthode (call)** – C'est le type le plus courant que nous traiterons dans la suite de la présentation.
- **Type signal** – Exemple : clic de souris, interruption d'entrées-sorties... La modélisation de la réception ou l'émission d'un signal est traitée dans le diagramme d'activité.
- **Type changement de valeur (vrai/faux)** – C'est le cas de l'évaluation d'une expression booléenne.
- **Type écoulement du temps** – C'est un événement lié à une condition de type after (durée) ou when (date).

Formalisme et exemple

Un objet reste dans un état pendant une certaine durée. La durée d'un état correspond au temps qui s'écoule entre le début d'un état déclenché par une transition i et la fin de l'état déclenché par la transition $i+1$. Une condition, appelée « garde », peut être associée à une transition.

Le formalisme de représentation d'état-transition est donné à la figure 3.9.



Figure 3.9 — Formalisme d'état-transition

La figure 3.10 donne un premier exemple d'état-transition. Dans cet exemple, pour un employé donné d'une entreprise, nous pouvons considérer les deux états significatifs suivants : état recruté, état en activité.

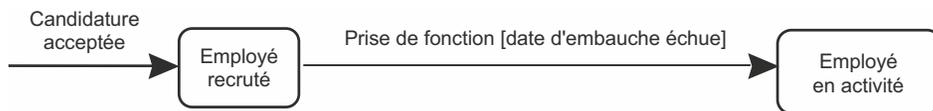


Figure 3.10 — Exemple d'état-transition

Action et activité

Une **action** est une opération instantanée qui ne peut être interrompue ; elle est associée à une transition.

Une **activité** est une opération d'une certaine durée qui peut être interrompue, elle est associée à un état d'un objet.

Formalisme et exemple

Le formalisme de représentation d'état-transition comprenant la représentation d'action et/ou activité est donné à la figure 3.11.

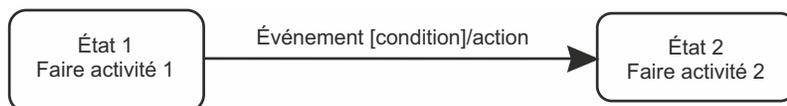


Figure 3.11 — Formalisme d'état-transition avec action et activité

La figure 3.12 montre un exemple des actions et activités d'états ainsi que la description complète d'une transition.



Figure 3.12 — Exemple d'état-transition avec action et activité

3.2.2 Représentation du diagramme d'état-transition d'un objet

L'enchaînement de tous les états caractéristiques d'un objet constitue le diagramme d'état. Un diagramme d'états débute toujours par un état initial et se termine par un ou plusieurs états finaux sauf dans le cas où le diagramme d'états représente une boucle. À un événement peut être associé un message composé d'attributs.

Formalisme et exemple

Le formalisme de représentation des états initial et final est donné à la figure 3.13.

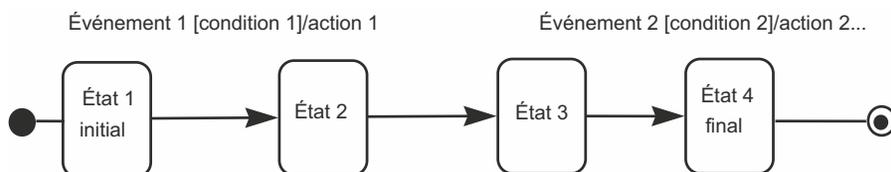


Figure 3.13 – Formalisme de représentation des états initial et final

Afin de nous rapprocher des situations réelles, nous proposons à la figure 3.14 un premier exemple tiré d'une gestion commerciale qui montre le diagramme d'état-transition de l'objet client.

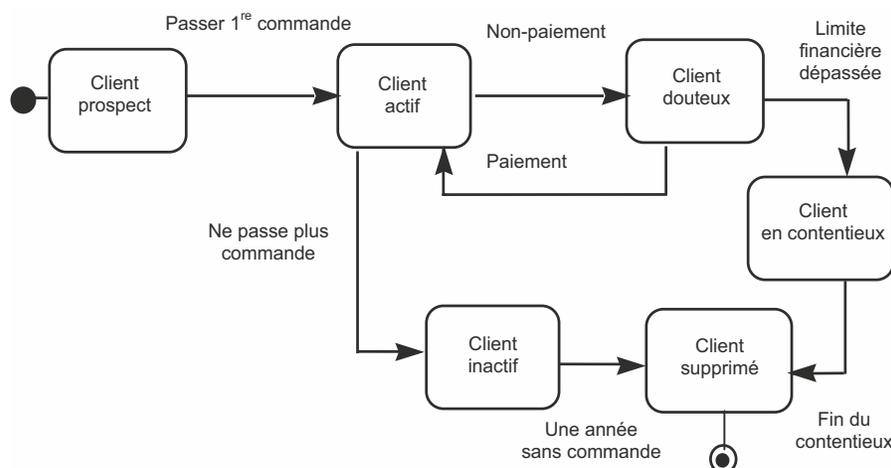


Figure 3.14 – Diagramme d'état-transition de l'objet client d'une gestion commerciale

Nous proposons comme second exemple, à la figure 3.15, le diagramme d'état-transition de l'objet « personnel » qui se caractérise par trois états :

- **En prévision d'arrivée** : si la date prévisionnelle est postérieure à la date du jour.
- **En activité** : état qui correspond à un personnel ayant une date d'arrivée renseignée.
- **Parti** : état qui correspond à un personnel ayant une date de départ renseignée.

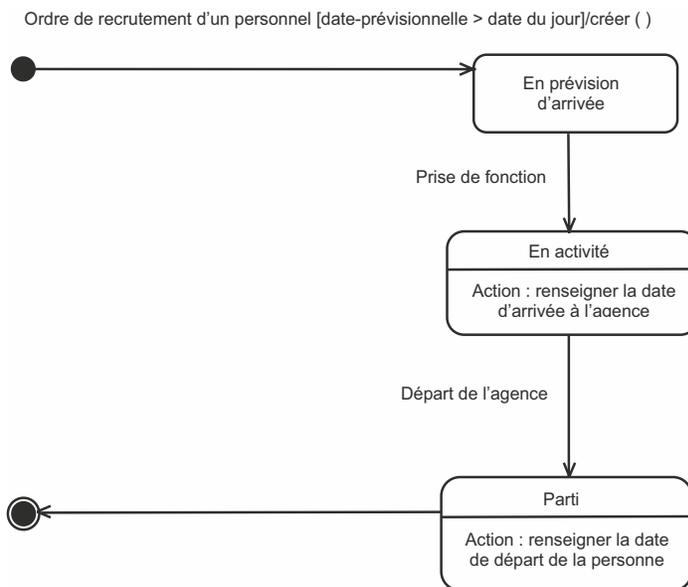


Figure 3.15 — Exemple de diagramme d'état-transition

3.2.3 Compléments sur le diagramme d'état-transition

Composition et décomposition d'état

Il est possible de décrire un diagramme d'état-transition à plusieurs niveaux. Ainsi, à un premier niveau, le diagramme comprendra des états élémentaires et des états composites. Les états composites seront ensuite décrits à un niveau élémentaire dans un autre diagramme. On peut aussi parler d'état composé et d'état composant.

Formalisme et exemple

Le formalisme de représentation d'états composites est donné à la figure 3.16.

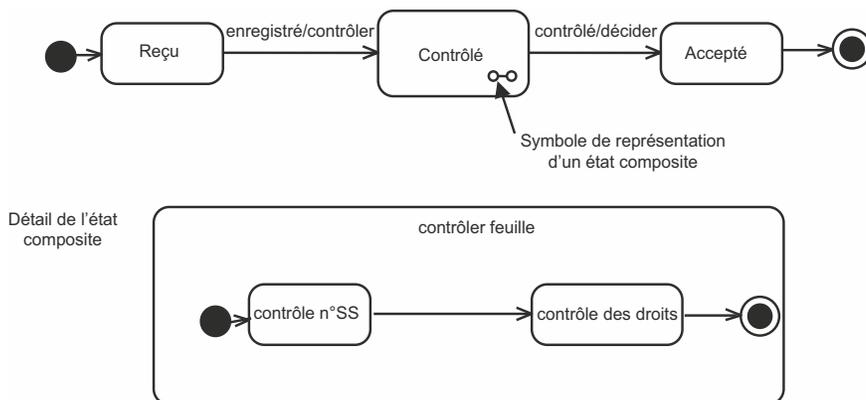


Figure 3.16 — Exemple d'état composite

Dans cet exemple, l'état contrôlé est un état composite qui fait l'objet d'une description individualisée à un second niveau que l'on appelle aussi sous-machine d'état.

Point d'entrée et de sortie

Sur une sous-machine d'état, il est possible de repérer un **point d'entrée** et un **point de sortie** particuliers.

Formalisme et exemple

Le formalisme de représentation d'une sous-machine d'état avec point d'entrée et de sortie est donné à la figure 3.17.

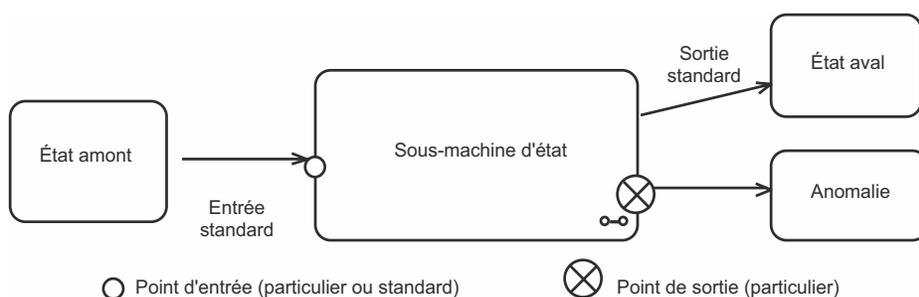


Figure 3.17 — Exemple d'une sous-machine d'état avec point d'entrée et de sortie

Point de jonction

Lorsque l'on veut relier plusieurs états vers d'autres états, un **point de jonction** permet de décomposer une transition en deux parties en indiquant si nécessaire les gardes propres à chaque segment de la transition.

À l'exécution, un seul parcours sera emprunté, c'est celui pour lequel toutes les conditions de garde seront satisfaites.

Formalisme et exemple

Le formalisme de représentation d'états-transitions avec point de jonction est donné à la figure 3.18.

Point de choix

Le **point de choix** se comporte comme un test de type : si condition faire action1 sinon faire action2.

Formalisme et exemple

Le formalisme de représentation d'états composites est donné à la figure 3.19.

● Point de jonction

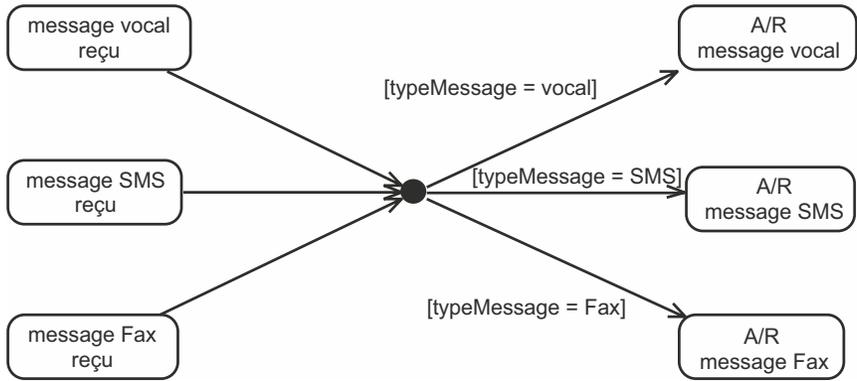


Figure 3.18 — Exemple d'états-transitions avec point de jonction

◇ Point de choix

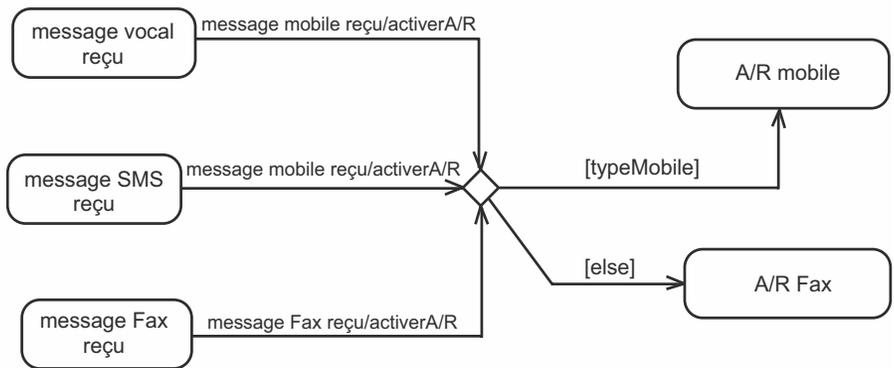


Figure 3.19 — Exemple d'états-transitions avec point de choix

État historique

La mention de l'historisation d'un état composite permet de pouvoir indiquer la réutilisation du dernier état historisé en cas de besoin.

Formalisme et exemple

Le formalisme de représentation d'états historisés est donné à la figure 3.20.

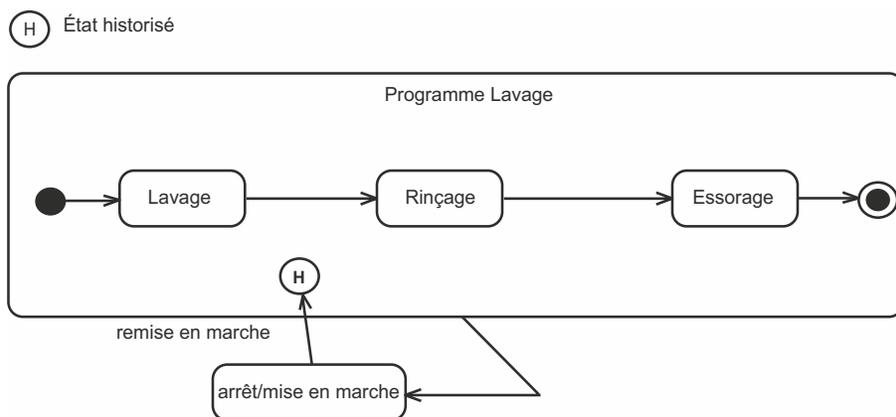


Figure 3.20 — Exemple d'états-transitions historisés

3.2.4 Exercices

Exercice 1

Énoncé

Soit à représenter le diagramme d'état-transition d'un objet personnel en suivant les événements de gestion depuis le recrutement jusqu'à la mise en retraite.

Après le recrutement, une personne est considérée en activité dès sa prise de fonction dans l'entreprise. Au cours de sa carrière, nous retiendrons seulement les événements : congé de maladie et prise de congé annuel. En fin de carrière, nous retiendrons deux situations : la démission et la retraite.

Corrigé

Nous proposons au lecteur un corrigé type à la figure 3.21. Ce corrigé ne représente qu'une solution parmi d'autres variantes possibles suivant la lecture faite de l'énoncé. Pour notre part, nous avons retenu les états caractéristiques : recruté, activité, en congé, en arrêt, parti et retraite.

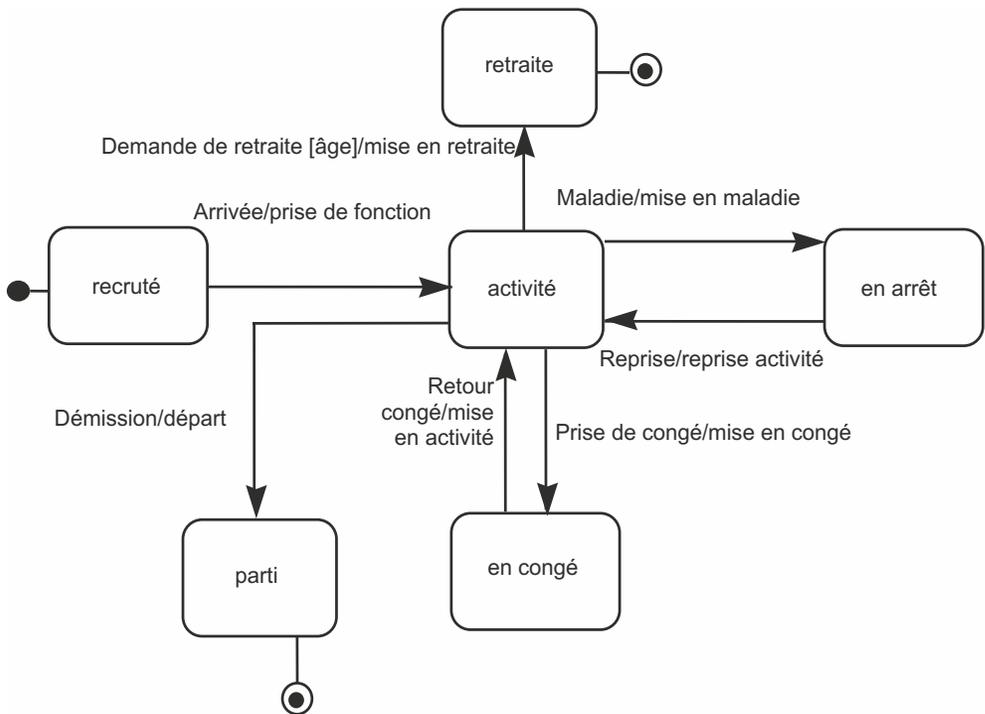


Figure 3.21 – Diagramme d'état-transition de l'exercice 1

Exercice de synthèse

En se replaçant dans l'exercice de synthèse Locagite, nous allons retenir l'objet « Gîtes gérés » comme support d'application du diagramme d'état-transition. Quatre états permettent de caractériser son comportement :

- État 1 : Gîte à louer
- État 2 : Gîte réservé
- État 3 : Gîte réservé ferme
- État 4 : Gîte réservé loué

La figure 3.22 représente le diagramme d'état-transition de cet objet.

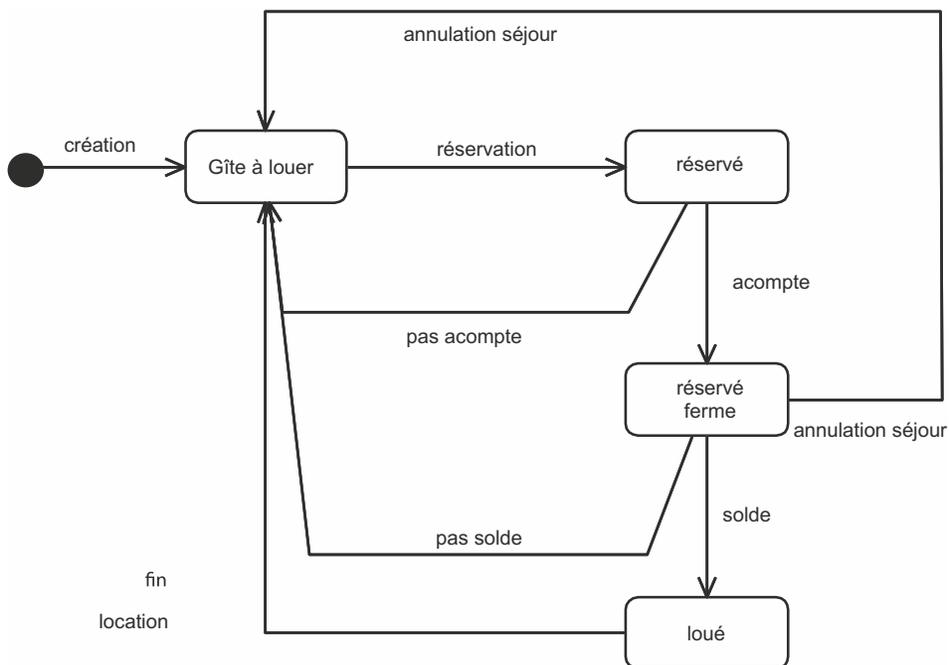


Figure 3.22 — Diagramme d'état-transition de l'objet « Gites gérés »

3.3 DIAGRAMME D'ACTIVITÉ (DAC)

3.3.1 Présentation générale et concepts de base

Le **diagramme d'activité** présente un certain nombre de points communs avec le diagramme d'état-transition puisqu'il concerne le comportement interne des opérations ou des cas d'utilisation. Cependant le comportement visé ici s'applique aux flots de contrôle et aux flots de données propres à un ensemble d'activités et non plus relativement à une seule classe.

Les concepts communs ou très proches entre le diagramme d'activité et le diagramme d'état-transition sont :

- transition,
- ● nœud initial (état initial),
- ⊙ nœud final (état final),
- ⊗ nœud de fin flot (état de sortie),
- ◇ nœud de décision (choix).

Le formalisme reste identique pour ces nœuds de contrôle.

Les concepts spécifiques au diagramme d'activité sont :

- nœud de bifurcation,
- nœud de jonction,
- nœud de fusion,
- pin d'entrée et de sortie,
- flot d'objet,
- partition.

Nous avons par ailleurs réservé un traitement particulier pour les concepts action et activité. En effet, étant donné que ces concepts sont au cœur du diagramme d'activité, nous avons préféré les traiter de manière détaillée à ce niveau alors qu'ils sont déjà évoqués dans le diagramme d'état-transition.

Action

Une **action** correspond à un traitement qui modifie l'état du système. Cette action peut être appréhendée soit à un niveau élémentaire proche d'une instruction en termes de programmation soit à un niveau plus global correspondant à une ou plusieurs opérations.

Formalisme et exemple

Une action est représentée par un rectangle dont les coins sont arrondis comme pour les états du diagramme d'état-transition (fig. 3.23).



Figure 3.23 — Formalisme et exemple d'une action

Transition et flot de contrôle

Dès qu'une action est achevée, une **transition** automatique est déclenchée vers l'action suivante. Il n'y a donc pas d'événement associé à la transition.

L'enchaînement des actions constitue le **flot de contrôle**.

Formalisme et exemple

Le formalisme de représentation d'une transition est donné à la figure 3.24.

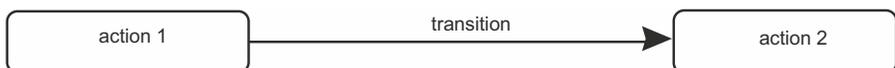


Figure 3.24 — Formalisme de base du diagramme d'activité : actions et transition

Activité

Une **activité** représente le comportement d'une partie du système en termes d'actions et de transitions. Une activité est composée de trois types de nœuds :

- nœud d'exécution (action, transition),
- nœud de contrôle (nœud initial, nœud final, flux de sortie, nœud de bifurcation, nœud de jonction, nœud de fusion-test, nœud de test-décision, pin d'entrée et de sortie),
- nœud d'objet.

Une activité peut recevoir des paramètres en entrée et en produire en sortie.

Formalisme et exemple

Nous donnons une première représentation simple à la figure 3.25.

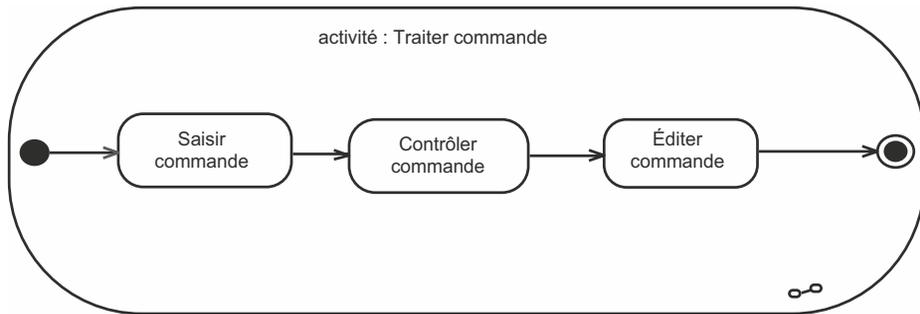


Figure 3.25 — Exemple de représentation d'une activité

Nœud de bifurcation (fourche)

Un **nœud de bifurcation** (fourche) permet à partir d'un flot unique entrant de créer plusieurs flots concurrents en sortie de la barre de synchronisation.

Formalisme et exemple

Le formalisme de représentation de nœud de bifurcation ainsi qu'un premier exemple sont donnés à la figure 3.26. Un second exemple avec nœud de bifurcation est donné à la figure 3.27.

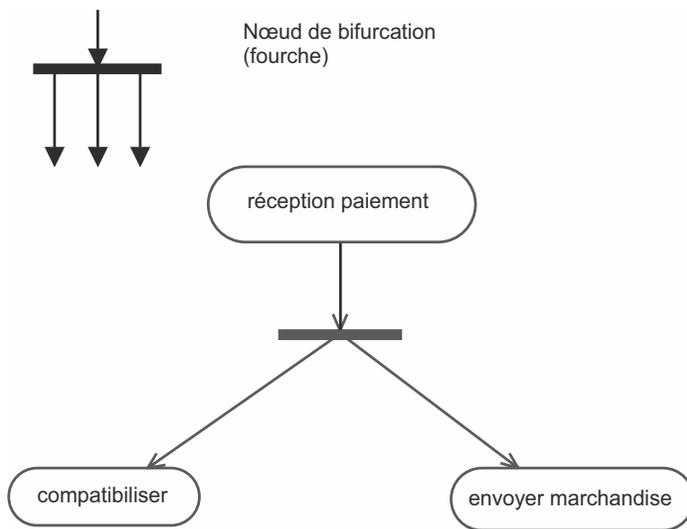


Figure 3.26 – Exemple 1 d'activités avec nœud de bifurcation

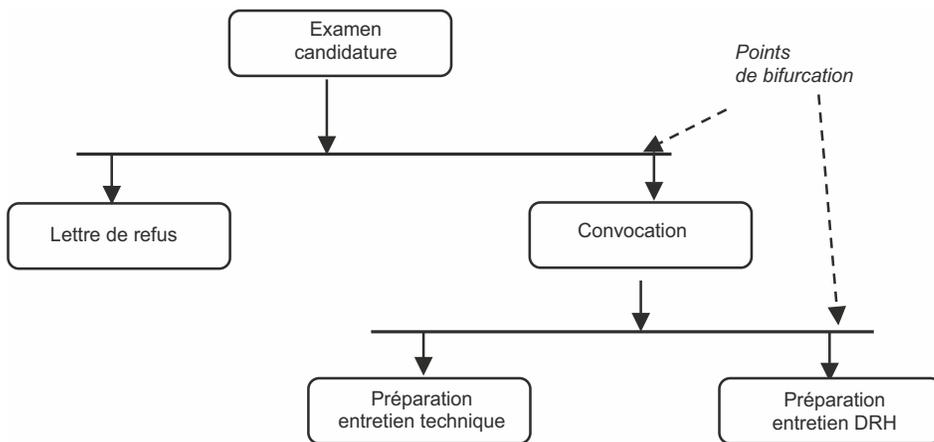


Figure 3.27 – Exemple 2 de diagramme d'activité avec bifurcation de flots de contrôle

Nœud de jonction (synchronisation)

Un **nœud de jonction** (synchronisation) permet, à partir de plusieurs flots concurrents en entrée de la synchronisation, de produire un flot unique sortant. Le nœud de jonction est le symétrique du nœud de bifurcation.

Formalisme et exemple

Le formalisme de représentation d'un nœud de jonction est donné à la figure 3.28.

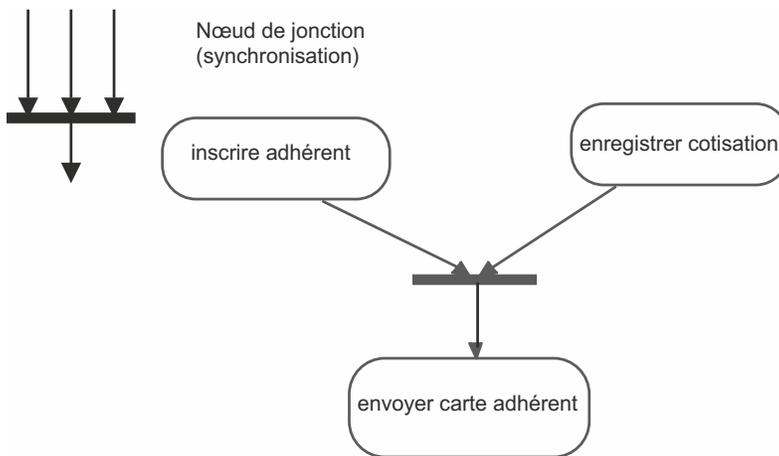


Figure 3.28 – Exemple d'activités avec nœud de jonction

Nœud de test-décision

Un **nœud de test-décision** permet de faire un choix entre plusieurs flots sortants en fonction des conditions de garde de chaque flot. Un nœud de test-décision n'a qu'un seul flot en entrée. On peut aussi utiliser seulement deux flots de sortie : le premier correspondant à la condition vérifiée et l'autre traitant le cas sinon.

Formalisme et exemple

Le formalisme de représentation d'un nœud de test-décision ainsi qu'un premier exemple sont donnés à la figure 3.29. Un second exemple avec nœud de test-décision est donné à la figure 3.30.

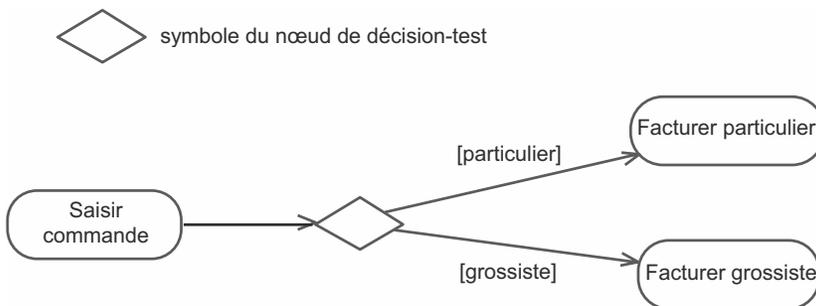


Figure 3.29 – Formalisme et exemple 1 d'activités avec nœud de test-décision

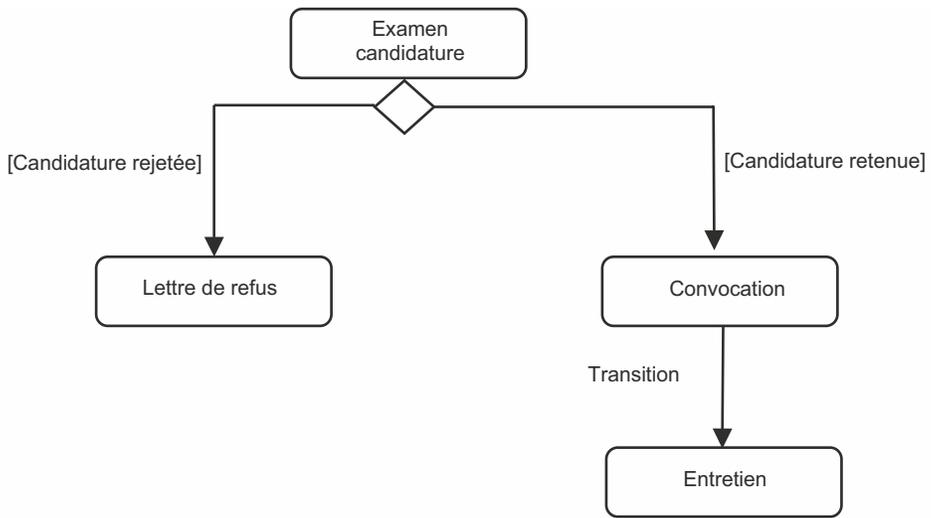


Figure 3.30 – Exemple 2 de diagramme d'activités avec un nœud de test-décision

Nœud de fusion-test

Un **nœud de fusion-test** permet d'avoir plusieurs flots entrants possibles et un seul flot sortant. Le flot sortant est donc exécuté dès qu'un des flots entrants est activé.

Formalisme et exemple

Le formalisme de représentation d'un nœud de fusion-test ainsi qu'un exemple sont donnés à la figure 3.31.

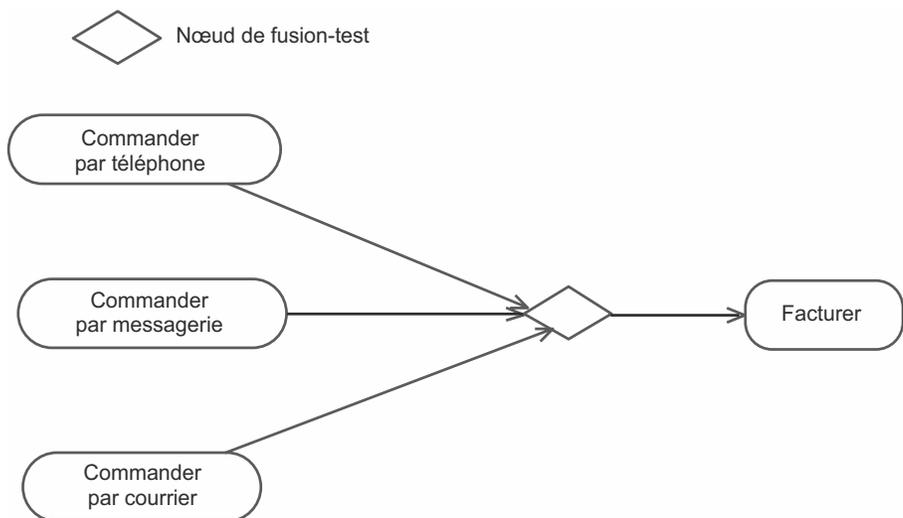


Figure 3.31 – Formalisme et exemple de diagramme d'activités avec un nœud de fusion-test

Pin d'entrée et de sortie

Un **pin d'entrée** ou **de sortie** représente un paramètre que l'on peut spécifier en entrée ou en sortie d'une action. Un nom de donnée et un type de donnée peuvent être associés au pin. Un paramètre peut être de type objet.

Formalisme et exemple

Chaque paramètre se représente dans un petit rectangle. Le nom du paramètre ainsi que son type sont aussi à indiquer. Le formalisme de représentation de pin d'entrée ou de sortie ainsi qu'un exemple sont donnés à la figure 3.32.

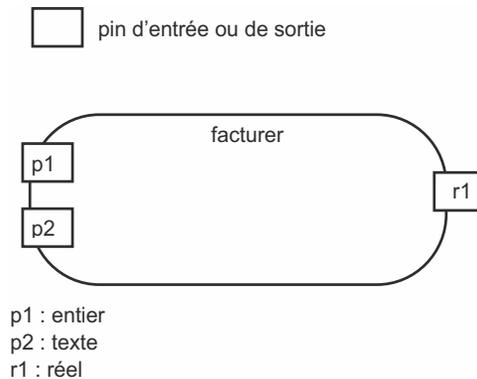


Figure 3.32 — Formalisme et exemple d'activité avec pin d'entrée et de sortie

Flot de données et nœud d'objet

Un **nœud d'objet** permet de représenter le **flot de données** véhiculé entre les actions. Les objets peuvent se représenter de deux manières différentes : soit en utilisant le pin d'objet soit en représentant explicitement un objet.

Formalisme et exemple

Le formalisme de représentation de flot de données et nœud d'objet est donné directement au travers d'un exemple (fig. 3.33).

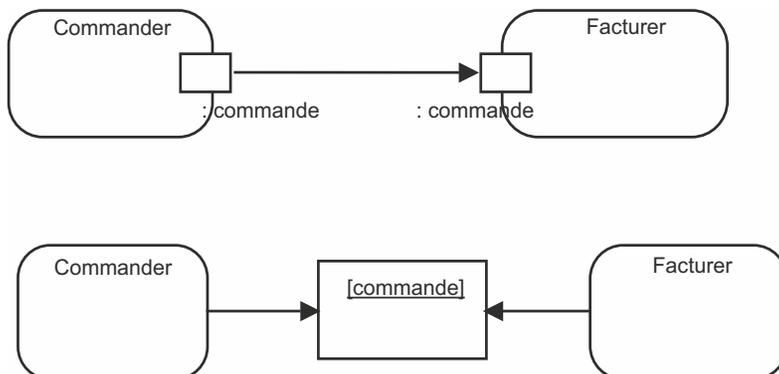


Figure 3.33 — Exemple de flot de données et de nœud d'objets

Partition

UML permet aussi d'organiser la présentation du diagramme d'activité en couloir d'activités. Chaque couloir correspond à un domaine de responsabilité d'un certain nombre d'actions.

Les flots d'objets sont aussi représentés dans le diagramme. L'ordre relatif des couloirs de responsabilité n'est pas significatif.

3.3.2 Représentation du diagramme d'activité

Un exemple général de diagramme d'activité est donné à la figure 3.34.

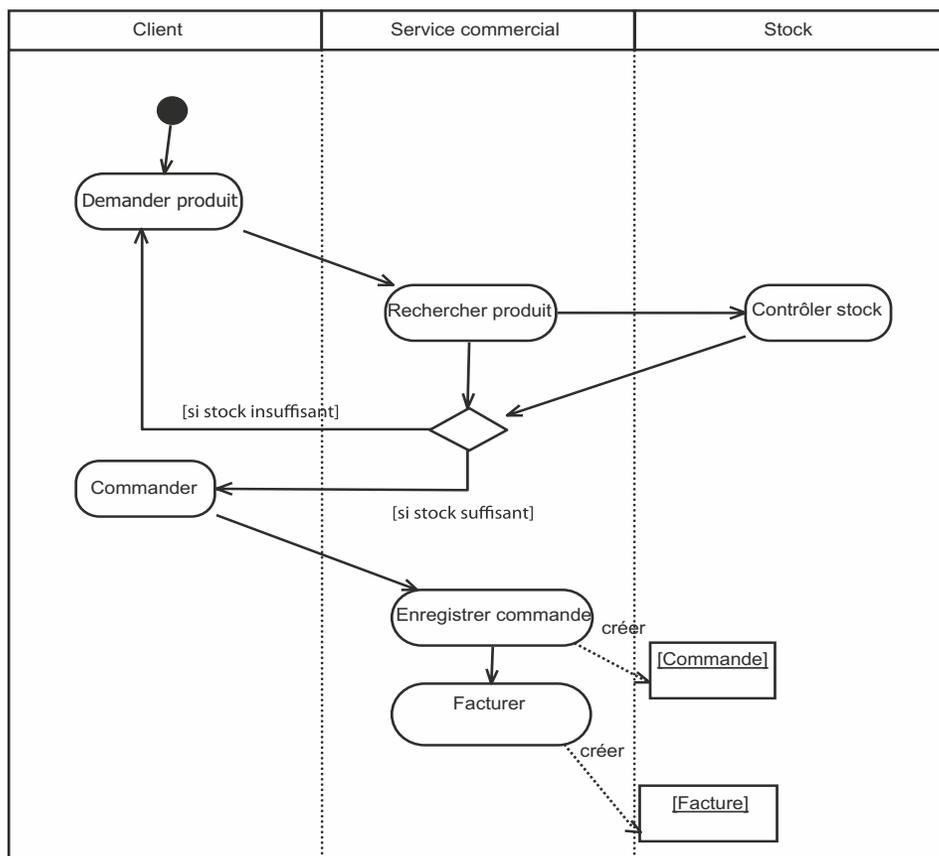


Figure 3.34 – Exemple de diagramme d'activité avec couloir d'activité

Représentation d'actions de communication

Dans un diagramme d'activité, comme dans un diagramme de temps, des interactions de communication liées à certains types d'événement peuvent se représenter. Les types d'événement concernés sont :

- signal,
- écoulement du temps.

Formalisme et exemple

Le formalisme de représentation ainsi qu'un exemple d'actions de communication sont donnés à la figure 3.35.

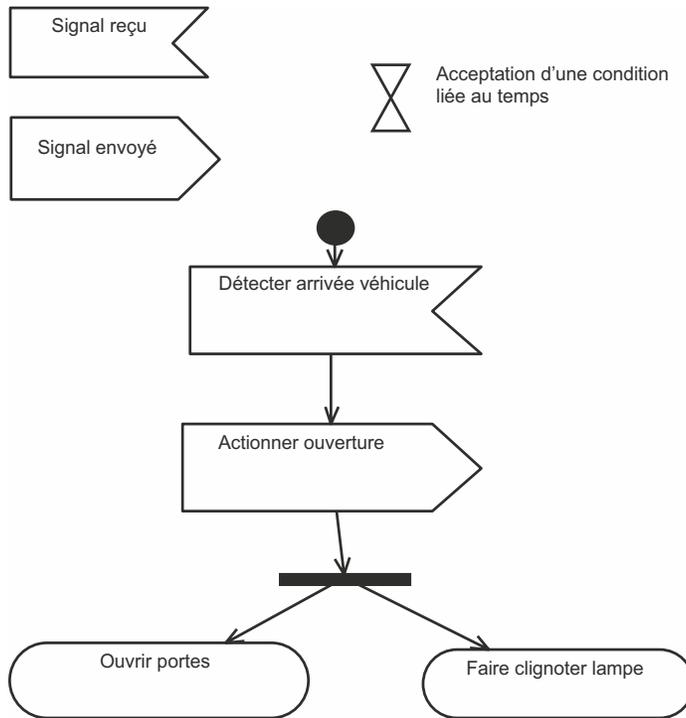


Figure 3.35 – Formalisme et exemple de diagramme d'activité avec actions de communication

3.3.3 Exercices

Exercice 1

En reprenant l'exercice relatif à la gestion de la bibliothèque traité dans les cas d'utilisation nous pouvons élaborer le diagramme d'activité correspondant.

Deux acteurs ont été identifiés :

- Bibliothécaire chargé de l'approvisionnement des ouvrages, de la gestion du catalogue et de l'enregistrement des emprunts et retours d'ouvrages ;
- Gestionnaire, chargé de l'inscription des adhérents et de la relance des adhérents ayant dépassé le délai de restitution des ouvrages.

La représentation du diagramme d'activité est donnée à la figure 3.36.

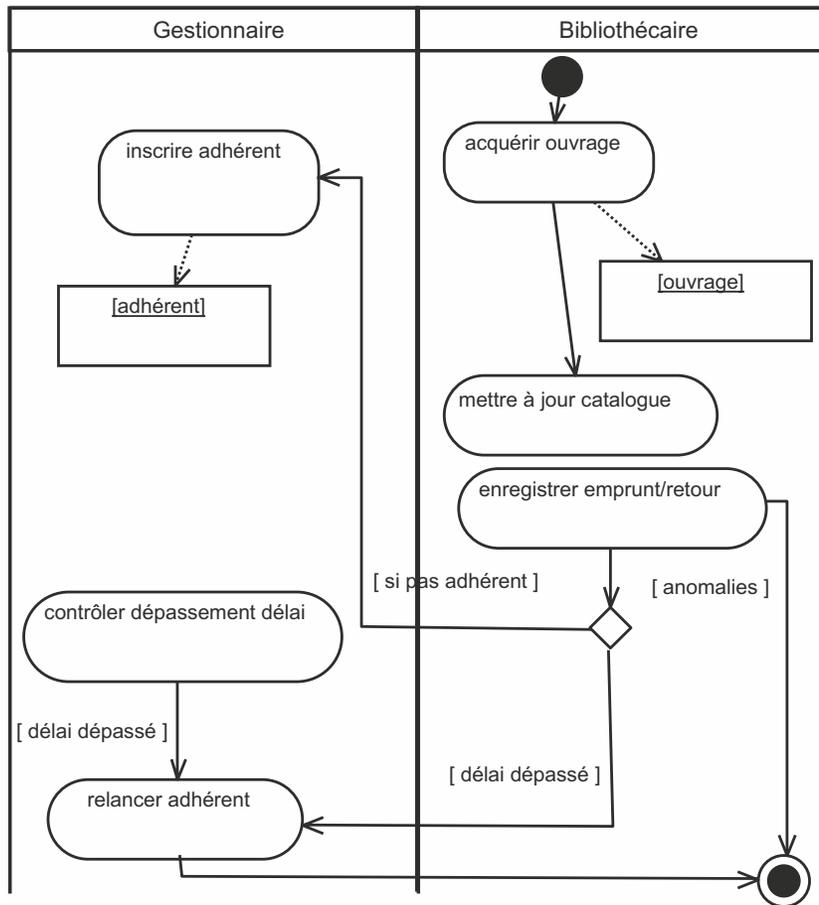


Figure 3.36 – Diagramme d'activité de l'exercice 1

Exercice de synthèse

La figure 3.37 représente le diagramme d'activité de Locagite. Ce diagramme nous permet de montrer, par couloir de responsabilité des acteurs internes à Locagite, les états-actions exécutés.

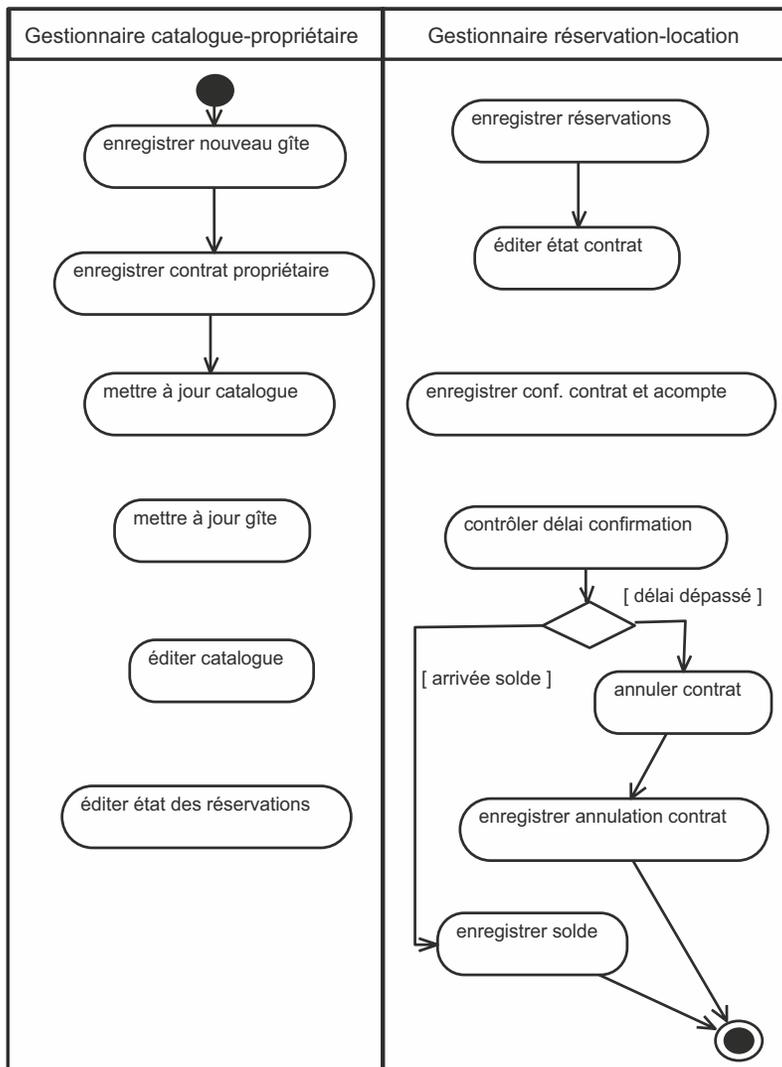


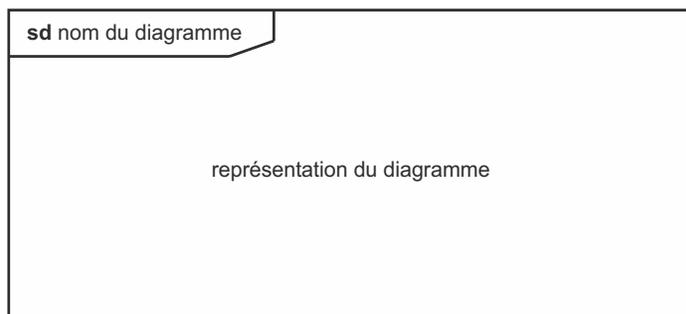
Figure 3.37 – Diagramme d'activité du cas Locagite

3.4 DIAGRAMME DE SÉQUENCE (DSE)

3.4.1 Présentation générale et concepts de base

L'objectif du **diagramme de séquence** est de représenter les interactions entre objets en indiquant la chronologie des échanges. Cette représentation peut se réaliser par cas d'utilisation en considérant les différents scénarios associés.

Un diagramme de séquence se représente globalement dans un grand rectangle avec indication du nom du diagramme en haut à gauche comme indiqué à la figure 3.38.



sd : abréviation de « sequence diagramm »

Figure 3.38 – Formalisme général du cadre d'un diagramme de séquence

Ligne de vie

Une **ligne de vie** représente l'ensemble des opérations exécutées par un objet. Un message reçu par un objet déclenche l'exécution d'une opération. Le retour d'information peut être implicite (cas général) ou explicite à l'aide d'un message de retour.

Message synchrone et asynchrone

Dans un diagramme de séquence, deux types de messages peuvent être distingués :

- **Message synchrone** – Dans ce cas l'émetteur reste en attente de la réponse à son message avant de poursuivre ses actions. La flèche avec extrémité pleine symbolise ce type de message. Le message retour peut ne pas être représenté car il est inclus dans la fin d'exécution de l'opération de l'objet destinataire du message. Voir le message 1 de la figure 3.39.
- **Message asynchrone** – Dans ce cas, l'émetteur n'attend pas la réponse à son message, il poursuit l'exécution de ses opérations. C'est une flèche avec une extrémité non pleine qui symbolise ce type de message. Voir le message 2 de la figure 3.39.

Formalisme et exemple

Le formalisme est donné dans l'exemple type présenté à la figure 3.39.

3.4.2 Opérations particulières

Création et destruction d'objet

Si un objet est créé par une opération, celui-ci n'apparaît qu'au moment où il est créé. Si l'objet est détruit par une opération, la destruction se représente par « X ». Un exemple type est donné à la figure 3.40.

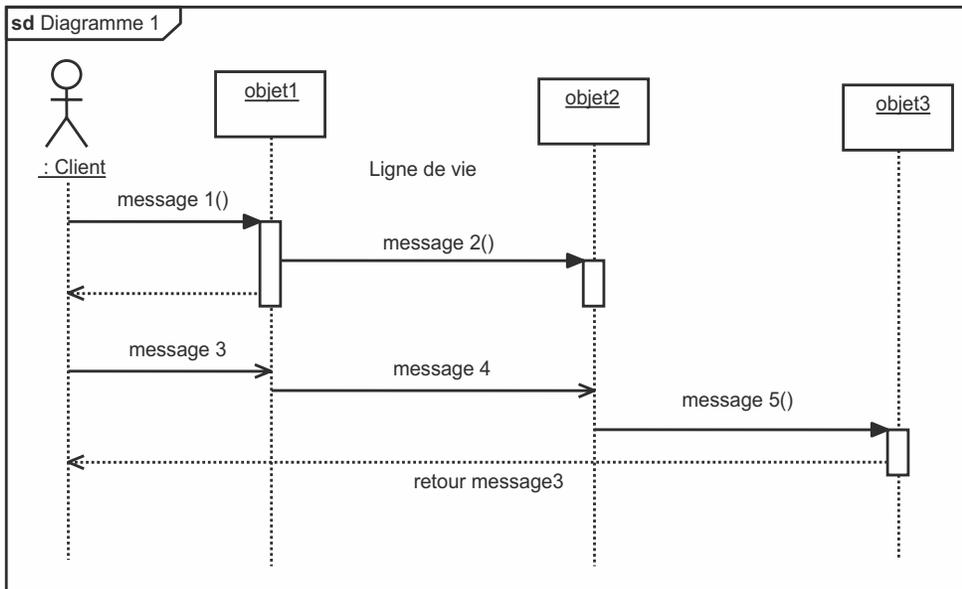


Figure 3.39 – Formalisme du diagramme de séquence

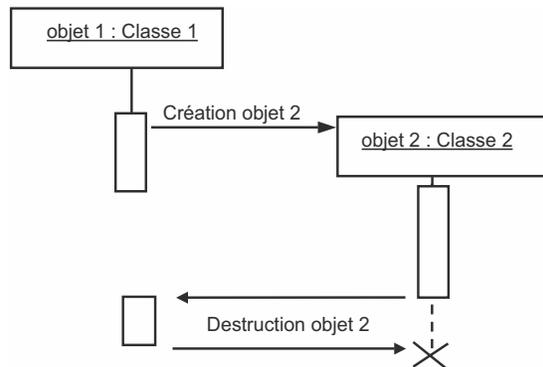


Figure 3.40 – Exemple type de création et de destruction d'objet

Il est aussi possible dans certains outils de modélisation d'indiquer plus simplement la création d'une nouvelle instance d'objet en utilisant le mot-clé « create » (voir exemple dans les études de cas présentées à la fin de cet ouvrage).

Contrainte temporelle

Des contraintes de chronologie entre les messages peuvent être spécifiées. De plus lorsque l'émission d'un message requiert une certaine durée, il se représente sous la forme d'un trait oblique. Un exemple général de **contrainte temporelle** est donné à la figure 3.41.

Lorsque le diagramme de séquence est utilisé pour représenter un sous-ensemble du logiciel à réaliser, il est possible d'indiquer le pseudo-code exécuté par un objet pendant le déroulement d'une opération.

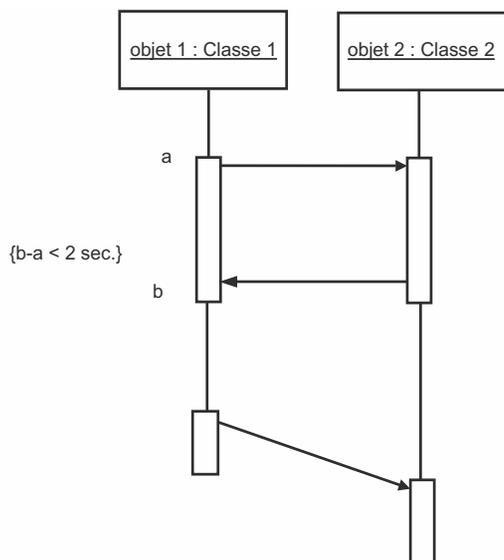


Figure 3.41 — Exemple type de représentation de contrainte temporelle

3.4.3 Fragment d'interaction

Types de fragments d'interaction

Dans un diagramme de séquence, il est possible de distinguer des sous-ensembles d'interactions qui constituent des fragments.

Un **fragment d'interaction** se représente globalement comme un diagramme de séquence dans un rectangle avec indication dans le coin à gauche du nom du fragment.

Un port d'entrée et un port de sortie peuvent être indiqués pour connaître la manière dont ce fragment peut être relié au reste du diagramme comme le montre la figure 3.42. Dans le cas où aucun port n'est indiqué c'est l'ensemble du fragment qui est appelé pour exécution.

Formalisme et exemple

Dans l'exemple proposé (fig. 3.42), le fragment « ContrôlerProduit » est représenté avec un port d'entrée et un port de sortie.

Un fragment d'interaction dit combiné correspond à un ensemble d'interaction auquel on applique un opérateur. Un fragment combiné se représente globalement comme un diagramme de séquence avec indication dans le coin à gauche du nom de l'opérateur.

Treize opérateurs ont été définis dans UML : alt, opt, loop, par, strict/weak, break, ignore/consider, critical, negative, assertion et ref.

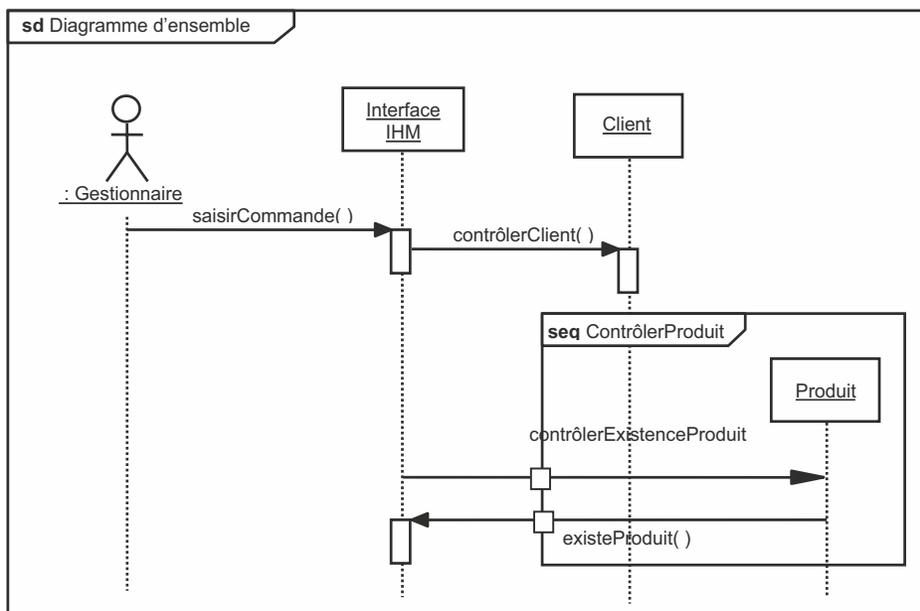


Figure 3.42 — Exemple de fragment d'interaction avec port d'entrée et de sortie

Opérateur *alt*

L'opérateur **alt** correspond à une instruction de test avec une ou plusieurs alternatives possibles. Il est aussi permis d'utiliser les clauses de type sinon.

Formalisme et exemple

L'opérateur **alt** se représente dans un fragment possédant au moins deux parties séparées par des pointillés. L'exemple donné (fig. 3.43) montre l'équivalent d'un test à deux conditions explicites (sans clause sinon).

Opérateur *opt*

L'opérateur **opt** (optional) correspond à une instruction de test sans alternative (sinon).

Formalisme et exemple

L'opérateur **opt** se représente dans un fragment possédant une seule partie (fig. 3.44).

Opérateur *loop*

L'opérateur **loop** correspond à une instruction de boucle qui permet d'exécuter une séquence d'interaction tant qu'une condition est satisfaite.

Il est possible aussi d'utiliser une condition portant sur un nombre minimum et maximum d'exécution de la boucle en écrivant : **loop** min, max. Dans ce cas, la boucle s'exécutera au minimum min fois et au maximum max fois. Il est aussi possible de combiner l'option min/max avec la condition associée à la boucle.

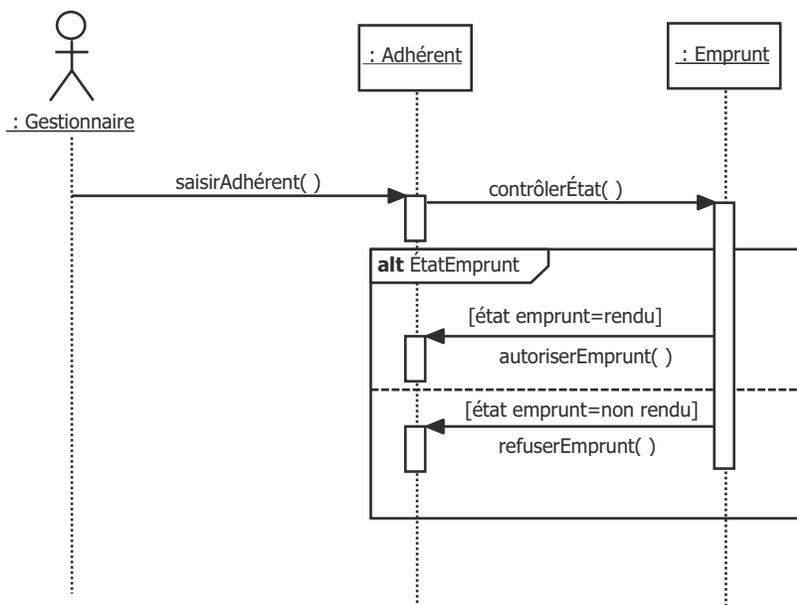


Figure 3.43 — Exemple de fragment d’interaction avec l’opérateur alt

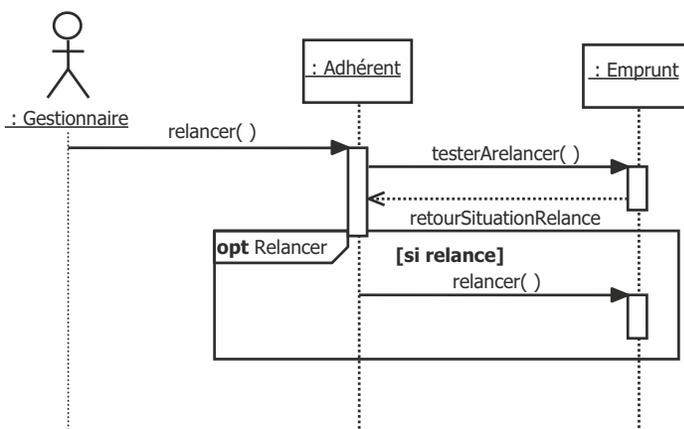


Figure 3.44 — Exemple de fragment d’interaction avec l’opérateur opt

Formalisme et exemple

L’opérateur loop se représente dans un fragment possédant une seule partie et englobant toutes les interactions faisant partie de la boucle. Un exemple est donné à la figure 3.45.

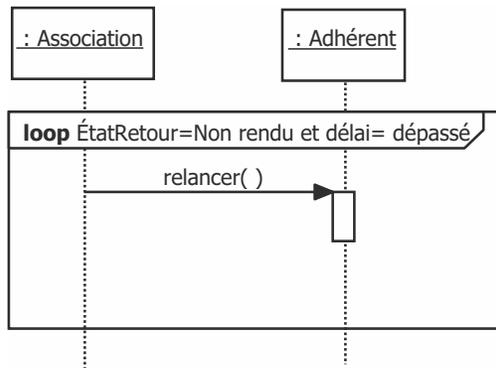


Figure 3.45 — Exemple de fragment d'interaction avec l'opérateur loop

Opérateur par

L'opérateur **par** (parallel) permet de représenter deux séries d'interactions qui se déroulent en parallèle.

Formalisme et exemple

L'opérateur **par** se représente dans un fragment possédant deux parties séparées par une ligne en pointillé. C'est un opérateur qui est à notre avis plutôt utilisé dans l'informatique temps réel et c'est pour cela que nous ne donnerons qu'un exemple type (fig. 3.46).

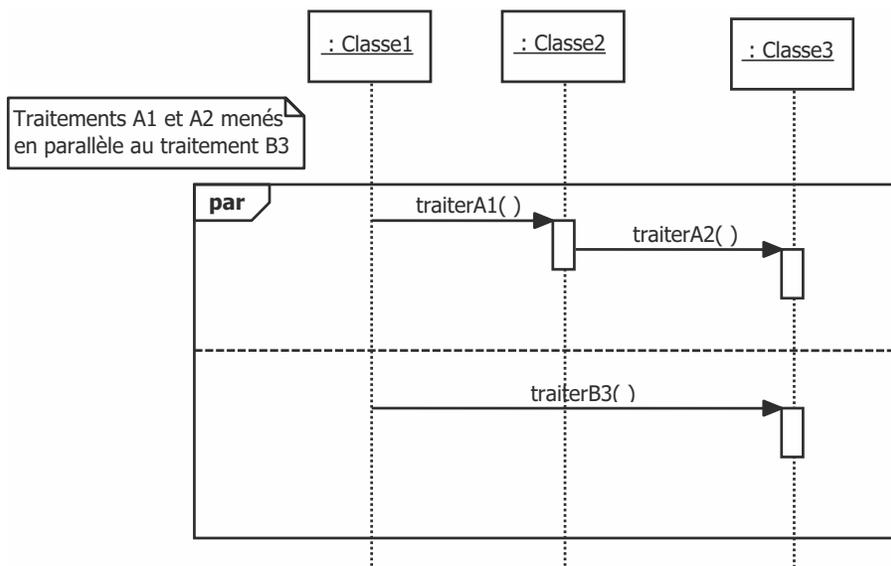


Figure 3.46 — Exemple de fragment d'interaction avec l'opérateur par

Opérateurs strict et weak sequencing

Les opérateurs **strict** et **weak** permettent de représenter une série d'interactions dont certaines s'opèrent sur des objets indépendants :

- L'opérateur strict est utilisé quand l'ordre d'exécution des opérations doit être strictement respecté.
- L'opérateur weak est utilisé quand l'ordre d'exécution des opérations n'a pas d'importance.

Formalisme et exemple

L'exemple présenté figure 3.47 montre que les opérations A1, A2, B1, B2 et A3 doivent être exécutées dans cet ordre puisqu'elles font partie du fragment d'interaction comportant l'opérateur strict.

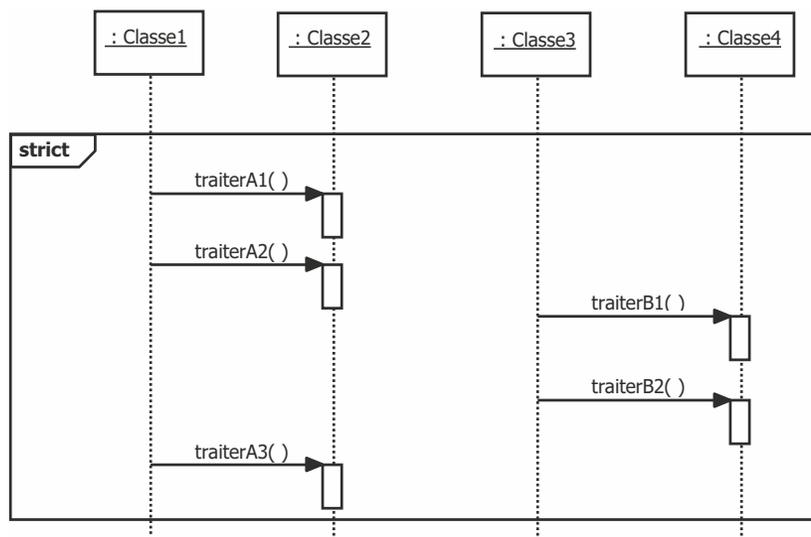


Figure 3.47 – Exemple de fragment d'interaction avec l'opérateur strict

Opérateur break

L'opérateur **break** permet de représenter une situation exceptionnelle correspondant à un scénario de rupture par rapport au scénario général. Le scénario de rupture s'exécute si la condition de garde est satisfaite.

Formalisme et exemple

L'exemple présenté figure 3.48 montre que les opérations annulerOp1(), annulerOp2() et afficherAide() ne seront exécutées que si la touche F1 est activée sinon le fragment est ignoré et la séquence de traitement passe directement de l'opération Op2() à l'opération Op3().

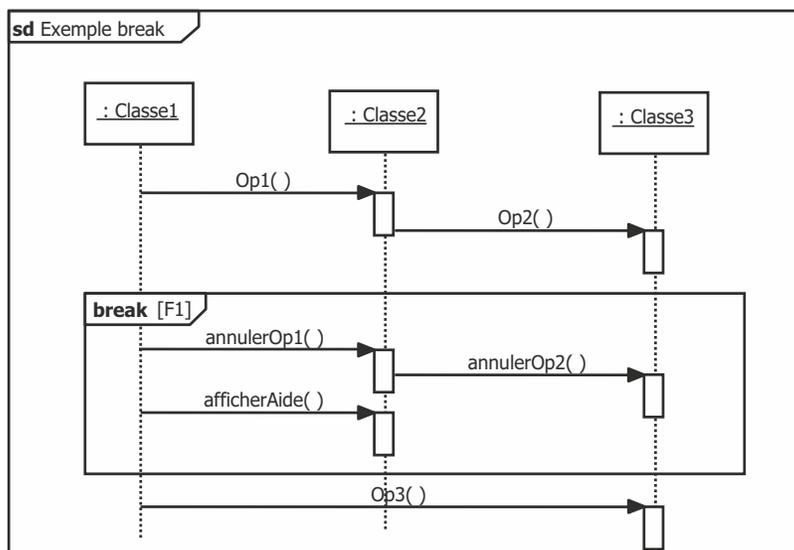


Figure 3.48 – Exemple de fragment d'interaction avec l'opérateur break

Opérateurs ignore et consider

Les opérateurs **ignore** et **consider** sont utilisés pour des fragments d'interactions dans lesquels on veut montrer que certains messages peuvent être soit absents sans avoir d'incidence sur le déroulement des interactions (ignore), soit obligatoirement présents (consider).

Formalisme et exemple

L'exemple présenté figure 3.49 montre que :

- dans le fragment consider, les messages Op1, Op2 et Op5 doivent être obligatoirement présents lors de l'exécution du fragment sinon le fragment n'est pas exécuté,
- dans le fragment ignore, les messages Op2 et Op3 peuvent être absents lors de l'exécution du fragment.

Opérateur critical

L'opérateur **critical** permet d'indiquer qu'une séquence d'interactions ne peut être interrompue compte tenu du caractère critique des opérations traitées. On considère que le traitement des interactions comprises dans la séquence critique est atomique.

Formalisme et exemple

L'exemple présenté figure 3.50 montre que les opérations Op1(), Op2() et Op3() du fragment critical doivent s'exécuter sans interruption.

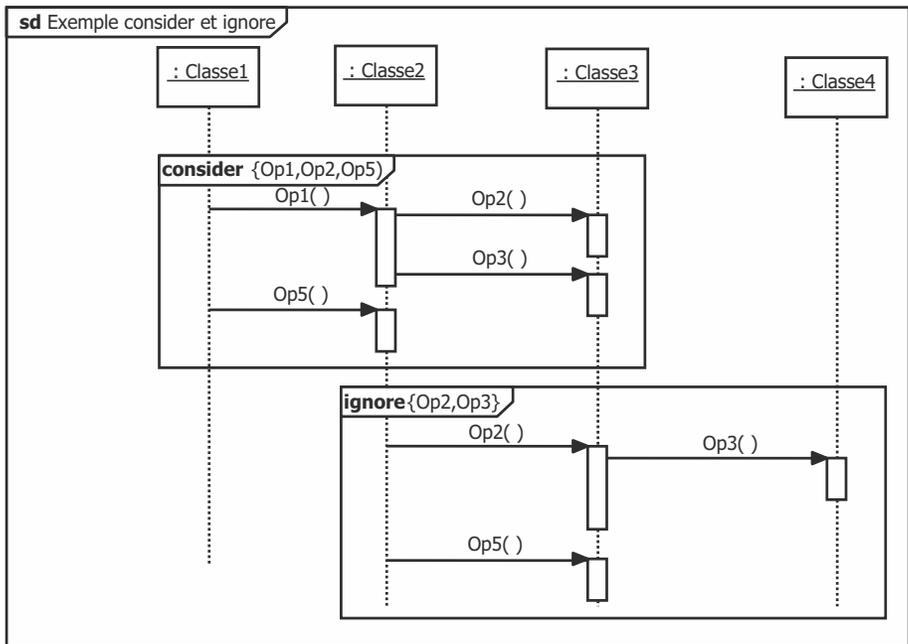


Figure 3.49 — Exemple de fragment d’interaction avec les opérateurs ignore et consider

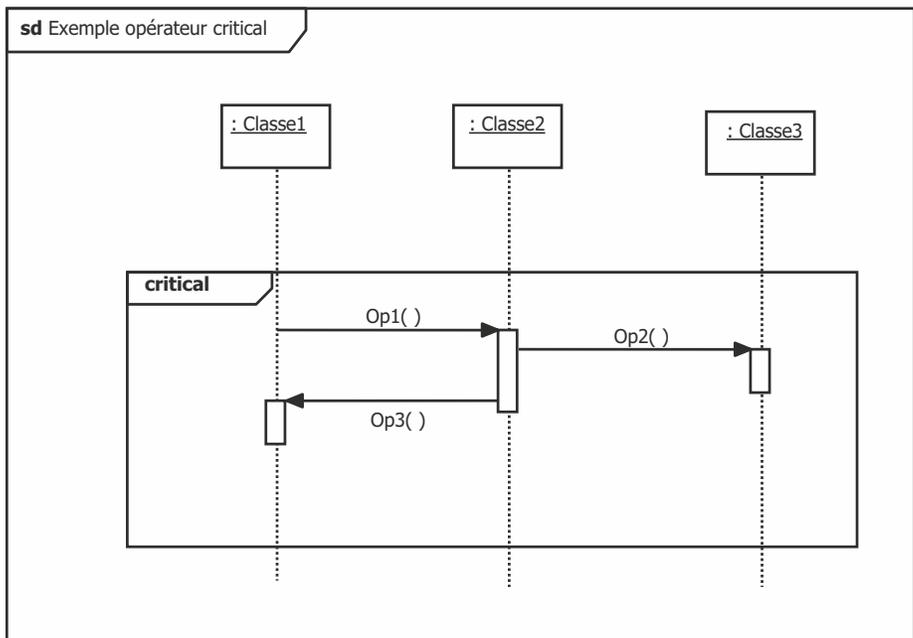


Figure 3.50 — Exemple de fragment d’interaction avec l’opérateur critical

Opérateur *negative*

L'opérateur **neg** (negative) permet d'indiquer qu'une séquence d'interactions est invalide.

Formalisme et exemple

L'exemple présenté figure 3.51 montre que les opérations `Op1()` et `Op2()` du fragment `neg` sont invalides. Une erreur sera déclenchée dans ce cas à l'exécution du fragment.

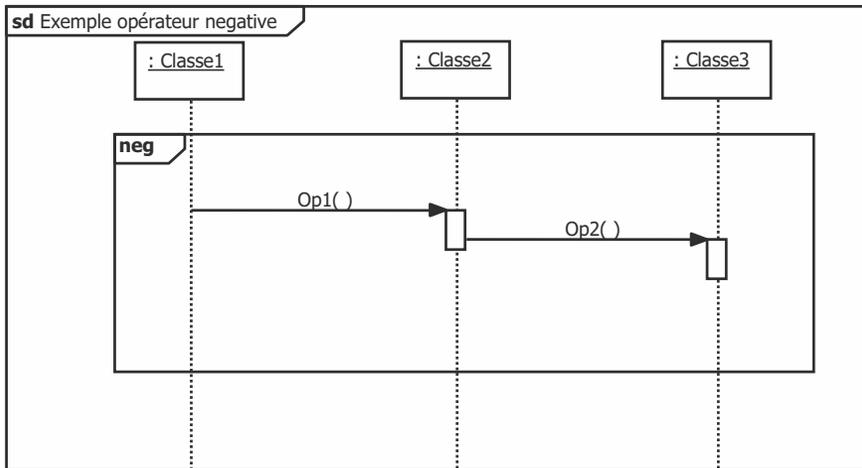


Figure 3.51 — Exemple de fragment d'interaction avec l'opérateur `neg`

Opérateur *assertion*

L'opérateur **assert** (assertion) permet d'indiquer qu'une séquence d'interactions est l'unique séquence possible en considérant les messages échangés dans le fragment. Toute autre configuration de message est invalide.

Formalisme et exemple

L'exemple présenté figure 3.52 montre que le fragment `assert` ne s'exécutera que si l'unique séquence de traitement `Op1()`, `Op2()` et `Op3()` se réalise en respectant l'ensemble des caractéristiques de ces opérations (paramètre d'entrée, type de résultat...). Toute autre situation sera considérée invalide.

Opérateur *ref*

L'opérateur **ref** permet d'appeler une séquence d'interactions décrite par ailleurs constituant ainsi une sorte de sous-diagramme de séquence.

Formalisme et exemple

L'exemple présenté figure 3.53 montre que l'on fait appel à un fragment « Contrôle des droits » qui est décrit par ailleurs.

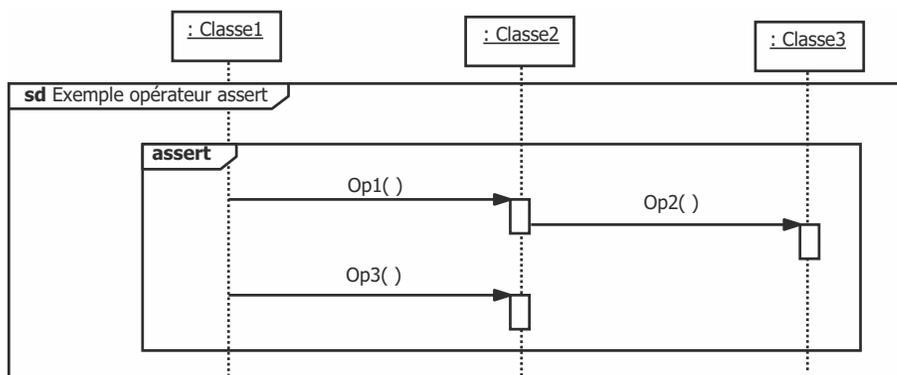


Figure 3.52 — Exemple de fragment d’interaction avec l’opérateur assert

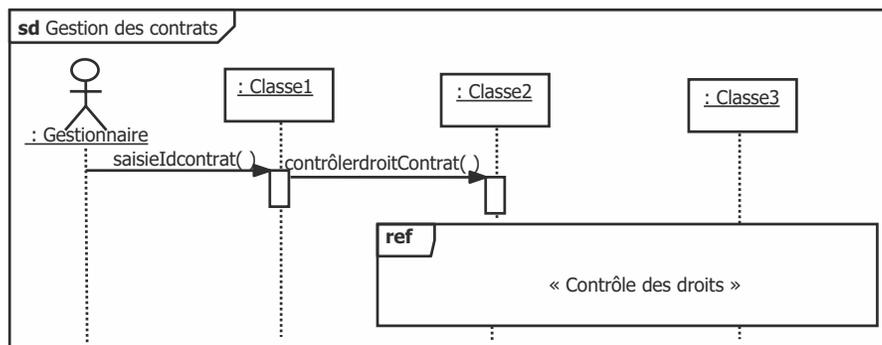


Figure 3.53 — Exemple de fragment d’interaction avec l’opérateur ref

3.4.4 Autre utilisation du diagramme de séquence

Le diagramme de séquence peut être aussi utilisé pour documenter un cas d’utilisation. Les interactions entre objets représentent, dans ce cas, des flux d’informations échangés et non pas de véritables messages entre les opérations des objets. Un exemple de cette utilisation du diagramme de séquence est donné à la figure 3.54.

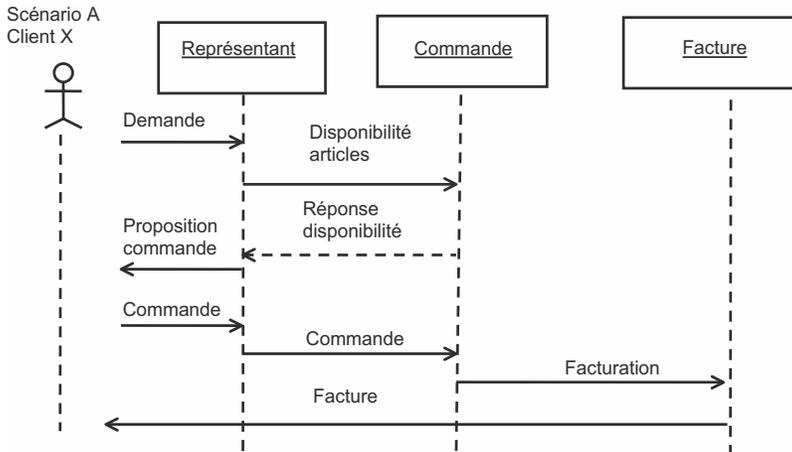


Figure 3.54 — Exemple de diagramme de séquence associé à un cas d'utilisation

3.4.5 Exercices

Exercice 1

En se référant au sujet de l'exercice 3 du diagramme de classe, nous donnons à titre d'exemple, à la figure 3.55 le diagramme de séquence. Ce scénario correspond à la création d'une agence intégrant la création d'une personne.

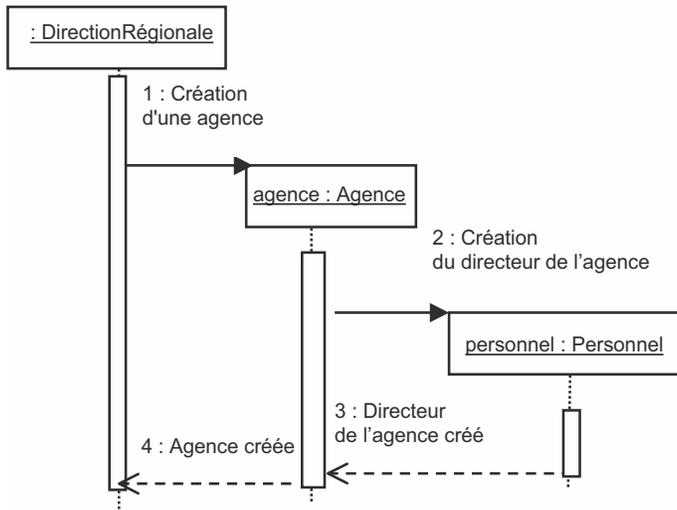


Figure 3.55 — Exemple de diagramme de séquence de l'exercice 1

Exercice de synthèse

Nous reprenons ici les cas d'utilisation décrits dans l'exercice de synthèse du DCU. La figure 3.56 représente le DSE du cas d'utilisation Gestion annuelle du catalogue. La figure 3.57 représente le DSE du cas d'utilisation Publication du catalogue.

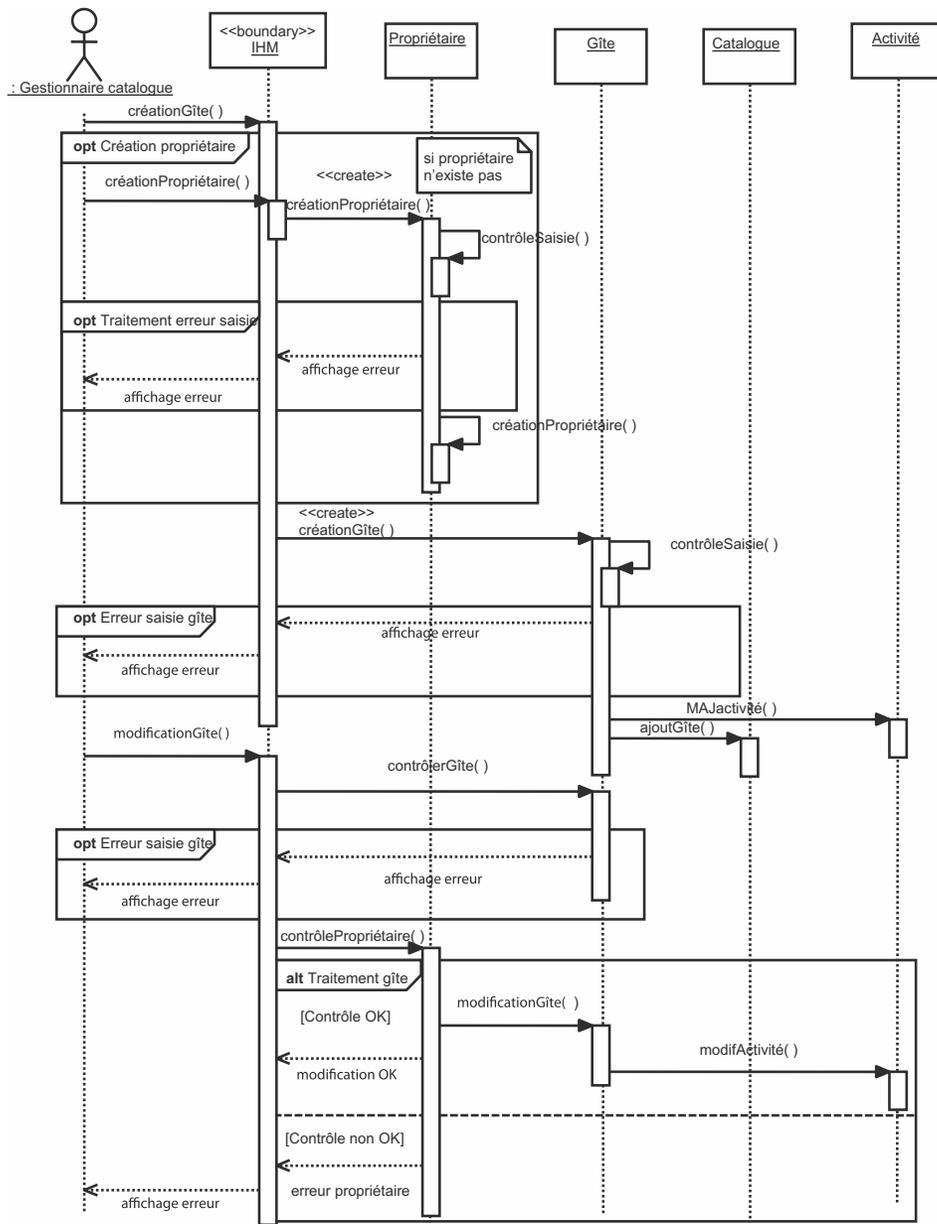


Figure 3.56 — Diagramme de séquence de la Gestion du catalogue de l'exercice de synthèse

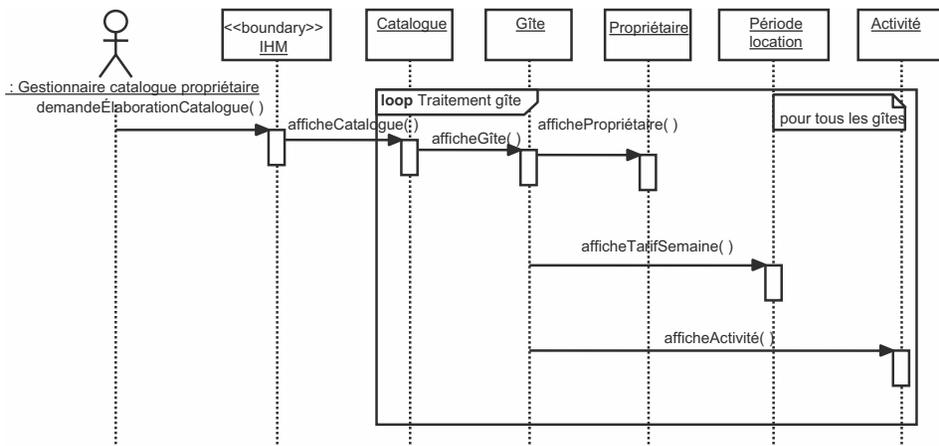


Figure 3.57 — Exemple de diagramme de séquence de l'exemple récapitulatif

3.5 DIAGRAMME DE COMMUNICATION (DCO)

3.5.1 Présentation générale et concepts de base

Le **diagramme de communication** constitue une autre représentation des interactions que celle du diagramme de séquence. En effet, le diagramme de communication met plus l'accent sur l'aspect spatial des échanges que l'aspect temporel.

Rôle

Chaque participant à un échange de message correspondant à une ligne de vie dans le diagramme de séquence se représente sous forme d'un **rôle** dans le diagramme de communication. Un rôle est identifié par :

<nom de rôle> : <nom du type>

Une des deux parties de cette identification est obligatoire ainsi que le séparateur « : ». Le **nom du rôle** correspond au nom de l'objet dans le cas où l'acteur ou la classe ont un rôle unique par rapport au système. Le **nom du type** correspond au nom de la classe lorsque l'on manipule des objets.

Exemple

administrateur : utilisateur

Pour un utilisateur qui est vu au travers de son rôle d'administrateur.

Message

Un **message** correspond à un appel d'opération effectué par un rôle émetteur vers un rôle récepteur. Le sens du message est donné par une flèche portée au-dessus du lien

reliant les participants au message (origine et destinataire). Chaque message est identifié par :

<numéro> : nom ()

Plus précisément l'identification d'un message doit respecter la syntaxe suivante :

[n° du message préc. reçu] « . » n° du message [clause d'itération] [condition]
« : » nom du message.

- Numéro du message précédent reçu : permet d'indiquer la chronologie des messages.
- Numéro du message : numéro hiérarchique du message de type 1.1, 1.2... avec utilisation de lettre pour indiquer la simultanéité d'envoi de message.
- Clause d'itération : indique si l'envoi du message est répété. La syntaxe est * [spécification de l'itération].
- Condition : indique si l'envoi du message est soumis à une condition à satisfaire.

Exemples

1.2.1 * [3 fois] pour un message à adresser trois fois de suite.

1.2a et 1.2b pour deux messages envoyés en même temps.

Exemple récapitulatif de désignation de message :

1.2a.1.1[si t > 100] : lancer()

Ce message signifie :

- 1.2a : numéro du message reçu avant l'envoi du message courant.
- 1.1 : numéro de message courant à envoyer.
- [si t > 100] : message à envoyer si t > 100.
- lancer() : nom du message à envoyer.

3.5.2 Formalisme et exemple

Les rôles correspondent à des objets. Le lien entre les rôles est représenté par un trait matérialisant le support des messages échangés. La figure 3.58 donne le formalisme de base du diagramme de communication.



Figure 3.58 — Formalisme de base du diagramme de communication

Un exemple de diagramme de communication est donné à la figure 3.59.

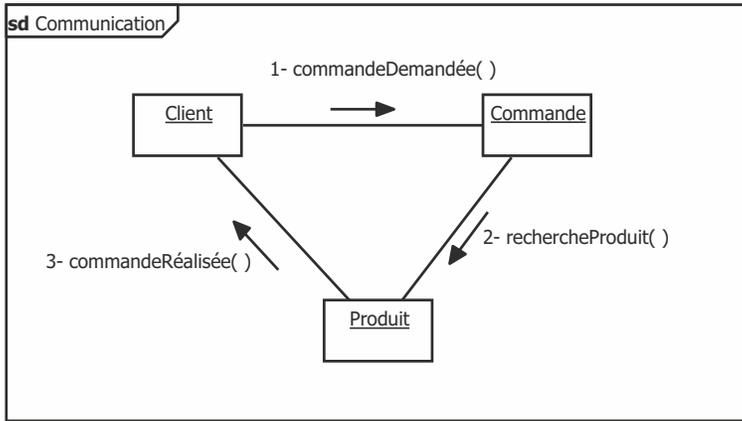


Figure 3.59 — Exemple de diagramme de communication

3.5.3 Exercices

Exercice 1

En reprenant le sujet de l'exercice 1 du diagramme de séquence, nous donnons à la figure 3.60 son équivalent en diagramme de communication.

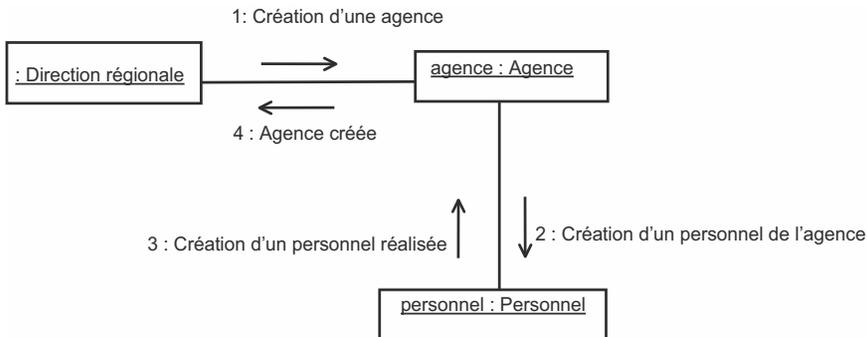


Figure 3.60 — Exemple d'ensemble du diagramme de collaboration

3.6 DIAGRAMME GLOBAL D'INTERACTION (DGI)

3.6.1 Présentation générale et concepts de base

Le **diagramme global d'interaction** permet de représenter une vue générale des interactions décrites dans le diagramme de séquence et des flots de contrôle décrits dans le diagramme d'activité.

Le diagramme global d'interaction privilégie la vue générale des flux de contrôle dans lesquels les nœuds sont des interactions ou des utilisations d'interactions (opérateur ref).

Autrement dit, le diagramme global d'interaction est un diagramme d'activité dans lequel on représente des fragments d'interaction ou des utilisations d'interactions. Ainsi, il est possible de représenter :

- des choix de fragments d'interactions (fusion) ;
- des déroulements parallèles de fragments d'interactions (débranchement et jonction) ;
- des boucles de fragments d'interaction.

Les lignes de vie concernées par le diagramme global d'interaction peuvent être citées dans l'en-tête du diagramme mais ne sont pas à représenter graphiquement.

Concepts manipulés

Le diagramme global d'interaction utilise les concepts du diagramme d'activité auquel on ajoute deux compléments :

- **Les fragments d'interaction du diagramme de séquence** – Il s'agit comme le montre la figure 3.61 de la notion de fragment d'interaction vue dans le diagramme de séquence mais qui ne doit pas être détaillé à ce niveau.

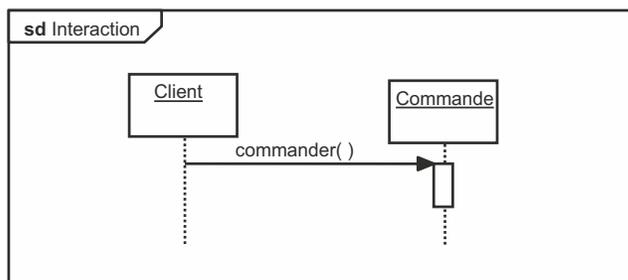


Figure 3.61 – Exemple de fragment d'interaction

- **Les utilisations de fragments d'interaction** – Il est aussi possible de faire appel à des fragments d'interaction à l'aide de l'opérateur ref comme le montre la figure 3.62.



Figure 3.62 – Exemple de fragment d'interaction avec l'opérateur ref

3.6.2 Représentation et exemple

La figure 3.63 donne un exemple de diagramme global d'interaction.

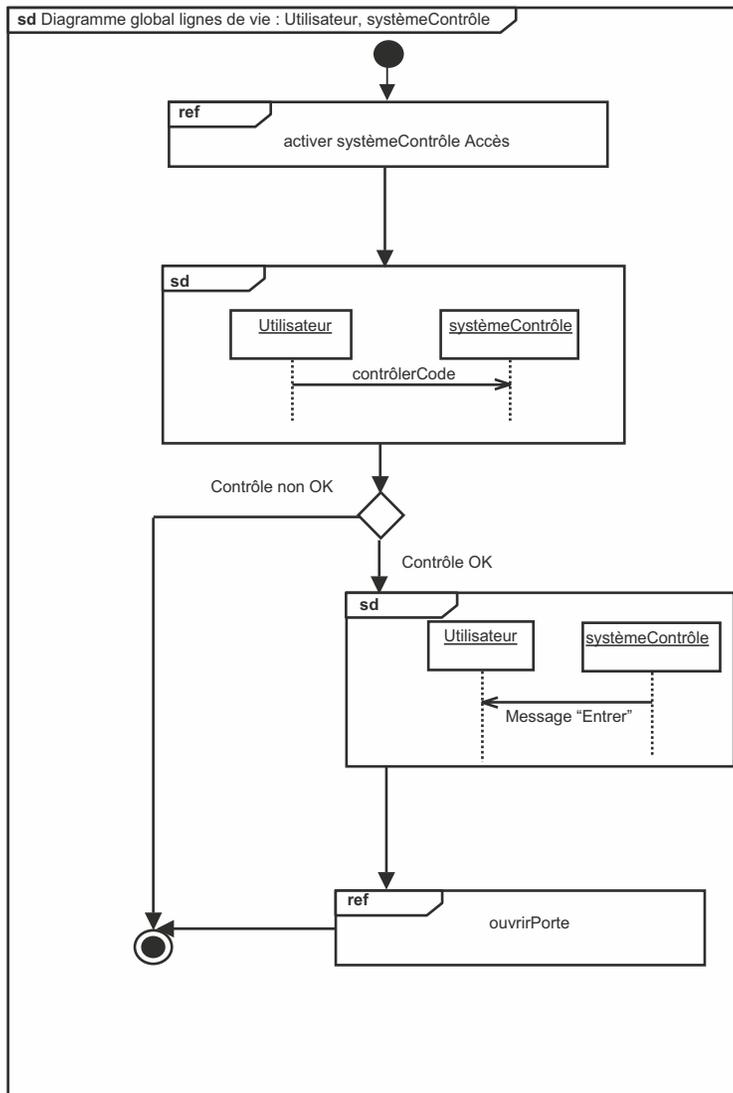


Figure 3.63 — Exemple de diagramme global d'interaction

3.7 DIAGRAMME DE TEMPS (DTP)

3.7.1 Présentation générale et concepts de base

Le **diagramme de temps** permet de représenter les états et les interactions d'objets dans un contexte où le temps a une forte influence sur le comportement du système à gérer.

Autrement dit, le diagramme de temps permet de mieux représenter des changements d'états et des interactions entre objets liés à des contraintes de temps.

Pour cela, le diagramme de temps utilise en plus des lignes de vie, les concepts suivants :

- Des états ou des lignes de temps conditionnées avec deux représentations graphiques possibles.
- Des représentations propres aux aspects temporels : échelle de temps, contrainte de durée, événements...

Concepts manipulés

Le diagramme de temps utilise trois concepts de base :

- **Ligne de vie** – Elle représente l'objet que l'on veut décrire. Elle se dessine de manière horizontale. Plusieurs lignes de vie peuvent figurer dans un diagramme de temps.
- **État ou ligne de temps conditionnée** – Les différents états que peut prendre l'objet d'étude sont listés en colonne permettant ainsi de suivre le comportement de l'objet ligne par ligne (une ligne pour un état).
- **États linéaires** – Il s'agit du même concept que le précédent, mais la représentation de la succession des états est faite de manière linéaire à l'aide d'un graphisme particulier.

3.7.2 Représentation et exemples

Soit à représenter le dispositif de chauffe d'un fer à repasser à vapeur au moment de sa mise en service selon les règles suivantes :

- la pompe à eau qui remplit la chambre de chauffe s'active dès que le témoin d'eau interne le demande ;
- la pompe à eau se désactive dès que le niveau d'eau nécessaire est atteint ;
- le chauffage de l'eau, permettant de produire la vapeur, se met en action à la première mise en service du fer à repasser dès que le niveau d'eau de la chambre de chauffe est suffisant ;
- le chauffage initial de l'eau dure 3 mm permettant ainsi de produire la vapeur.

Dans cet exemple, nous avons deux objets à étudier : pompe à eau et chauffage de l'eau. Nous allons considérer pour chacun d'entre eux deux états significatifs : activé et désactivé.

La figure 3.64 donne la représentation du diagramme de temps en utilisant le formalisme des états « en escalier » et la figure 3.65 fournit la représentation linéaire des états.

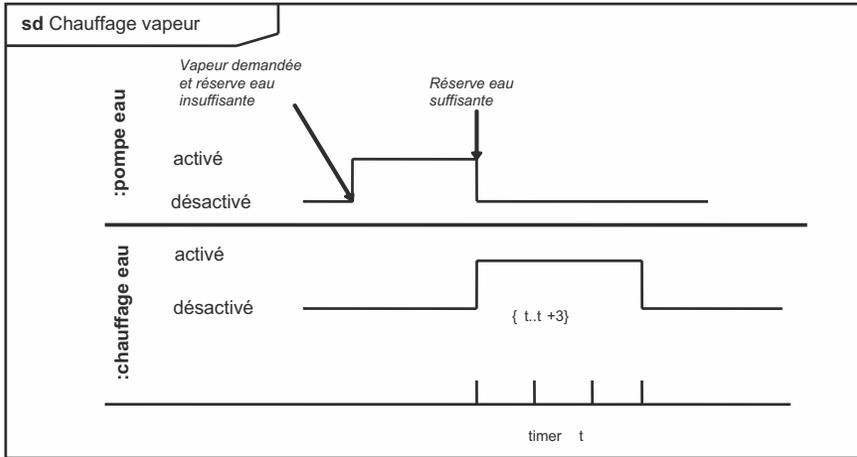


Figure 3.64 — Exemple de diagramme de temps avec représentation « en escalier »

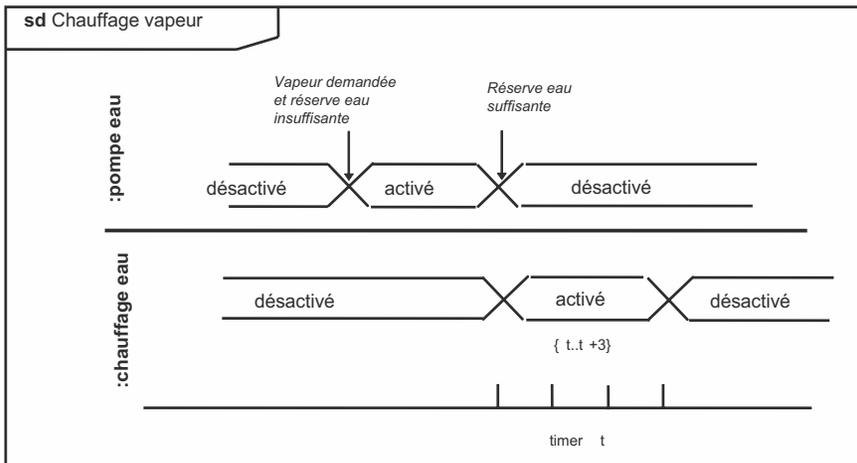


Figure 3.65 — Exemple de diagramme de temps avec représentation linéaire

4

Démarche de développement

Nous proposons dans ce chapitre tout d'abord une synthèse des deux principaux processus de développement objet qui ont été associés à UML. Il s'agit de UP (*Unified Process*) et de RUP (*Rational Unified Process*). Ensuite nous présentons notre propre démarche de développement UP7 qui est fondée sur UP.

4.1 PRÉSENTATION D'UP

UML n'est qu'un langage de modélisation. Nous n'avons pas aujourd'hui dans la norme, de démarche unifiée pour construire les modèles et conduire un projet mettant en œuvre UML. Cependant les auteurs d'UML, ont décrit, dans un ouvrage [Jacobson2000a] le processus unifié (UP, *Unified Process*) qui doit être associé à UML. Nous n'allons pas, dans le cadre de cet ouvrage, donner une présentation détaillée d'UP. Cependant il nous a paru intéressant de dégager les idées fondatrices d'UP dans le cadre d'une présentation générale. Nous allons tout d'abord expliciter les principes de la méthode UP. Nous compléterons ensuite cette présentation générale en décrivant l'architecture à deux dimensions d'UP et ses principaux concepts, nous passerons aussi en revue les différentes phases d'UP, et pour finir nous détaillerons les activités d'UP.

4.2 LES PRINCIPES D'UP

Le processus de développement UP, associé à UML, met en œuvre les principes suivants :

- processus guidé par les cas d'utilisation,
- processus itératif et incrémental,
- processus centré sur l'architecture,
- processus orienté par la réduction des risques.

Ces principes sont à la base du processus unifié décrit par les auteurs d'UML.

4.2.1 Processus guidé par les cas d'utilisation

L'orientation forte donnée ici par UP est de montrer que le système à construire se définit d'abord avec les utilisateurs. Les **cas d'utilisation** permettent d'exprimer les interactions du système avec les utilisateurs, donc de capturer les besoins.

Une seconde orientation est de montrer comment les cas d'utilisation constituent un vecteur structurant pour le développement et les tests du système. Ainsi le développement peut se décomposer par cas d'utilisation et la réception du logiciel sera elle aussi articulée par cas d'utilisation.

4.2.2 Processus itératif et incrémental

Ce type de démarche étant relativement connu dans l'approche objet, il paraît naturel qu'UP préconise l'utilisation du principe de développement par **itérations** successives. Concrètement, la réalisation de maquette et prototype constitue la réponse pratique à ce principe. Le développement progressif, par **incrément**, est aussi recommandé en s'appuyant sur la décomposition du système en cas d'utilisation.

Les avantages du développement itératif se caractérisent comme suit :

- les risques sont évalués et traités au fur et à mesure des itérations,
- les premières itérations permettent d'avoir un feed-back des utilisateurs,
- les tests et l'intégration se font de manière continue,
- les avancées sont évaluées au fur et à mesure de l'implémentation.

4.2.3 Processus centré sur l'architecture

Les auteurs d'UP mettent en avant la préoccupation de **l'architecture du système** dès le début des travaux d'analyse et de conception. Il est important de définir le plus tôt possible, même à grandes mailles, l'architecture type qui sera retenue pour le développement, l'implémentation et ensuite le déploiement du système. Le vecteur des cas d'utilisation peut aussi être utilisé pour la description de l'architecture.

4.2.4 Processus orienté par la réduction des risques

L'analyse des **risques** doit être présente à tous les stades de développement d'un système. Il est important de bien évaluer les risques des développements afin d'aider à la bonne prise de décision. Du fait de l'application du processus itératif, UP contribue à la diminution des risques au fur et à mesure du déroulement des itérations successives.

4.3 LES CONCEPTS ET LES DEUX DIMENSIONS DU PROCESSUS UP

4.3.1 Définition des principaux concepts et schéma d'ensemble

Le processus unifié décrit qui fait quoi, comment et quand les travaux sont réalisés tout au long du cycle de vie du projet. Quatre concepts d'UP répondent à ces questions :

- Rôle (qui ?)
- Activité (comment ?)
- Artefact (quoi ?)
- Workflow (quand ?)

Rôle

Un **rôle** définit le comportement et les responsabilités d'une ressource ou d'un groupe de ressources travaillant en équipe. Le rôle doit être considéré en termes de « casquette » qu'une ressource peut revêtir sur le projet. Une ressource peut jouer plusieurs rôles sur le projet.

Par exemple sur un projet, Paul peut être à la fois chef de projet et architecte. Il représente deux rôles au sens d'UP (fig. 4.1).

Activité

Les rôles ont des **activités** qui définissent le travail qu'ils effectuent. Une activité est une unité de travail qu'une ressource, dans un rôle bien précis, peut effectuer et qui produit un résultat dans le cadre du projet. L'activité a un but clairement établi, généralement exprimée en termes de création ou de mise à jour d'artefacts, comme un modèle, une classe ou un planning. Les ressources sont affectées aux activités selon leurs compétences et leur disponibilité.

Par exemple, les activités « planifier une itération » et « anticiper les risques » sont attribuées au rôle de chef de projet.

Le schéma de la figure 4.1 présente sur un exemple le positionnement et les liens entre ressources, rôles et activités.

Artefacts

Un **artefact** est un ensemble d'informations qui est produit, modifié ou utilisé par le processus. Les artefacts sont les produits effectifs du projet. Les artefacts sont utilisés comme input par les ressources pour effectuer une activité et sont le résultat d'output d'activités du processus.

Un exemple d'artefacts rencontrés au cours du projet est un planning d'une itération ou un diagramme produit dans une activité.

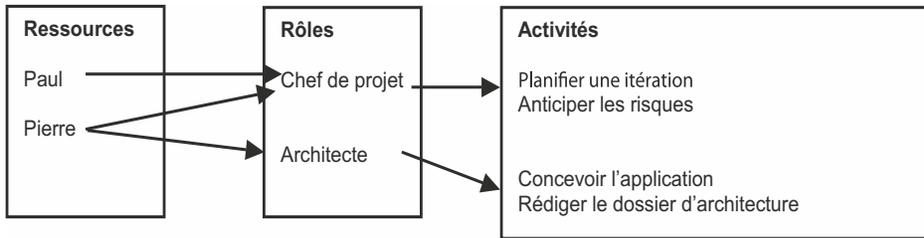


Figure 4.1 – Schéma de positionnement des ressources, rôles et activités

Workflows

Une énumération de tous les rôles, activités et artefacts ne constitue pas un processus. En effet, il est nécessaire d'avoir une façon de décrire des séquences d'activités mesurables qui produisent un résultat de qualité et montre l'interaction entre les ressources. Le **workflow** est une séquence d'activités qui produit un résultat mesurable. En UML, il peut être exprimé par un diagramme de séquence, un diagramme de communication ou un diagramme d'activité.

Schéma d'ensemble

UP peut être décrit suivant deux dimensions traduites en deux axes comme le montre la figure 4.2 :

- Un axe horizontal représentant le temps et montrant l'aspect dynamique du processus. Sur cet axe, le processus est organisé en phases et itérations.
- Un axe vertical représentant l'aspect statique du processus. Sur cet axe, le processus est organisé en activités et workflows.

4.3.2 Phases et itérations du processus (aspect dynamique)

Le processus unifié, organisé en fonction du temps, est divisé en quatre phases successives.

- Inception (Lancement).
- Élaboration.
- Construction.
- Transition.

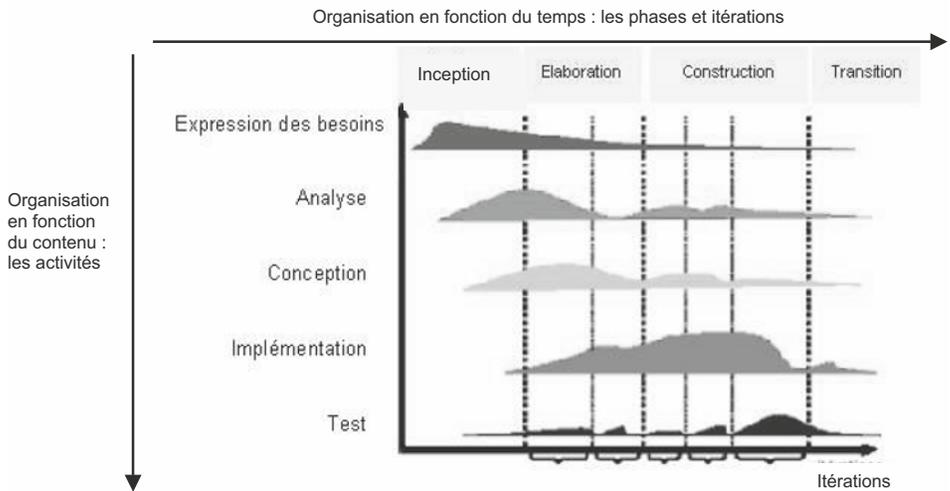


Figure 4.2 — Schéma d'ensemble d'UP

Inception (Lancement)

Cette phase correspond à l'**initialisation du projet** où l'on mène une étude d'opportunité et de faisabilité du système à construire. Une évaluation des risques est aussi réalisée dès cette phase.

En outre, une identification des principaux cas d'utilisation accompagnée d'une description générale est modélisée dans un diagramme de cas d'utilisation afin de définir le périmètre du projet. Il est possible, à ce stade, de faire réaliser des maquettes sur un sous-ensemble des cas d'utilisation identifiés.

Ce n'est qu'à l'issue de cette première phase que l'on peut considérer le projet véritablement lancé.

Élaboration

Cette phase reprend les résultats de la phase d'inception et élargit l'appréciation de la **faisabilité** sur la quasi-totalité des cas d'utilisation. Ces cas d'utilisation se retrouvent dans le diagramme des cas d'utilisation qui est ainsi complété.

Cette phase a aussi pour but d'analyser le domaine technique du système à développer afin d'aboutir à une architecture stable. Ainsi, toutes les exigences non recensées dans les cas d'utilisation, comme par exemple les exigences de performances du système, seront prises en compte dans la conception et l'élaboration de l'architecture.

L'évaluation des risques et l'étude de la rentabilité du projet sont aussi précisées. Un **planning** est réalisé pour les phases suivantes du projet en indiquant le nombre d'itérations à réaliser pour les phases de construction.

Construction

Cette phase correspond à la **production** d'une première version du produit. Elle est donc fortement centrée sur les activités de conception, d'implémentation et de test. En effet, les composants et fonctionnalités non implémentés dans la phase précédente le sont ici.

Au cours de cette phase, la gestion et le contrôle des ressources ainsi que l'optimisation des coûts représentent les activités essentielles pour aboutir à la réalisation du produit. En parallèle est rédigé le manuel utilisateur de l'application.

Transition

Après les opérations de test menées dans la phase précédente, il s'agit dans cette phase de **livrer le produit** pour une exploitation réelle. C'est ainsi que toutes les actions liées au déploiement sont traitées dans cette phase.

De plus, des « bêta tests » sont effectués pour valider le nouveau système auprès des utilisateurs.

Itérations

Une phase peut-être divisée en **itérations**. Une itération est un circuit complet de développement aboutissant à une livraison (interne ou externe) d'un produit exécutable.

Ce produit est un sous-ensemble du produit final en cours de développement, qui croît incrémentalement d'itération en itération pour devenir le système final.

Chaque itération au sein d'une phase aboutit à une livraison exécutable du système.

4.3.3 Activités du processus (aspect statique)

Les activités menées à l'intérieur des quatre phases sont plus classiques, car déjà bien documentées dans les méthodes existantes par ailleurs. Nous nous limiterons donc à ne donner qu'une brève explication de chaque activité.

Expression des besoins

UP propose d'appréhender l'**expression des besoins** en se fondant sur une bonne compréhension du domaine concerné pour le système à développer et une modélisation des procédures du système existant.

Ainsi, UP distingue deux types de besoins :

- les besoins fonctionnels qui conduisent à l'élaboration des cas d'utilisation,
- les besoins non fonctionnels (techniques) qui aboutissent à la rédaction d'une matrice des exigences.

Analyse

L'**analyse** permet une formalisation du système à développer en réponse à l'expression des besoins formulée par les utilisateurs. L'analyse se concrétise par l'élaboration

de tous les diagrammes donnant une représentation du système tant statique (diagramme de classe principalement), que dynamique (diagramme des cas d'utilisation, de séquence, d'activité, d'état-transition...).

Conception

La **conception** prend en compte les choix d'architecture technique retenus pour le développement et l'exploitation du système. La conception permet d'étendre la représentation des diagrammes effectuée au niveau de l'analyse en y intégrant les aspects techniques plus proches des préoccupations physiques.

Implémentation

Cette phase correspond à la **production du logiciel** sous forme de composants, de bibliothèques ou de fichiers. Cette phase reste, comme dans toutes les autres méthodes, la plus lourde en charge par rapport à l'ensemble des autres phases (au moins 40 %).

Test

Les **tests** permettent de vérifier :

- la bonne implémentation de toutes les exigences (fonctionnelles et techniques),
- le fonctionnement correct des interactions entre les objets,
- la bonne intégration de tous les composants dans le logiciel.

Classiquement, différents niveaux de tests sont réalisés dans cette activité : test unitaire, test d'intégration, test de réception, test de performance et test de non-régression.

Après cette présentation d'UP et afin d'éclairer le lecteur sur les principaux processus de développement actuellement utilisés dans l'approche objet, nous donnons ci-après une description générale de RUP.

En son temps, la société Rational Software (rachetée par IBM) avait développé une version spécifique d'UP sous le nom de RUP (*Rational Unified Process*), cette démarche a fait l'objet d'un ouvrage [Kruchten2000]. Dans la présentation qui suit, nous avons surtout mis l'accent sur les principaux apports de RUP par rapport à UP.

4.4 LES PRINCIPAUX APPORTS DE RUP

RUP® (*Rational Unified Process*) est un processus basé sur une approche disciplinée afin de bien maîtriser l'assignation des tâches et la responsabilisation des différents acteurs participant au cycle de développement du logiciel. RUP a pour objectif principal de faire appliquer les bonnes pratiques de développement aux entreprises, ce qui confère au produit final une meilleure qualité.

RUP se veut être un modèle évolutif qui doit être configuré pour pouvoir être utilisé en intégrant les contraintes, les spécificités et l'historique de l'organisation qui l'adopte.

Nous allons présenter les principaux apports de RUP par rapport à UP en traitant les points suivants :

- les bonnes pratiques,
- les phases du processus,
- les activités du processus.

4.4.1 Les bonnes pratiques

RUP adhère à six bonnes pratiques de développement observées dans l'industrie pour leurs succès. Sur ces six bonnes pratiques, trois sont issues des principes d'UP :

- Développement itératif et incrémental.
- Développement piloté par les cas d'utilisation.
- Forte importance de l'architecture.

Trois autres bonnes pratiques ont été introduites par RUP :

- Modélisation visuelle.
- Vérification continue de la qualité.
- Contrôle des changements du logiciel.

Modélisation visuelle

RUP préconise l'utilisation d'un langage de modélisation standard comme UML qui permet aux membres de l'équipe de développement de communiquer sans ambiguïté.

L'utilisation d'outils de **modélisation visuelle** est fortement recommandée par RUP. Ceux-ci permettent de modéliser l'architecture et ses composants à l'aide de diagrammes. De plus, ils facilitent la gestion des modèles de RUP et contribuent à maintenir la cohérence entre les différentes phases du processus : de l'expression des besoins à l'implémentation. En résumé, la modélisation visuelle permet de gérer la complexité des logiciels.

Vérification continue de la qualité

RUP met l'accent sur l'importance d'évaluer continuellement la **qualité** d'un système du point de vue des fonctionnalités, de la fiabilité et de la performance. Pour cela, RUP vous assiste dans la planification, la conception, l'implémentation et l'exécution des tests adéquats.

Ces tests sont réalisés tout au long du processus, dans toutes les activités, en impliquant tous les acteurs, et en utilisant des critères et des mesures objectifs (ex. tests d'intégration continus).

Contrôle des changements du logiciel

RUP propose une coordination des activités et des livrables des équipes afin de gérer activement les **changements du logiciel**. Pour cela le processus organise les activités en enchaînement d'activités (workflows). Ces workflows décrivent comment contrôler, suivre et mesurer les changements du logiciel. De plus, ils permettent une meilleure allocation des ressources basée sur les priorités et les risques du projet et facilitent la gestion du travail sur ces changements au travers des itérations. Combinée au développement itératif, cette technique permet de contrôler les changements de sorte qu'il soit possible de découvrir rapidement les éventuels problèmes et d'y réagir.

4.4.2 Les phases et les activités du processus

Comme UP, RUP est un **processus** à deux dimensions. Il est modélisé par un schéma articulé suivant deux axes (fig. 4.3) :

- L'axe horizontal représentant le temps et montrant les phases et les itérations du processus,
- L'axe vertical représentant l'aspect statique du processus. Les activités sont représentées sur cet axe, RUP propose neuf activités (quatre de plus que le processus UP).

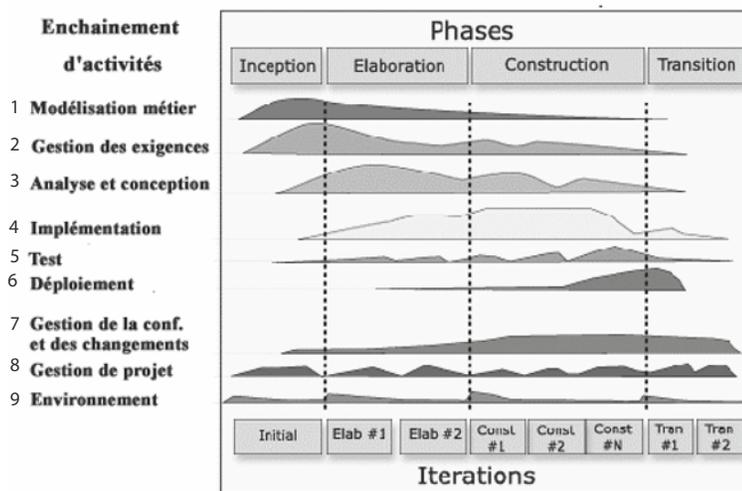


Figure 4.3 — Schéma d'ensemble de RUP

Les phases et les jalons

Les phases dans le processus RUP sont les mêmes que celles du processus UP. Le concept de **jalon** est introduit par RUP. Chaque phase est conclue par un jalon. Ce

jalón est constitué d'un ensemble de critères d'évaluation. Ces critères doivent être satisfaits pour passer à la phase suivante. La figure 4.4 illustre les différents jalons au cours du processus.

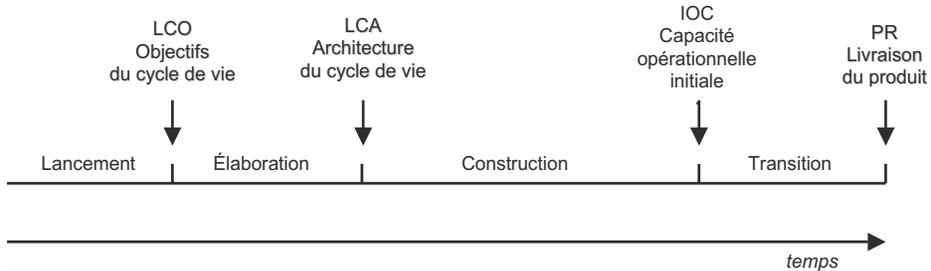


Figure 4.4 – Les phases et jalons dans RUP

Le jalon LCO

La phase de lancement se termine par le jalon « Objectifs du cycle de vie ». Ce jalon permet de s'assurer que les critères suivants sont bien pris en compte par le projet :

- Les exigences fondamentales, les caractéristiques clés et les contraintes principales sont documentées.
- Étude de rentabilité définie et approuvée.
- Estimation de charge réaliste, phases identifiées, priorités définies et risques définis.
- Planning de phase d'élaboration défini.
- Périmètre défini.

Le jalon LCA

La phase d'élaboration se termine par le jalon « Architecture du cycle de vie ». Ce jalon permet de s'assurer que les critères suivants sont bien appliqués dans le projet :

- Un ou plusieurs prototypes ont été réalisés pour explorer les fonctionnalités critiques et les scénarios architecturalement significatifs.
- Architecture du projet définie et stable.
- Risques définis et pris en compte dans l'architecture.
- Planning des phases suivantes détaillé et précis.

Le jalon IOC

La phase de construction se termine par le jalon « Capacité opérationnelle initiale ». Ce jalon permet de s'assurer que les critères suivants sont bien appliqués dans le projet :

- Version du produit assez stable et mature pour être fournie au client.
- Les tests sont établis et développés pour valider les versions exécutables.

- Les clients sont prêts à accueillir le produit.
- Les risques sont maîtrisés.
- Le plan d'itération pour la phase de transition est complété et vérifié.

Le jalon PR

La phase de transition se termine par le jalon « Livraison du produit ». Ce jalon permet de s'assurer que les critères suivants sont bien appliqués dans le projet :

- Le produit complet et achevé en accord avec les exigences du client est déployé.
- Le client est satisfait.
- Les dépenses du projet correspondent aux dépenses prévues.

Les activités

Les **activités** de RUP, décrites ci-après, sont celles qui sont nouvelles par rapport à UP. Les autres activités sont déjà décrites dans UP.

Modélisation métier

La **modélisation métier** permet de mieux comprendre la structure et la dynamique de l'organisation étudiée. Elle assure au client que les utilisateurs finaux et les développeurs partagent une vision commune de l'organisation. Elle permet d'aboutir à une cartographie des processus métier de l'organisation cliente.

Gestion des exigences

La **gestion des exigences** a pour but de définir ce que doit faire le système. Pour cela, les cas d'utilisation sont réalisés. Ils permettent aussi de structurer les documents de spécifications fonctionnelles. Les cas d'utilisation sont en effet décomposés en scénarios d'usage du système dans lesquels l'utilisateur décrit ce qu'il fait grâce au système et ses interactions avec le système (description détaillée des cas d'utilisation).

Les exigences non fonctionnelles (performances, qualités...) peuvent aussi être décrites dans un document complémentaire de spécification.

Déploiement

Le but de l'enchaînement des activités de **déploiement** est de livrer le produit aux utilisateurs finaux. Cela inclut de nombreuses sous-activités :

- Packaging du logiciel.
- Livraison du logiciel.
- Migration des données existantes (dans certains cas).
- Installation du logiciel.
- Assistance aux utilisateurs.

Ces sous-activités sont réalisées pour la plupart lors de la phase de transition. Néanmoins, un certain nombre d'entre elles doivent être préparées lors de la phase de construction comme par exemple le packaging du logiciel.

Gestion de la configuration et du changement

L'objectif de la **gestion de la configuration et des changements** est de garder la trace de tous les éléments tangibles qui participent au développement et de suivre leur évolution.

Pour cela, il faut traiter les points suivants :

- Identifier les éléments en configuration du projet.
- Contrôler les modifications de ces éléments *via* un outil de gestion de configuration.
- Gérer les configurations de ces éléments au cours du processus de développement du logiciel à l'aide d'espaces de travail (espace de développement, espace d'intégration et espace de livraison).
- Gérer les changements des éléments en configuration du projet à l'aide de demandes de changement. Celles-ci sont utilisées pour documenter et tracer les anomalies, les demandes d'évolution et tout type de requête entraînant une modification du produit. Elles assurent que les impacts des modifications sont pris en compte à tous les niveaux.
- Auditer la mise en place de l'activité de gestion de configuration et du changement.

La gestion de configuration est réalisée tout au long du projet. La gestion du changement s'applique principalement dans les phases de construction et de transition.

Gestion de projet

La **gestion d'un projet** logiciel est l'art de mesurer les objectifs compétitifs, de gérer les risques et les contraintes pour fournir un produit qui satisfait les besoins exprimés par les utilisateurs.

Pour cela, RUP insiste sur l'aspect itératif du processus de développement et fournit un cadre de travail déjà documenté par :

- un framework pour la gestion d'un projet logiciel,
- des conseils pour la planification, l'allocation des ressources, la prise de décision et le suivi de projet,
- un framework pour gérer les risques.

Cette approche permet d'augmenter les chances de succès du projet.

Environnement

La gestion de l'**environnement** consiste à mettre en place tout ce qui permet aux différents acteurs/rôles de réaliser au mieux leurs activités. Ce qui revient à fournir à l'organisation, l'environnement de développement – tant processus qu'outils – qui va aider l'équipe de développement.

Cet environnement peut être adapté pour chaque projet. Ainsi RUP fournit des modèles et des outils facilement adaptables aux différents types de projet.

4.5 DÉMARCHE DE DÉVELOPPEMENT UP7

4.5.1 Présentation générale

Schéma d'ensemble

Nous proposons ici une démarche d'application d'UML qui prend appui sur UP mais qui se veut avant tout être pragmatique. Cette démarche est fondée d'une part sur notre vision du processus de développement et d'autre part sur notre propre expérience tirée de la réalisation en entreprise de projets avec UML.

La démarche que nous proposons est articulée suivant deux axes : les quatre phases qui correspondent à celles d'UP et sept activités. Ainsi, on peut présenter dès ce stade un premier schéma d'ensemble de la démarche suivant ces deux axes (fig. 4.5).

PHASES ► ACTIVITÉS ▼	Lancement	Élaboration	Construction	Transition
1- Modélisation métier				
2- Exigences fonctionnelles				
3- Analyse des cas d'utilisation				
4- Synthèse de l'analyse				
5- Conception				
6- Implémentation				
7- Test				

Figure 4.5 — Schéma d'ensemble de la démarche UP7

Le grisé sur le schéma représente l'effort à consentir pour chaque activité durant les phases d'un projet. Ainsi par exemple, pour l'activité 3 (Analyse des cas d'utilisation), l'activité commence légèrement lors de la phase de lancement puis se déroule principalement lors de la phase d'élaboration et se termine en phase de construction.

Il est à noter que sur le schéma, la proportion que représente chaque activité par rapport à l'ensemble de la charge de développement d'un projet a été respectée graphiquement.

Compte tenu de notre expérience et des ratios habituellement constatés dans la profession, nous avons retenu la répartition indicative suivante pour les volumes d'effort à consentir sur les activités d'un projet.

- Modélisation métier : 5 %.
- Exigences fonctionnelles : 5 %.
- Analyse des cas d'utilisation : 20 %.
- Synthèse de l'analyse : 5 %.
- Conception : 10 %.
- Implémentation : 40 %.
- Test : 15 %.

Il importe bien entendu, de tenir compte pour chaque projet de ses spécificités en termes par exemple de complexité fonctionnelle, contraintes techniques, et compétence des ressources affectées.

Les activités 6 et 7, « Implémentation » et « Tests », sont présentes sur notre schéma pour couvrir à ce niveau la totalité des activités en référence à UP, mais elles ne sont pas traitées dans l'ouvrage.

Cette démarche est par définition itérative. Il est ainsi possible, si nécessaire, de traiter des itérations à l'intérieur de chaque phase. Chaque itération doit se conclure par un produit livrable sous forme de maquette ou de prototype. Des itérations successives permettent d'affiner les résultats de chaque phase.

Activité 1 – Modélisation métier

La première activité de la démarche consiste à mieux connaître et comprendre les processus dans lesquels va s'intégrer le futur système informatique. Cette activité aboutit à trois résultats :

- Au positionnement du système à étudier au sein de l'ensemble des processus de l'entreprise et à la définition du périmètre fonctionnel global du système (schéma de contexte du domaine d'étude).
- À la définition des processus métiers concernés par le système à développer et à l'identification des acteurs (diagramme d'activité).
- À la définition des concepts métiers du domaine sous forme de classe (diagramme de classe métier). Les concepts métiers correspondent aux informations créées, transformées ou manipulées par les experts du domaine. L'expert du domaine y retrouve le vocabulaire de son métier.

À l'issue de cette activité, le périmètre du système à étudier est défini.

Activité 2 – Exigences fonctionnelles

La deuxième activité de la démarche a pour but de définir ce que doit faire le système d'un point de vue métier. Cette activité permet d'obtenir trois résultats :

- La définition des cas d'utilisation métier et leur description générale (diagramme de cas d'utilisation système).
- Les scénarios des cas d'utilisation métier (diagramme de séquence système).
- La navigation générale du système à étudier, c'est-à-dire l'interface homme-machine (schéma de navigation générale).

Au terme de ces deux premières activités, l'expression des besoins (au sens UP) est couverte.

Activité 3 – Analyse des cas d'utilisation

La troisième activité de la démarche a pour but de fournir une vue informatique du système. Cette activité permet d'obtenir cinq résultats :

- La définition de tous les cas d'utilisation (métiers + informatiques) et leur description détaillée (diagramme des cas d'utilisation).
- L'identification des scénarios pour chaque cas d'utilisation (diagramme de séquence).
- Les différents états des objets étudiés (diagramme d'état-transition). Cette partie de l'activité est optionnelle et s'applique selon les systèmes étudiés.
- Les interfaces utilisateur pour chaque cas d'utilisation.
- Les classes pour chaque cas d'utilisation (diagramme de classe).

À l'issue de cette activité, l'analyse des cas d'utilisation est produite mais non encore consolidée ni validée définitivement.

Activité 4 – Synthèse de l'analyse

La quatrième activité de la démarche consiste à consolider et valider toute l'analyse des cas d'utilisation. Cette activité permet d'obtenir deux résultats :

- Le regroupement de l'ensemble des classes en un seul diagramme (diagramme de classe récapitulatif).
- La validation de l'analyse de chaque cas par le biais d'une matrice de validation. Celle-ci permet de vérifier que l'analyse des cas est complète, c'est-à-dire que tous les cas d'utilisation métier ont été repris dans l'analyse.

Au terme des activités 3 et 4, l'analyse (au sens activité UP) est couverte.

Activité 5 – Conception

La cinquième activité de la démarche se concentre sur le « comment faire ». Elle a pour but de définir et de mettre en place les choix d'architecture technique, et de compléter la description du système sous l'angle technique. Cette activité permet d'obtenir quatre résultats :

- Les choix techniques retenus.
- Les scénarios techniques par cas d'utilisation (diagrammes de séquence technique).
- Les classes techniques par cas d'utilisation (diagrammes de classe technique).
- Le regroupement sous forme de paquetage de l'ensemble des classes techniques en un seul diagramme (diagramme de paquetage).

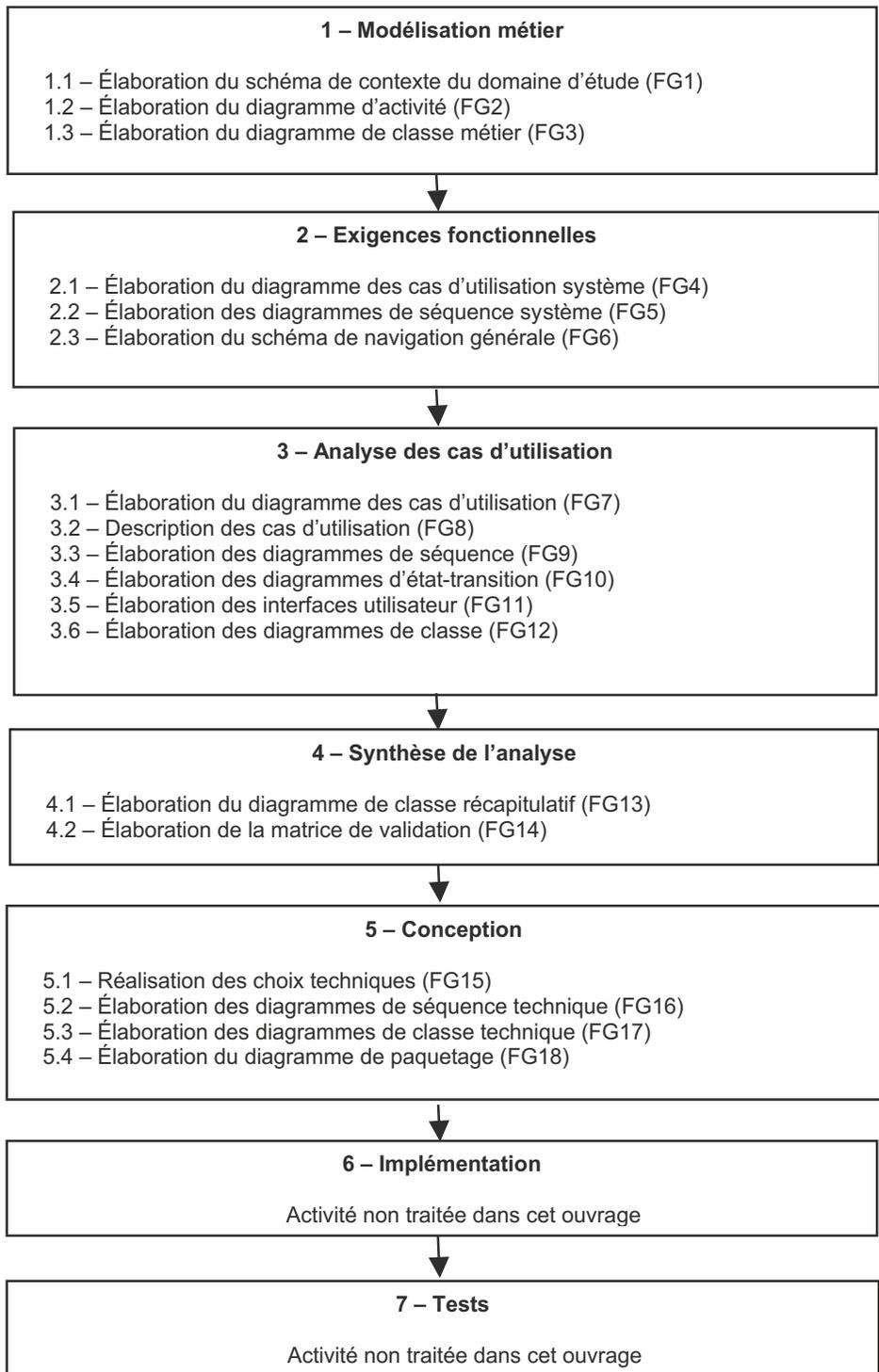
De ce fait la conception préliminaire est couverte par cette activité. La conception détaillée qui est la traduction des scénarios et classes techniques dans les solutions technologiques retenues n'est pas traitée dans cet ouvrage.

Cette activité couvre la conception (au sens UP).

Les activités 6 et 7, « Implémentation » et « Tests » se réfèrent aux activités d'UP, mais ne sont pas traitées dans l'ouvrage.

Schéma de la démarche

Pour illustrer la description générale des activités de la démarche, la figure 4.6 présente chaque activité avec ses différentes sous-activités. Une sous-activité aboutit à l'élaboration d'un ou plusieurs diagrammes UML ou d'un schéma de support au développement du système (hors UML). À chaque sous-activité est associée une fiche guide qui est ensuite détaillée.

**Figure 4.6** – Schéma détaillé de la démarche

4.5.2 Description des activités (fiche guide par sous-activité)

Nous proposons au lecteur une description de la démarche à l'aide de fiches guides (FG). Un plan standard a été adopté pour la présentation de ces fiches. Ce plan traite : l'objectif, le point de départ, le point d'arrivée, la démarche d'élaboration et les recommandations.

Ces fiches peuvent être reprises et adaptées pour chaque projet compte tenu de son contexte et de ses particularités.

Modélisation métier

FICHE GUIDE – FG1	
Activité 1 : Modélisation métier Sous-activité 1.1 : Élaboration du schéma de contexte du domaine d'étude	
Projet :	
Date :	
Objectif	Positionner le système à étudier au sein des processus de l'entreprise.
Point de départ	Esquisse fonctionnelle du besoin exprimé.
Point d'arrivée	Système à étudier positionné par rapport aux processus de l'entreprise et périmètre fonctionnel défini.
Démarche d'élaboration	
1 Identifier les processus connexes au système étudié. 2 Déterminer les interactions entre les processus connexes et le système étudié. 3 Préciser le périmètre du système à étudier (définition des sous-ensembles fonctionnels).	
Recommandation : Mettre en évidence le sous-ensemble à étudier (sous-ensemble grisé/mis en pointillé) dans le schéma de contexte.	

FICHE GUIDE – FG2	
Activité 1 : Modélisation métier	Projet :
Sous-activité 1.2 : Élaboration du diagramme d'activité	Date :
Objectif	Décrire les processus métiers du système à étudier.
Point de départ	Système à étudier positionné par rapport aux processus de l'entreprise et périmètre fonctionnel défini.
Point d'arrivée	Acteurs identifiés et processus métiers du système étudié définis (flot de contrôle, flot de données...) dans le DAC.
Démarche d'élaboration	
<p>1 Identifier les acteurs internes et externes du système étudié.</p> <p>2 Identifier des actions du processus.</p> <p>3 Définir le flot de contrôle (enchaînement des actions) :</p> <ul style="list-style-type: none"> – transitions automatiques, – transitions gardées, – synchronisation, – début/fin du flot. <p>4 Représenter les données liées aux actions. Ces données sont décrites à l'aide des concepts du domaine. Ces concepts permettent d'initialiser le diagramme de classe métier.</p> <p>5 Déterminer le flot de données c'est-à-dire l'enchaînement des données entre elles et/ou avec des actions.</p> <p>6 Décrire les rôles des acteurs du système.</p>	
<p>Recommandations : Veiller à la lisibilité du DAC :</p> <ul style="list-style-type: none"> – Limiter le nombre d'activités-actions représentées soit en privilégiant celles qui sont structurantes, soit en utilisant une présentation à plusieurs niveaux. – Essayer dans la mesure du possible de tracer des flots faciles à lire (éviter les traits obliques). 	

FICHE GUIDE – FG3	
Activité 1 : Modélisation métier Sous-activité 1.3 : Élaboration du diagramme de classe métier	
Projet :	
Date :	
Objectif	Définir les concepts métiers du domaine sous forme de classe.
Point de départ	Acteurs identifiés et processus métiers du système étudié définis (flot de contrôle, flot de données).
Point d'arrivée	Concepts métiers identifiés définis dans le DCL métier et le glossaire métier.
Démarche d'élaboration	
<p>1 Identifier les concepts du domaine sous forme de classes en prenant comme base ceux définis dans le diagramme d'activité.</p> <p>2 Préciser les principaux attributs utiles à la compréhension des experts métiers.</p> <p>3 Déterminer les relations entre les classes : – nom de l'association, – multiplicité.</p> <p>4 Décrire de manière générale les concepts du domaine afin d'obtenir un glossaire métier.</p>	
Recommandations : 1 Se limiter à ce niveau aux concepts structurants pour le système à étudier. 2 Élaborer en synthèse un dossier de modélisation métier.	

Exigences fonctionnelles

FICHE GUIDE – FG4	
Activité 2 : Exigences fonctionnelles	
Sous-activité 2.1 : Élaboration du diagramme des cas d'utilisation système	
Projet :	
Date :	
Objectif	Recueillir et décrire les besoins métiers des acteurs du système (boîte noire).
Point de départ	Concepts métiers identifiés et définis dans le DCL métier et le glossaire métier.
Point d'arrivée	Besoins métiers recueillis sous la forme de cas d'utilisation (DCU système) et décrits de manière générale.
DÉMARCHE D'ÉLABORATION	
<p>1 Identifier les acteurs métiers* du système (acteurs internes et acteurs et/ou systèmes externes) en prenant comme base ceux définis dans le DAC.</p> <p>2 Identifier les cas d'utilisation métiers**.</p> <p>3 Représenter les interactions entre les acteurs métiers et les cas d'utilisation métiers.</p> <p>4 Définir les dépendances entre les cas d'utilisation métiers.</p> <p>5 Décrire de manière générale les cas d'utilisation métiers.</p> <p>(*) Les acteurs métiers identifiés lors de cette étape sont ceux correspondant à une vue métier du système. (**) Les cas d'utilisation identifiés lors de cette étape sont ceux correspondant à une vue métier du système.</p>	
Recommandations : <p>1 Utiliser des verbes pour décrire les cas d'utilisation.</p> <p>2 Il est bon de se limiter à moins d'une dizaine de cas d'utilisation à ce niveau là.</p>	

FICHE GUIDE – FG5	
Activité 2 : Exigences fonctionnelles Sous-activité 2.2 : Élaboration des diagrammes de séquence système	
Projet :	
Date :	
Objectif	Définir les interactions entre les acteurs métiers et le système (boîte noire) pour tous les cas d'utilisation métiers.
Point de départ	Besoins métiers recueillis sous la forme de cas d'utilisation et décrits de manière générale.
Point d'arrivée	Interactions entre acteurs métiers et système définis dans le DSE.
Démarche d'élaboration	
Pour chaque cas d'utilisation (CU) : 1 Reporter le ou les acteurs du CU sélectionné sur le diagramme de séquence système (DSE). 2 Représenter le système sous forme d'objet. 3 Déterminer les messages échangés entre les acteurs et le système (synchrones, asynchrones, résultats). 4 Représenter les fragments d'interaction combinés si nécessaire (loop, alt, opt...).	
Recommandation : Ne pas chercher, dans cette activité, à décrire le détail des interactions entre les objets du système.	

FICHE GUIDE – FG6	
Activité 2 : Exigences fonctionnelles	Projet :
Sous-activité 2.3 : Élaboration du schéma de navigation générale	Date :
Objectif	Déterminer la cinématique des écrans du système.
Point de départ	Interactions entre acteurs et système définies pour les cas d'utilisation métiers.
Point d'arrivée	L'interface homme-machine générale définie.
Démarche d'élaboration	
<p>1 Identifier les principaux écrans provenant des cas d'utilisation métiers et des interactions décrites dans le DSE système. Les écrans correspondent aux états du schéma de navigation générale si l'on utilise un diagramme d'état-transition pour la représentation de la navigation.</p> <p>2 Préciser les évènements/conditions associés à la transition entre les écrans.</p> <p>3 Indiquer l'état début/fin si nécessaire.</p>	
<p>Recommandation :</p> <p>Se limiter à une première vue générale de l'enchaînement des écrans correspondant à la vision métier.</p>	

Analyse des cas d'utilisation

FICHE GUIDE – FG7	
Activité 3 : Analyse des cas d'utilisation	Projet :
Sous-activité 3.1 : Élaboration du diagramme des cas d'utilisation	Date :
Objectif	Définir la totalité des cas d'utilisation (métiers et informatiques). Les cas d'utilisation informatiques sont des fonctions complémentaires qui n'ont pas été identifiées lors de l'activité précédente (ex. un module d'administration).
Point de départ	Besoins métiers, interactions acteurs métiers/système, IHM définies.
Point d'arrivée	Tous les cas d'utilisation définis dans le DCU.
Démarche d'élaboration	
<p>1 Identifier tous les acteurs du système en prenant comme base ceux définis dans le DCU système de l'activité précédente. Les acteurs informatiques apparaissent à ce niveau de l'analyse (ex. Administrateur d'une application).</p> <p>2 Identifier tous les cas d'utilisation en prenant comme base ceux définis dans le DCU système de l'activité précédente. Ceux-ci sont souvent décomposés à un niveau plus détaillé. Les cas d'utilisation informatiques apparaissent à ce niveau de l'analyse.</p> <p>3 Représenter les interactions entre les acteurs et les cas d'utilisation.</p> <p>4 Définir les dépendances entre les cas d'utilisations.</p>	
<p>Recommandation :</p> <p>Veiller à limiter le nombre de cas d'utilisation. Ne pas dépasser la dizaine par niveau de description.</p>	

FICHE GUIDE – FG8	
Activité 3 : Analyse des cas d'utilisation	
Sous-activité 3.2 : Description des cas d'utilisation	
Projet :	
Date :	
Objectif	Décrire de manière textuelle et détaillée les cas d'utilisation.
Point de départ	Tous les cas d'utilisation définis.
Point d'arrivée	Cas d'utilisation décrits de manière textuelle.
Démarche d'élaboration	
Pour chaque cas d'utilisation (CU) : <ol style="list-style-type: none"> 1 Identifier l'objectif du CU. 2 Identifier les acteurs du CU à partir du DCU de la sous-activité précédente. 3 Définir les pré conditions et éventuellement post conditions du CU. 4 Décrire le scénario nominal sous forme de liste d'actions avec une numérotation chronologique (1- Action 1, 2- Action 2...). L'objectif du CU doit être atteint au terme du scénario. 5 Décrire le(s) scénario(s) alternatif(s) sous forme de liste d'actions avec une numérotation chronologique le reliant au scénario nominal (1a : titre 1, Cas alternatif « a » de l'étape 1 du scénario nominal). Le(s) scénario(s) alternatif(s) doit s'achever par la satisfaction ou l'abandon de l'objectif. 	
Recommandations : <ol style="list-style-type: none"> 1 Pour le scénario nominal : <ul style="list-style-type: none"> – Utiliser 3 à 9 actions. – Éviter de faire apparaître les détails de l'IHM dans les actions. – Utiliser des phrases courtes et simples pour les actions. – Ne pas utiliser les « si ... sinon » dans les actions. 2 Pour le scénario alternatif : <ul style="list-style-type: none"> – Terminer celui-ci par une action de ce type : « le CU se termine ... » ou le « le CU reprend au point N ... ». 	

FICHE GUIDE – FG9	
Activité 3 : Analyse des cas d'utilisation	
Sous-activité 3.3 : Élaboration des diagrammes de séquence	
Projet :	
Date :	
Objectif	Représenter les interactions entre les acteurs et les objets du système pour tous les cas d'utilisation en considérant les différents scénarios décrits.
Point de départ	Cas d'utilisation décrits de manière textuelle et concepts métiers identifiés et définis.
Point d'arrivée	Interactions entre acteurs et objets du système décrits dans le DSE pour tous les cas d'utilisation.
Démarche d'élaboration	
Pour chaque cas d'utilisation (CU) et chaque scénario identifié :	
1 Reporter le ou les acteurs du DCU impliqués dans le scénario.	
2 Représenter les objets du système impliqués dans le scénario en prenant comme base ceux identifiés lors de l'activité de modélisation métier.	
3 Identifier les messages (synchrones/asynchrones) entre les acteurs et les objets et entre les objets eux-mêmes. La chronologie des échanges doit être respectée.	
4 Modéliser les fragments d'interactions si nécessaire.	
Recommandations :	
1 Se limiter à la représentation des principaux scénarios de chaque CU.	
2 Pour des messages ou des fragments d'interactions complexes, les expliciter par des notes.	
3 Utiliser des fragments d'interaction de type « ref » pour faciliter la lecture du DSE.	

FICHE GUIDE – FG10	
Activité 3 : Analyse des cas d'utilisation Sous-activité 3.4 : Élaboration des diagrammes d'état-transition	
Projet : Date :	
Objectif	Définir d'une part les états des objets significatifs du système étudié et d'autre part les actions associées aux transitions.
Point de départ	Cas d'utilisation décrits de manière textuelle et concepts métiers identifiés et définis.
Point d'arrivée	Les différents états des objets et les transitions entre objets représentés dans le DET.
Démarche d'élaboration	
Pour chaque objet significatif du système étudié : <ol style="list-style-type: none"> 1 Identifier les états pertinents de l'objet à représenter. Considérer les états élémentaires mais aussi composites si nécessaire. 2 Définir les transitions entre les états des objets avec éventuellement les gardes associées. Utiliser, le cas échéant, tous les opérateurs disponibles (points de jonction, points de choix...). 3 Définir les actions pour les transitions et les activités pour les objets. 4 Représenter l'état initial et le ou les états finaux. 	
Recommandation : Utiliser uniquement les diagrammes d'état pour des objets dont il est nécessaire de comprendre le comportement dans tout le système (états caractéristiques).	

FICHE GUIDE – FG11	
Activité 3 : Analyse des cas d'utilisation	
Sous-activité 3.5 : Élaboration des interfaces utilisateur	
Projet :	
Date :	
Objectif	Modéliser les interfaces utilisateur de tous les cas d'utilisation pour donner une vue concrète de l'application aux utilisateurs.
Point de départ	Cas d'utilisation décrits de manière textuelle et interactions entre acteurs et objets du système définies pour tous les cas d'utilisation.
Point d'arrivée	Interfaces utilisateur définies pour tous les cas d'utilisation.
Démarche d'élaboration	
Pour chaque cas d'utilisation (CU) :	
1 Identifier toutes les « entrées » de l'IHM qui correspondent aux paramètres des messages du DSE pour chaque CU.	
2 Représenter les dispositifs d'« entrées » sous forme de composants IHM : <ul style="list-style-type: none"> – zone de saisie à l'écran, – liste de choix, – boutons radios... 	
3 Ajouter à l'IHM, les composants de validation (boutons).	
4 Représenter les « sorties » de l'IHM qui correspondent aux résultats du message du DSE pour chaque CU.	
Recommandation : Réaliser des interfaces utilisateur simples facilement modifiables compte tenu des fréquents retours des utilisateurs à traiter.	

FICHE GUIDE – FG12	
Activité 3 : Analyse des cas d'utilisation	Projet :
Sous-activité 3.6 : Élaboration des diagrammes de classe	Date :
Objectif	Définir les classes pour chaque cas d'utilisation.
Point de départ	Interactions entre acteurs et objets du système définies pour tous les cas d'utilisation et concepts métiers définis dans la modélisation métier.
Point d'arrivée	Classes et associations définies pour chaque cas d'utilisation dans les DCL par cas d'utilisation.
Démarche d'élaboration	
<p>Pour chaque cas d'utilisation (CU) :</p> <ol style="list-style-type: none"> 1 Identifier les classes du CU en prenant comme base les objets définis dans le diagramme de séquence du CU et les concepts métiers. 2 Préciser les attributs des classes à partir de ceux identifiés dans le DSE (paramètres des messages) et des « entrées » de l'IHM. 3 Préciser les opérations des classes à partir des messages du DSE et des actions et activités du DET. Un message « entrant » d'un objet correspond à une opération de la classe sollicitée. 4 Déterminer les relations entre les classes : <ul style="list-style-type: none"> – nom de l'association, – multiplicité, – type d'association (composition, agrégation, association qualifiée, dépendance, héritage...). 	
<p>Recommandation :</p> <p>S'assurer que le DCL de chaque CU reste lisible pour les acteurs métiers au moins au niveau du nom des classes et des principales relations.</p>	

Synthèse de l'analyse

FICHE GUIDE – FG13	
Activité 4 : Synthèse de l'analyse Sous-activité 4.1 : Élaboration du diagramme de classe récapitulatif	
Projet :	
Date :	
Objectif	Regrouper l'ensemble des classes dans un seul diagramme pour avoir une vision globale du système étudié.
Point de départ	Tous les diagrammes de classe des cas d'utilisation étudiés.
Point d'arrivée	Regroupement des classes en un seul DCL.
Démarche d'élaboration	
<p>1 Récupérer l'ensemble des DCL de tous les CU.</p> <p>2 Mettre en commun, pour les mêmes classes, les attributs et opérations de chaque classe définis dans les DCL par CU. Chaque classe doit être décrite de manière exhaustive.</p> <p>3 Mettre en commun toutes les associations entre les classes définies dans les DCL par CU.</p> <p>4 Déterminer les relations entre différents DCL des CU. Mettre en place les nouvelles relations nécessaires dans le DCL récapitulatif.</p>	
Recommandation : S'assurer que le DCL récapitulatif reste lisible pour les acteurs métiers au moins au niveau du nom des classes et des principales relations. Établir, le cas échéant, un DCL récapitulatif à deux niveaux de lecture.	

FICHE GUIDE – FG14	
Activité 4 : Synthèse de l'analyse	
Sous-activité 4.2 : Élaboration de la matrice de validation	
Projet :	
Date :	
Objectif	Vérifier la complétude de l'analyse du système et établir la traçabilité entre les CU métiers (activité exigences fonctionnelles) et les CU d'analyse (activité analyse des CU).
Point de départ	Les CU métiers et d'analyse.
Point d'arrivée	Complétude de l'analyse du système vérifiée et traçabilité entre les CU effectuée.
Démarche d'élaboration	
<p>1 Reprendre les CU métiers et d'analyse.</p> <p>2 Établir une correspondance entre les CU métiers et les CU d'analyse <i>via</i> la matrice de validation. NB : des cas d'utilisation peuvent être présents uniquement dans l'activité d'analyse des CU car ils répondent à une problématique spécifique liée par exemple à la gestion des utilisateurs par un administrateur.</p> <p>3 Conclure sur la complétude de l'activité d'analyse des CU en s'assurant que tous les CU métier ont été repris dans l'activité d'analyse des CU.</p>	
Recommandation : Mettre en évidence (gras) dans la matrice, les CU qui n'ont pas pour origine un CU métier.	

Conception

FICHE GUIDE – FG15	
Activité 5 : Conception Sous-activité 5.1 : Réalisation des choix techniques	
Projet :	
Date :	
Objectif	Choisir l'architecture technique cible et les technologies associées.
Point de départ	Étapes d'expression des besoins et d'analyse terminées (Activités 1, 2, 3, 4).
Point d'arrivée	Architecture technique et technologies associées choisies.
Démarche d'élaboration	
<p>1 Choix de l'architecture technique en fonction des contraintes techniques imposées lors de l'expression des besoins (ex. contraintes de performances), du contexte client (environnement technique cible) et de l'état de l'art des technologies.</p> <p>2 Choix des technologies utilisées en fonction de l'architecture sélectionnée et des contraintes techniques.</p> <p>NB : nous donnons, après les fiches guides, une description complémentaire sur cette sous-activité.</p>	
<p>Recommandation : Ne pas négliger la prise en compte des exigences techniques (contraintes de performances) qui peuvent avoir une influence forte sur le choix d'une architecture.</p>	

FICHE GUIDE – FG16	
Activité 5 : Conception Sous-activité 5.2 : Élaboration des diagrammes de séquence technique	
Projet : Date :	
Objectif	Représenter les interactions entre les acteurs et tous les objets du système (incluant les objets techniques) pour tous les cas d'utilisation en considérant les différents scénarios associés. Représenter les objets en utilisant la classification d'Ivar Jacobson* (Dialogue, Contrôleur, Entité). * voir le paragraphe sur les compléments de la conception.
Point de départ	Architecture choisie, DSE et DCL de l'activité 3.
Point d'arrivée	Interactions entre les acteurs et tous les objets du système (incluant les objets techniques) définies pour tous les cas d'utilisation.
Démarche d'élaboration	
Pour chaque cas d'utilisation (CU) et chaque scénario identifié : 1 Reporter le ou les acteurs du DCU impliqués dans le scénario. 2 Représenter les objets du système impliqués dans le scénario en prenant comme base ceux identifiés lors du DSE de l'activité 3 : – Objets « Dialogue ». – Objets « Contrôleur ». – Objets « Entité ». – Objets techniques (Collection, Date...) identifiés uniquement lors de cette étape. 3 Représenter les messages (synchrone/asynchrone) entre les acteurs et les objets et entre les objets eux-mêmes. La chronologie des échanges doit être respectée. 4 Modéliser les fragments d'interaction si nécessaire.	
Recommandations : 1 Expliciter par des notes, les messages ou les fragments d'interactions complexes. 2 Utiliser des fragments d'interaction pour faciliter la lecture du DSE.	

FICHE GUIDE – FG17	
Activité 5 : Conception Sous-activité 5.3 : Élaboration des diagrammes de classe techniques	
Projet :	
Date :	
Objectif	Définir toutes les classes (incluant les classes techniques) et associations pour tous les cas d'utilisation. Représenter les classes en utilisant la classification d'Ivar Jacobson (Dialogue, Contrôleur, Entité).
Point de départ	Interactions entre les acteurs et tous les objets du système (incluant les objets techniques) définies pour tous les cas d'utilisation et DCL de l'activité 3.
Point d'arrivée	Toutes les classes (incluant les classes techniques) et associations définies pour tous les cas d'utilisation.
Démarche d'élaboration	
<p>Pour chaque cas d'utilisation (CU) :</p> <p>1 Représenter les classes du CU en prenant comme base les objets définis dans le diagramme de séquence technique du CU et le DCL de l'activité 3. Répartir les classes en plusieurs catégories :</p> <ul style="list-style-type: none"> – Classes « Dialogue ». – Classes « Contrôleur ». – Classes « Entité » (classes du DCL activité 3). – Classes techniques (Collection, Date...) identifiés uniquement lors de cette étape. <p>2 Préciser les attributs des classes et leurs caractéristiques (visibilité, type, valeur initiale...) à partir de ceux identifiés dans le DSE (paramètres des messages).</p> <p>3 Préciser les opérations des classes et leurs caractéristiques (paramètres avec type, type de résultat) à partir des messages du DSE.</p> <p>4 Déterminer les relations entre les classes :</p> <ul style="list-style-type: none"> – Nom de l'association. – Multiplicité. – Type d'association (composition, agrégation, association qualifiée, dépendance, héritage...). – Contraintes (ordonnées, non ordonnées...). 	
<p>Recommandation : Veiller à regrouper dans la modélisation les classes « dialogue », « contrôleur », et « entité » pour une meilleure lecture du diagramme.</p>	

FICHE GUIDE – FG18	
Activité 5 : Conception Sous-activité 5.4 : Élaboration du diagramme de paquetage	
Projet : Date :	
Objectif	Regrouper l'ensemble des classes en paquetage pour avoir une vision globale et structurée du système étudié.
Point de départ	Tous les diagrammes de classe par cas d'utilisation.
Point d'arrivée	Regroupement de l'ensemble des classes dans un seul diagramme de paquetage.
Démarche d'élaboration	
<p>1 Regrouper l'ensemble des classes par ensemble homogène. Chaque ensemble correspond à un paquetage (ex . Gestion Mail). L'ensemble peut correspondre à un découpage technique (architecture en couches) ou fonctionnel.</p> <p>2 Déterminer les dépendances entre les paquetages :</p> <ul style="list-style-type: none"> – « import » – « access » – ... 	
<p>Recommandations :</p> <p>1 Veiller à respecter deux principes pour le regroupement des classes en paquetage :</p> <ul style="list-style-type: none"> – Cohérence interne : constitution d'un ensemble homogène fonctionnel/technique. – Faible couplage externe. <p>2 Ne pas représenter sur le DPA les dépendances « import » par transitivité.</p>	

4.5.3 Compléments sur la conception

Deux points abordés dans les fiches guides sur la conception méritent un approfondissement :

- En premier lieu, la sous-activité « réalisation des choix techniques » effectuée au début de l'activité de conception pour déterminer l'architecture technique et les technologies associées. Cette sous-activité ne fait pas partie directement du champ d'UML et est fortement liée à l'évolution des technologies.
- En second lieu, la classification d'Ivar Jacobson qui permet de répartir les classes pour les diagrammes de l'activité de conception dans trois catégories (« dialogue », « contrôleur », et « entité »).

Réalisation de choix techniques

Les **choix techniques** portent sur une architecture technique et des technologies associées. Pour mieux comprendre les choix techniques à réaliser, nous allons prendre l'exemple du domaine du web. Pour cela, une présentation des principales architectures web est détaillée.

L'architecture web la plus utilisée aujourd'hui est l'architecture multitiers (n -tiers). Elle vient s'opposer aux architectures plus anciennes comme l'architecture mainframe et client/serveur par la répartition des couches applicatives.

La structuration des applications en couches permet :

- de maîtriser la complexité des applications (développement, échanges entre les applications et interactions entre objets) ;
- d'améliorer le découplage de l'application ;
- de diminuer les temps de développement, en factorisant certaines briques applicatives ;
- de favoriser la communication :
 - à l'intérieur d'une application, en structurant les échanges entre les différentes couches ;
 - entre les applications, en précisant les principes de communication liés aux couches de diverses applications.

Deux principales architectures n -tiers sont utilisées aujourd'hui.

L'architecture classique à trois couches

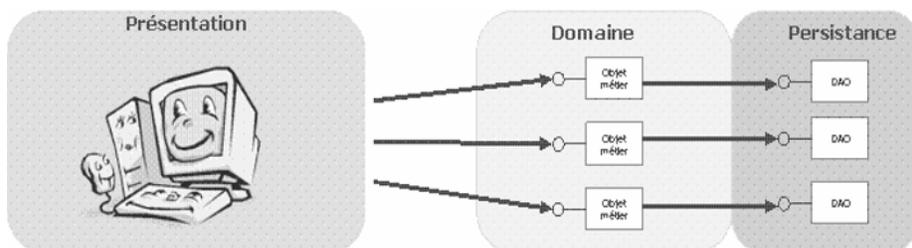


Figure 4.7 — Architecture classique à trois couches

Les trois couches suivantes sont présentes :

- **Couche présentation** – Cette couche contient l'interface homme-machine. C'est la couche de présentation des données à l'utilisateur. Elle contient uniquement les pages de mise en forme des données en vue de leur affichage.
- **Couche domaine** – Cette couche contient les objets du domaine (facture, bon de commande, client...). Ces objets encapsulent toutes les règles métier et appliquent la logique fonctionnelle de l'application.
- **Couche persistance** – Cette couche contient les usines d'objets métier, c'est-à-dire les classes chargées de créer des objets métier de manière totalement transparente, indépendamment de leur mode de stockage (DAO, objet de mapping...).

L'architecture orientée service (SOA)

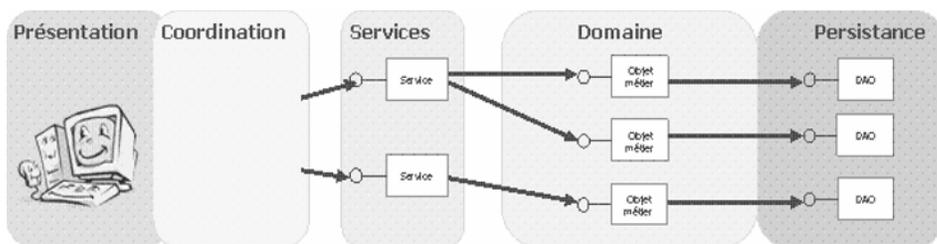


Figure 4.8 – Architecture orientée service (SOA)

Dans cette architecture, deux nouvelles couches apparaissent.

- **Couche coordination** – Cette couche gère l'organisation des données à afficher. C'est un contrôleur qui gère l'enchaînement des pages à afficher (Page Flow) en fonction des différentes demandes formulées par l'utilisateur.
- **Couche services** – Cette couche gère l'aspect SOA de l'application. Chaque demande de l'utilisateur correspond à un service appelé par cette couche, qui appelle les couches inférieures, et renvoie le résultat de son traitement à la couche supérieure. C'est également la couche service qui gère les transactions.

Cette architecture présente les spécificités suivantes par rapport aux architectures classiques :

- La couche coordination ne manipule plus directement les objets métiers, mais passe par des appels de services.
- Les objets métiers se trouvent dans des bibliothèques de classe directement chargées dans le même processus que les services ; le coût des appels aux objets métiers est alors très faible.
- Les services agissent comme des « boîtes noires » faisant abstraction de la complexité du modèle objet. Ces services présentent un ensemble de fonctionnalités restreintes et permettent de réduire les échanges entre les couches.

Concernant les technologies utilisées pour ces architectures, deux solutions dominent le marché : J2EE proposée par Sun et .NET proposée par Microsoft. Ces solutions ne seront pas détaillées dans cet ouvrage.

Classification d'Ivar Jacobson

La classification d'Ivar Jacobson permet d'identifier les classes « dialogue », « contrôleur » et « entité » dans un système :

- **Les classes « dialogue »** – Elles servent à modéliser les interactions entre le système et ses utilisateurs. Les paramètres saisis par les utilisateurs peuvent être représentés sous forme d'attributs de classe. Les actions proposées à l'utilisateur sur chaque page sont représentées sous forme d'opérations nommées par un verbe.
Elles ne peuvent être qu'associées uniquement aux classes « contrôleur » ou à d'autres classes « dialogue ».
- **Les classes « contrôleur »** – Elles sont utilisées pour représenter la coordination, l'enchaînement et le contrôle d'autres objets. Généralement, une seule classe « contrôleur » par cas d'utilisation. Mais sur le diagramme on peut montrer qu'un contrôleur appelle le contrôleur d'un autre cas d'utilisation. Il est possible de modéliser les opérations effectuées par le contrôleur et déclenchées par des actions au niveau des dialogues ou périodiquement.
Elles peuvent être associées à tous les types de classe.
- **Les classes « entité »** – Elles servent à modéliser des informations durables et souvent persistantes. Les classes « entité » sont issues des concepts métier du modèle de domaine ou bien sont nouvellement créées dans le diagramme si ce sont des entités purement « applicatives » ou techniques.
Les classes « entité » ne peuvent être qu'associées uniquement aux classes « contrôleur » ou à d'autres classes « entité ».

5

Étude de cas n° 1 Analyse

5.1 ÉNONCÉ DU CAS ALLOC

Chaque année, au troisième trimestre, les directeurs de laboratoire de recherche expriment leurs demandes de moyens pour l'année à venir auprès de leur direction scientifique. Une demande porte sur les moyens humains et sur les moyens financiers.

Chaque demande est étudiée par la direction scientifique à laquelle le laboratoire est rattaché.

Les propositions d'allocation de moyens des directions scientifiques font ensuite l'objet d'une consolidation générale par un coordonnateur afin de soumettre ces propositions à l'arbitrage de la direction générale. Cet ultime arbitrage permet d'aboutir à une décision définitive d'allocation de moyens aux laboratoires.

Chaque directeur scientifique notifie à ses laboratoires les décisions d'allocation de moyens pour l'année $n + 1$.

Dans le cadre de cette première étude cas, il est systématiquement indiqué, pour chaque diagramme ou activité à produire, le numéro de la fiche guide qui apporte un support méthodologique à la mise en œuvre de la démarche d'UML (UP7) préconisée dans cet ouvrage.

5.2 MODÉLISATION MÉTIER

5.2.1 Élaboration du schéma de contexte du domaine d'étude (FG1)

Conformément à notre démarche UP7, nous recommandons d'établir en premier un schéma de contexte permettant de situer le domaine d'étude par rapport aux autres processus de l'entreprise.

Ainsi, nous observons (fig. 5.1) que le domaine d'étude est en étroite relation avec deux processus importants traitant respectivement les ressources humaines et les moyens financiers.

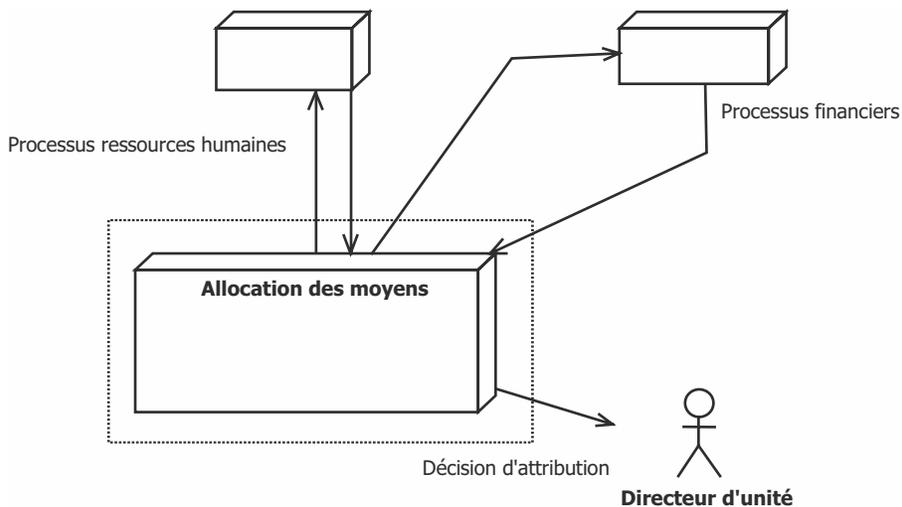


Figure 5.1 – Schéma de contexte du domaine d'étude

5.2.2 Élaboration du diagramme d'activité (FG2)

Quatre acteurs principaux interviennent dans les processus d'allocation des moyens (fig. 5.2) :

- **Le directeur de laboratoire (DU)** – C'est lui qui exprime la demande de moyens à sa direction scientifique.
- **Le directeur scientifique (DS)** – Il instruit la demande et élabore des propositions d'allocation de moyens.
- **Le coordonnateur (CO)** – Il saisit les cadrages des moyens à respecter par les DS et consolide les propositions faites par les DS avant de les soumettre à l'arbitrage du DG. Il saisit ensuite les éventuels ajustements des cadrages après les arbitrages.
- **La direction générale (DG)** – Elle arbitre définitivement les propositions d'allocation des moyens aux laboratoires après discussion avec les directeurs scientifiques.

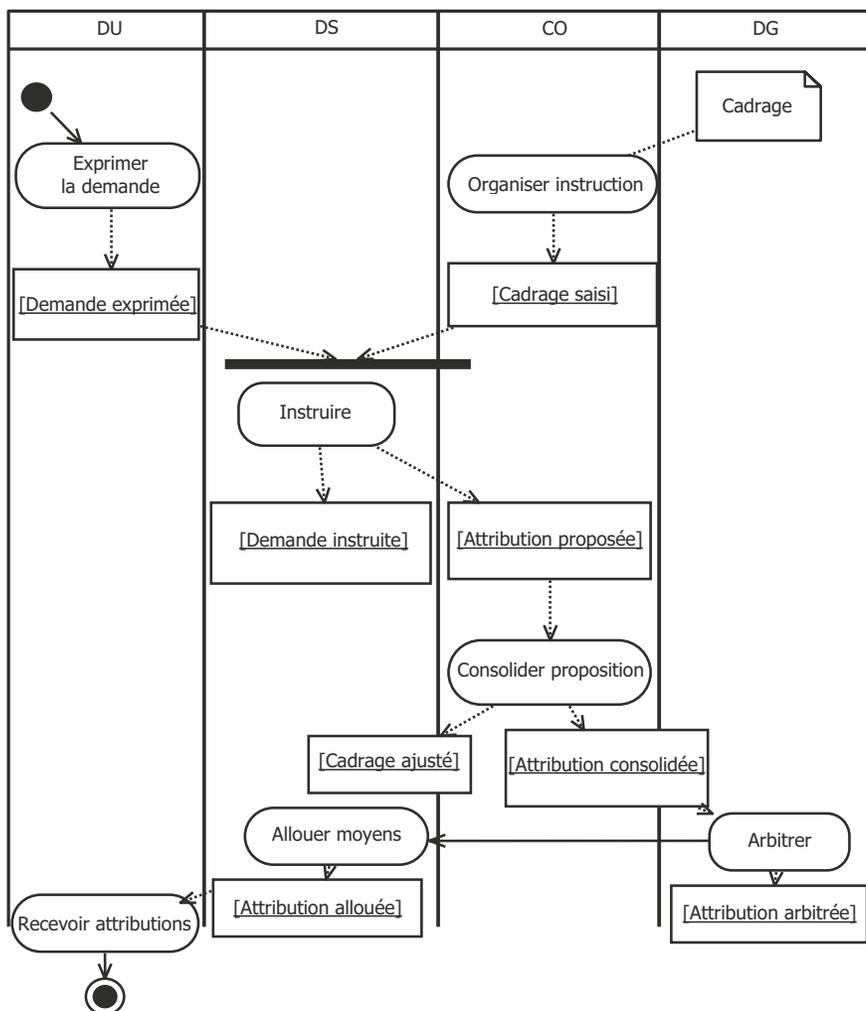


Figure 5.2 – Diagramme d'activité du domaine

5.2.3 Élaboration du diagramme de classe métier (FG3)

Les concepts métier pris en compte dans le diagramme de classe métier (fig. 5.3) sont :

- **Unité** – Laboratoire de recherche exprimant une demande annuelle de moyens.
- **Demande de l'unité** – Demande annuelle de moyens exprimée par le directeur de l'unité.
- **Attribution des moyens** – Attribution de moyens proposée par le DS et ensuite arbitrée par le DG.
- **Cadrage DS** – Enveloppe fixée par le DG pour chaque DS et type de moyens.
- **DS** – Département scientifique de rattachement de l'unité.

- **Histo-demandes** – Historique de toutes les demandes en ressources humaines ou en ressources financières exprimées par l’unité.
- **Histo-attributions** – Historique des attributions en ressources humaines ou en ressources financières faites à l’unité.

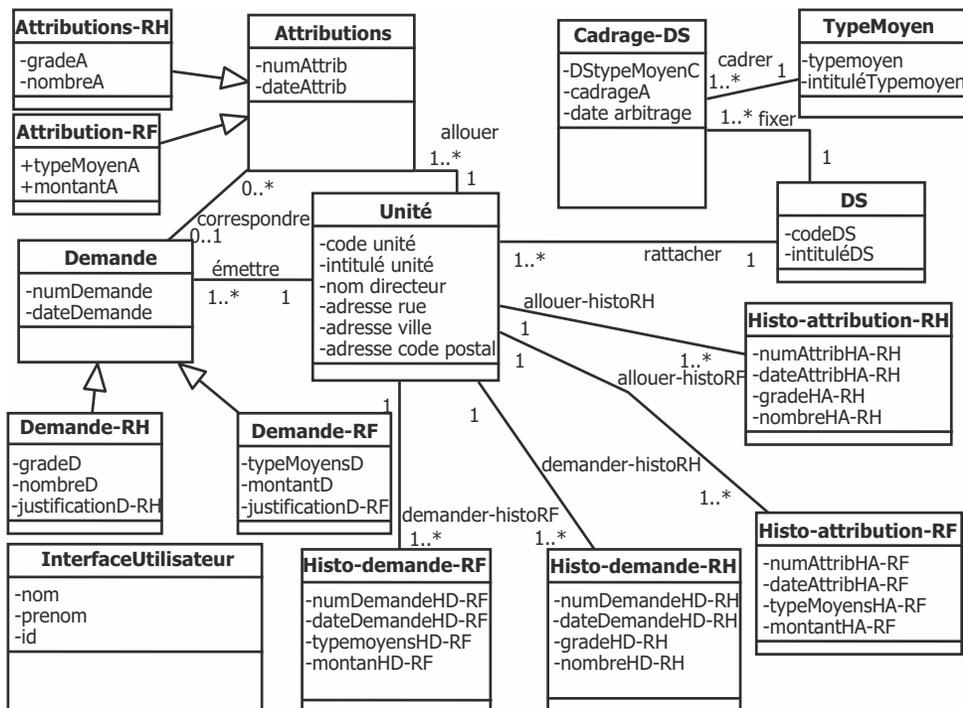


Figure 5.3 – Le diagramme de classe métier

5.2.4 Extrait des documents de cadrage

Des exemples de cadrage sont donnés ci-après.

Cadrage des moyens à allouer pour l’année <i>n</i>				
Départements scientifiques	Ressources humaines (RH)		Ressources financières (RF) en k€	
	Chercheurs	Ingénieurs	Budget de fonctionnement	Budget d’équipement
Chimie	12	15	21 000	4 000
Physique	8	7	12 000	3 000
Sciences de la vie	22	25	63 000	11 000
...				

Demande de moyens des unités pour l'année n				
Départements Chimie	Ressources humaines (RH)		Ressources financières (RF) en k€	
	Chercheurs	Ingénieurs	Budget de fonctionnement	Budget d'équipement
Unité 1	2	3	1 000	500
Unité 2	1	2	800	200
Unité 3	1	2	900	700

En résumé, les quatre types de moyen considérés dans cette étude de cas sont :

- RH-chercheurs,
- RH-ingénieurs,
- RF-budget,
- RF-équipement.

5.3 EXIGENCES FONCTIONNELLES

5.3.1 Élaboration du diagramme des cas d'utilisation système (FG4)

Représentation du DCU

À partir du diagramme d'activité et de la connaissance des besoins des acteurs, nous élaborons une vision générale des cas d'utilisation du futur système en produisant le DCU système (fig. 5.4).

Description générale des cas d'utilisation

- **Cas d'utilisation 0- « Saisir demande »** – Il s'agit de la saisie des données de la demande de moyens par les directeurs d'unités (DU). Cette saisie ne fait pas partie du champ d'étude du système car elle est prise en charge par un système d'information existant. Il convient seulement de prévoir une interface permettant de récupérer les informations après saisie.
- **Cas d'utilisation 1- « Gérer les cadrages »** – Chaque année des cadrages sont fixés par le DG pour chaque DS et chaque type de moyens. Ces cadrages sont saisis par le coordonnateur (CO). Après arbitrage par le DG, les cadrages peuvent éventuellement être ajustés.
- **Cas d'utilisation 2- « Éditer les fiches de demande »** – Après intégration des données saisies dans le système, celles-ci doivent pouvoir être consultées par les personnes qui sont chargées de leur exploitation. C'est l'édition des fiches de demande qui répondra à ce besoin.

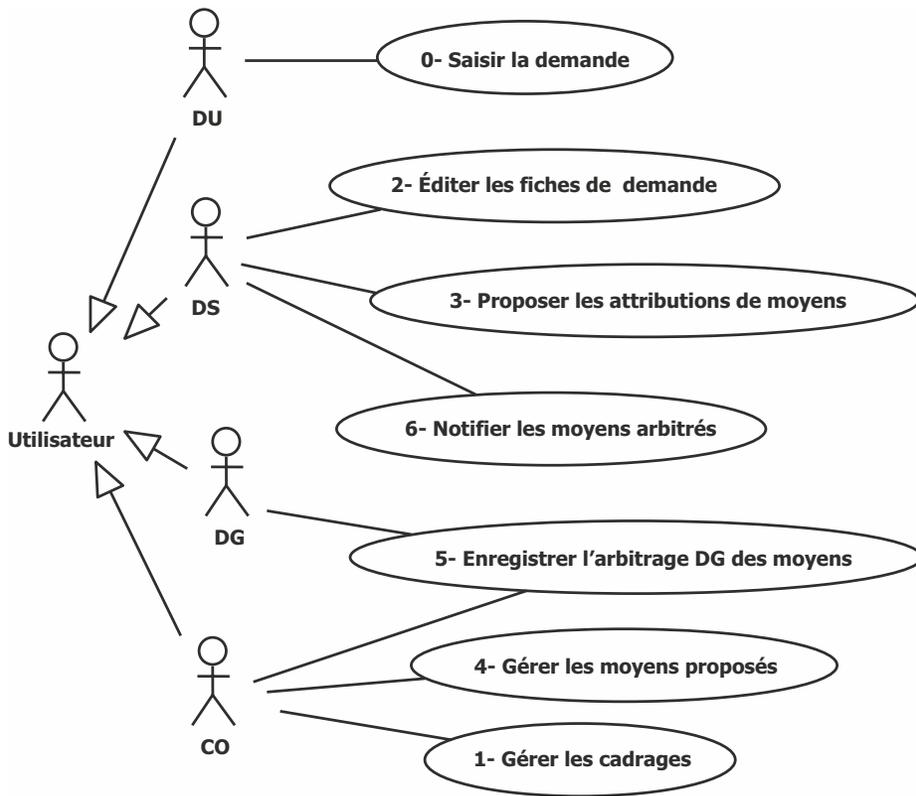


Figure 5.4 – Le diagramme des cas d'utilisation système

- **Cas d'utilisation 3-** « Proposer les attributions de moyens » – Après étude des demandes et compte tenu des moyens disponibles, les DS procèdent à l'attribution des moyens humains et financiers unité par unité. Ces attributions sont en fait à considérer comme des propositions tant que l'arbitrage de la direction générale n'a pas été rendu.
- **Cas d'utilisation 4-** « Gérer les moyens proposés » – La gestion des moyens consiste en la consolidation générale des moyens à attribuer et à la production de tableaux de bord de suivi.
- **Cas d'utilisation 5-** « Enregistrer l'arbitrage DG des propositions » – Un certain nombre de moyens ne peuvent être attribués que si le directeur général a donné son accord. Ce dernier doit être enregistré dans le système par le coordonnateur.
- **Cas d'utilisation 6-** « Notifier les moyens arbitrés » – Les moyens arbitrés doivent être communiqués aux unités à l'aide de courriers produits automatiquement.

5.3.2 Élaboration du diagramme de séquence système (FG5)

Au stade de la description du niveau métier, il est possible de donner une première représentation des diagrammes de séquence (DSE) en considérant les interactions entre les acteurs et le système pris dans son ensemble. Ainsi, nous établissons :

- Le DSE du cas d'utilisation 1 « Gérer les cadrages » (fig. 5.5).
- Le DSE du cas d'utilisation 2 « Éditer les fiches de demande » (fig. 5.6).
- Le DSE du cas d'utilisation 3 « Proposer les attributions de moyens » (fig. 5.7).
- Le DSE du cas d'utilisation 4 « Gérer les moyens proposés » (fig. 5.8).
- Le DSE du cas d'utilisation 5 « Enregistrer l'arbitrage DG des propositions » (fig. 5.9).
- Le DSE du cas d'utilisation 6 « Notifier les moyens arbitrés » (fig. 5.10).

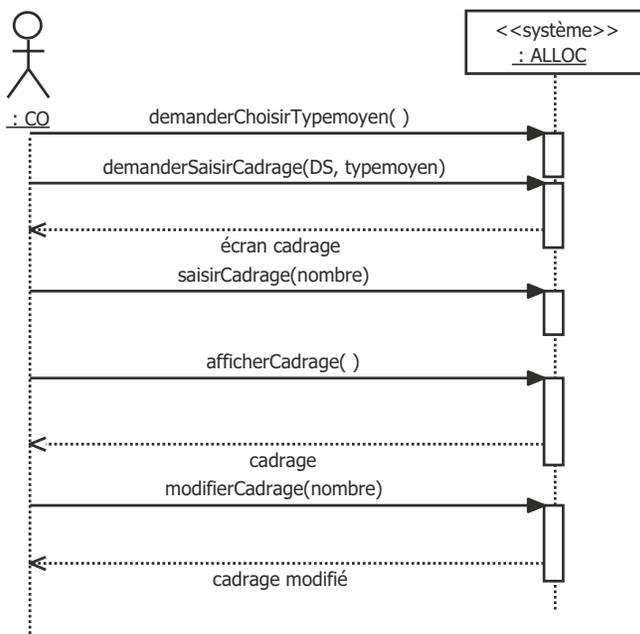


Figure 5.5 — DSE du cas d'utilisation 1- « Gérer les cadrages »

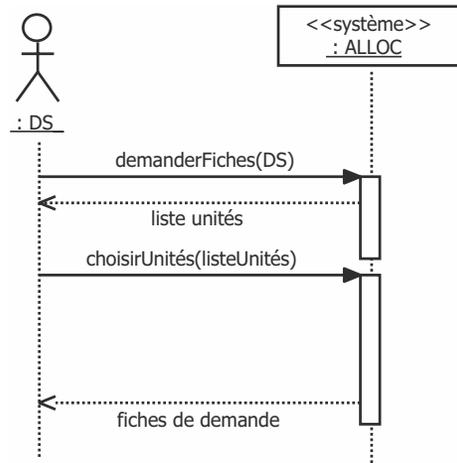


Figure 5.6 – DSE du cas d'utilisation 2- « Éditer les fiches de demande »

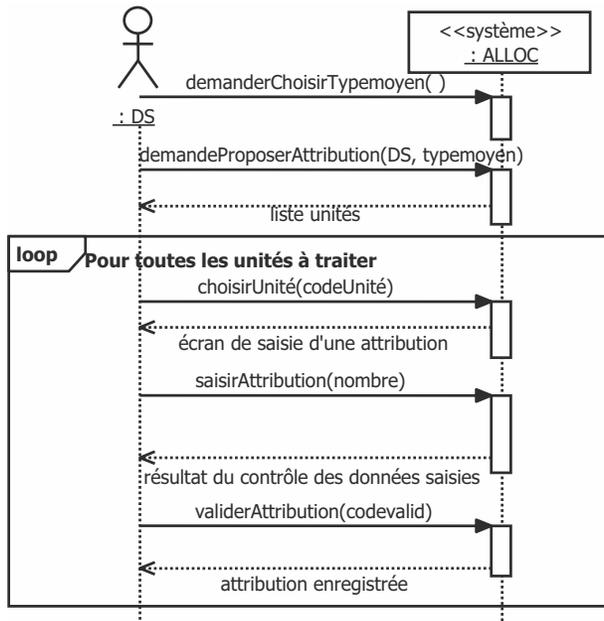


Figure 5.7 – DSE du cas d'utilisation 3- « Proposer les attributions de moyens »

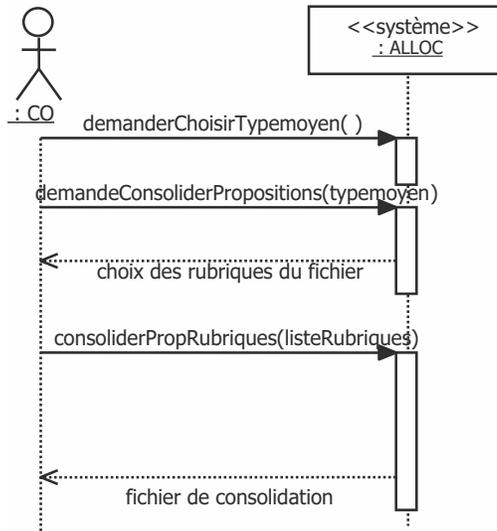


Figure 5.8 — DSE du cas d'utilisation
4- « Gérer les moyens proposés »

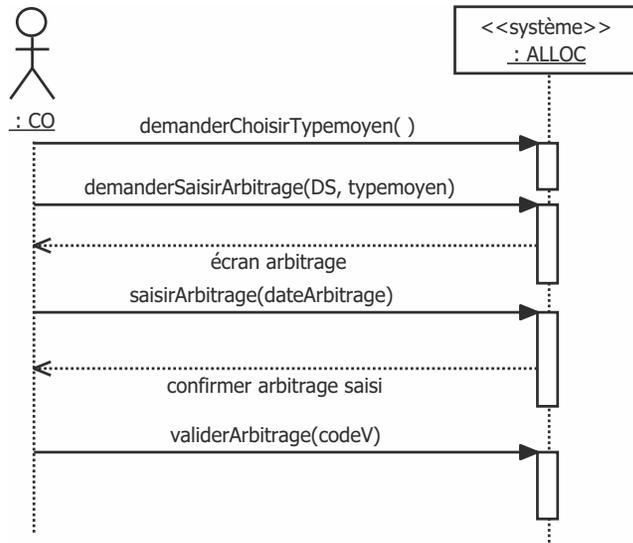


Figure 5.9 — DSE du cas d'utilisation
5- « Enregistrer l'arbitrage DG des propositions »

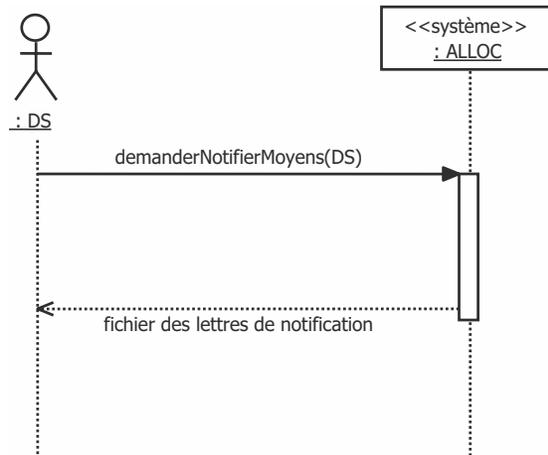


Figure 5.10 — DSE du cas d'utilisation
6- « Notifier les moyens arbitrés »

5.3.3 Élaboration du schéma de navigation générale (FG6)

L'enchaînement global des écrans peut être donné à ce stade (fig. 5.11).

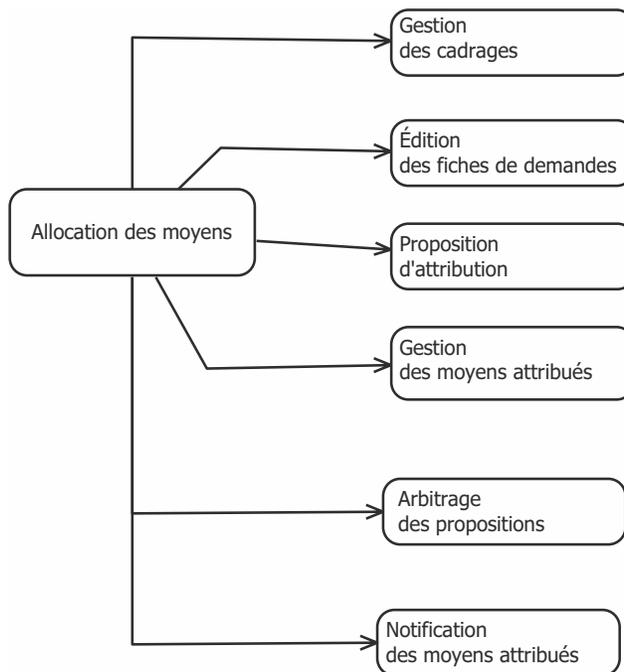


Figure 5.11 — Schéma de navigation générale

5.4 ANALYSE DES CAS D'UTILISATION

5.4.1 Élaboration du diagramme des cas d'utilisation (FG7)

À partir du premier DCU élaboré dans la partie « exigences fonctionnelles », il est possible d'affiner maintenant l'analyse des différents cas. Cette analyse conduit à ajouter deux cas d'utilisation.

Choisir type de moyens

Ce cas permet de décrire une seule fois les actions liées au choix d'un type de moyens et de proposer aux autres cas d'y recourir avec la fonction « include » si besoin.

Suivre l'avancement des attributions

Ce cas est en fait une extension du cas n° 4 avec l'utilisation de la fonction « extend ». Il permet au coordonnateur de disposer, quand cela est utile, des tableaux de suivi des attributions. Le diagramme complet des cas d'utilisation est donné à la figure 5.12.

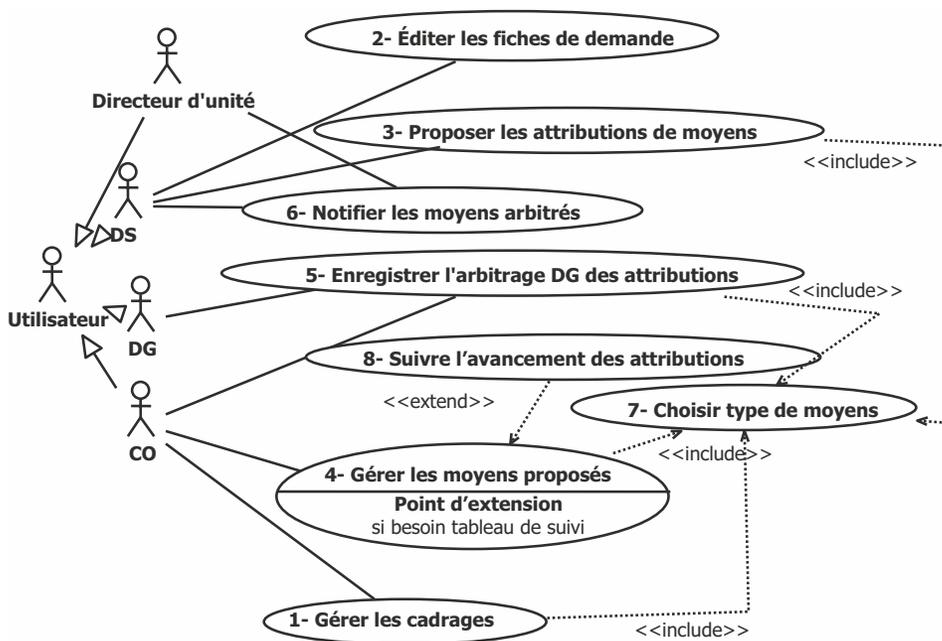


Figure 5.12 — Diagramme des cas d'utilisation

5.4.2 Description des cas d'utilisation (FG8, FG9, FG11, FG12)

Pour la suite de l'étude de cas, nous allons produire l'analyse des huit cas d'utilisation :

- Cas 1- « Gérer les cadrages »
- Cas 2- « Éditer les fiches de demande »

- Cas 3- « Proposer les attributions de moyens »
- Cas 4- « Gérer les moyens proposés »
- Cas 5- « Enregistrer l'arbitrage DG des propositions »
- Cas 6- « Notifier les moyens arbitrés »
- Cas 7- « Choisir type de moyens »
- Cas 8- « Suivre l'avancement des attributions »

Pour chaque cas d'utilisation, les sous-activités suivantes de l'activité « Analyse des cas d'utilisation » sont réalisées :

- Description (textuelle) du cas d'utilisation (FG8)
- Élaboration du diagramme de séquence (FG9)
- Élaboration de l'interface utilisateur (FG11)
- Élaboration du diagramme de classe (FG12)

Cas d'utilisation 1- « Gérer les cadrages »

Description textuelle du cas d'utilisation

- **Objectif** – Permettre au coordonnateur de saisir, de consulter ou de modifier des données de cadrage pour un type de moyens.
- **Acteur concerné** – Coordonnateur.
- **Pré condition** – Aucune.
- **Scénario nominal : saisie d'un nouveau cadrage**
 - 1 Le coordonnateur choisit un type de moyen pour un DS donné.
 - 2 Le coordonnateur renseigne les données de cadrage.
 - 3 Le système vérifie la présence des données obligatoires.
 - 4 Le système affiche les données à enregistrer pour validation.
 - 5 Le système enregistre la saisie validée.
- **Scénarios alternatifs**
 - 2-a Modification des données de cadrage :**
 - Le système affiche le formulaire de saisie des données de cadrage enregistrées.
 - Le coordonnateur modifie les données.
 - Le cas d'utilisation reprend à l'action 3 du scénario nominal.
 - 2-b Consultation des données de cadrage :**
 - Le système affiche les données de cadrage déjà enregistrées.
 - Fin du cas d'utilisation.
 - 4-a Erreurs détectées dans la saisie :**
 - Le système réaffiche le formulaire de saisie en indiquant les erreurs détectées.
 - Le coordonnateur corrige les erreurs.
 - Le cas d'utilisation reprend au point 3 du scénario nominal.

Description des diagrammes d'analyse du cas d'utilisation

La suite de l'analyse du cas d'utilisation se poursuit par l'élaboration du diagramme de séquence (fig. 5.13), l'élaboration de l'interface utilisateur (tab. 5.1) et l'élaboration du diagramme de classe (fig. 5.14).

Dans le DSE, les cas d'erreurs ainsi que la recherche des intitulés DS et type de moyen n'ont pas été représentés.

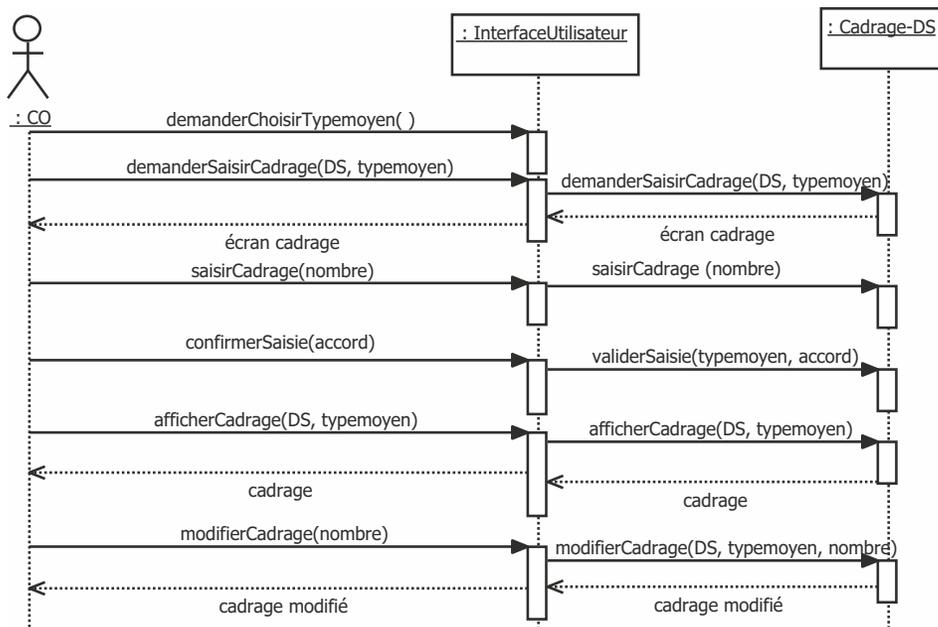


Figure 5.13 – Diagramme de séquence du cas d'utilisation 1- « Gérer les cadrages »

Tableau 5.1 – Données de l'interface utilisateur du cas d'utilisation 1

Données affichées	Données saisies
- Code et intitulé DS	- DS et typemoyen
- Code et intitulé du type de moyen à traiter	- Nombre correspondant au cadrage du type de moyen sélectionné
- Nombre correspondant au cadrage du type de moyen sélectionné	- Validation

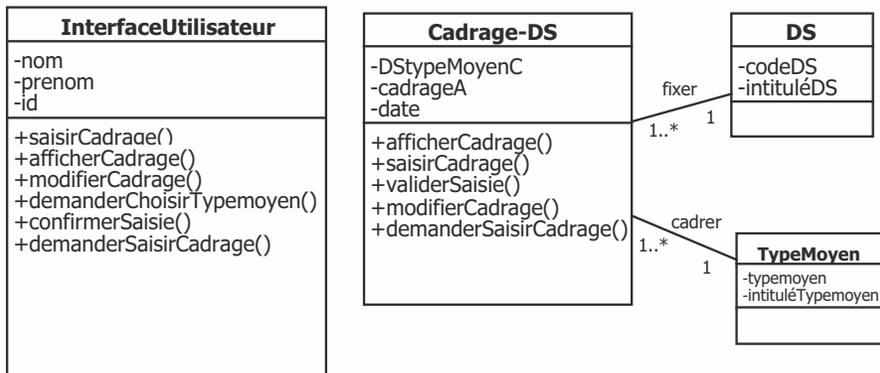


Figure 5.14 – Diagramme de classe du cas d'utilisation 1

Cas d'utilisation 2- « Éditer les fiches de demande »

Description textuelle du cas d'utilisation

- **Objectif** – Permettre aux départements scientifiques de produire les fiches de demande de moyens.
Une fiche de demande (FD) présente un ensemble d'information concernant une unité donnée. Elle regroupe l'ensemble de ses demandes pour l'année $N + 1$. Un rappel de sa situation à l'année N est aussi indiqué (en demande de moyens et moyens attribués).
- **Acteur concerné** – DS.
- **Pré condition** – Toutes les unités du département ont exprimé leur demande.
- **Scénario nominal**
 - Le DS demande l'édition de fiches.
 - Le système affiche les unités du DS.
 - Le DS sélectionne les unités concernées.
 - Le système construit et affiche le contenu de la fiche pour les unités sélectionnées.

Description des diagrammes d'analyse du cas d'utilisation

La suite de l'analyse du cas d'utilisation se poursuit par l'élaboration du diagramme de séquence (fig. 5.15), l'élaboration de l'interface utilisateur (tab. 5.2) et l'élaboration du diagramme de classe (fig. 5.16).

Cas d'utilisation 3- « Proposer les attributions de moyens »

Description textuelle du cas d'utilisation

- **Objectif** – Permettre au DS de saisir ou de consulter des propositions d'attributions.
- **Acteur concerné** – DS.
- **Pré condition** – Aucune.

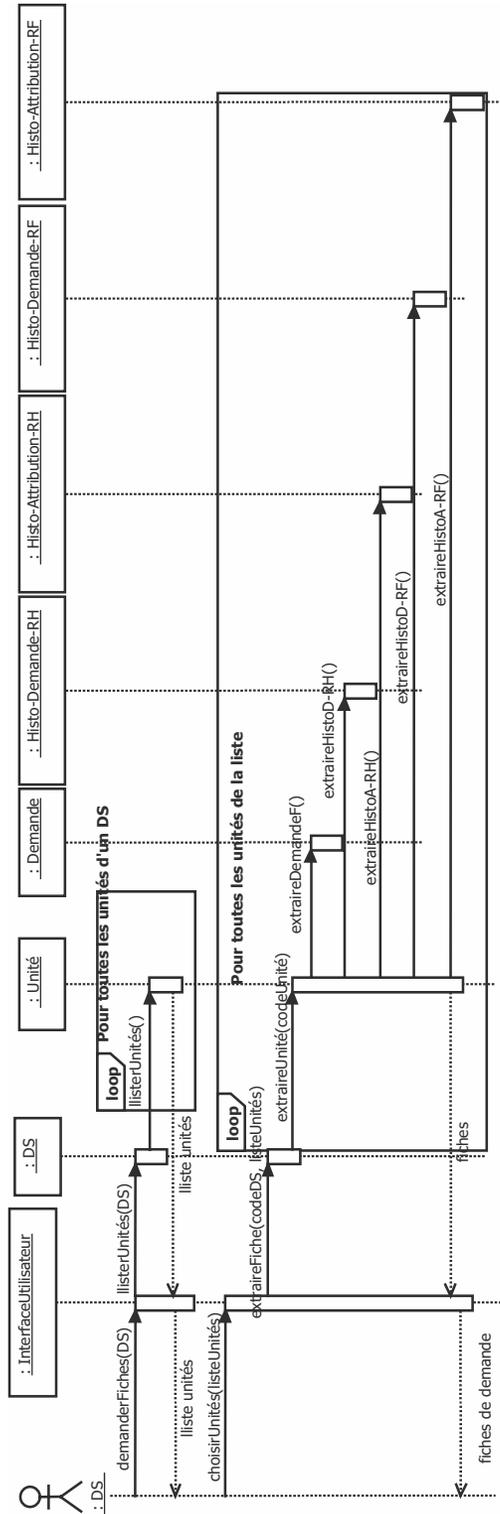
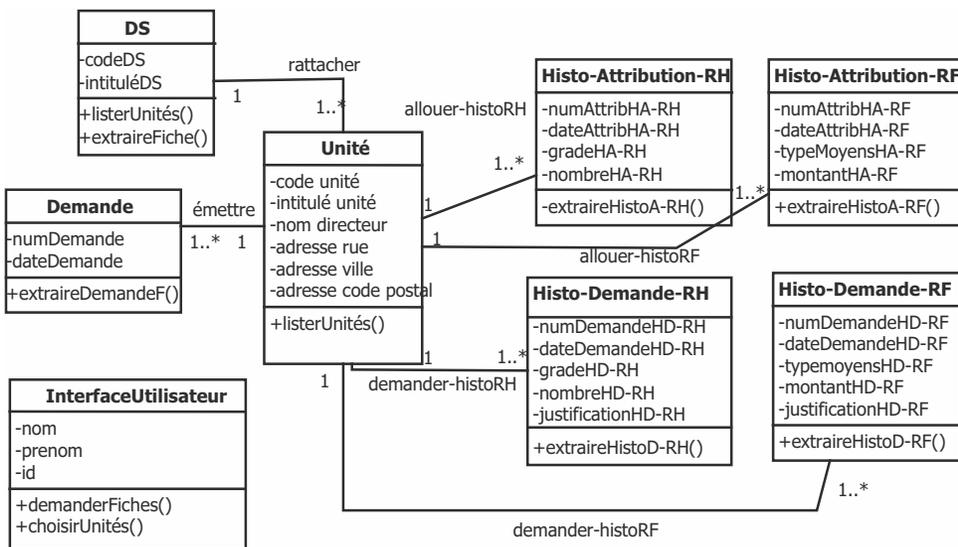


Figure 5.15 — Diagramme de séquence du cas d'utilisation 2

Tableau 5.2 — Données de l'interface utilisateur du cas d'utilisation 2

Données affichées	Données saisies
- Code et intitulé DS	- DS
- Liste des unités du DS	- Codes unités choisies
Pour chaque unité choisie : - Nom du directeur - Adresse - Numéro de demande - Date de la demande - Demandes RH et RF - Historique demandes RH et RF - Historique attributions RH et RF	

**Figure 5.16** — Diagramme de classe du cas d'utilisation 2

- Scénario nominal

- 1 Le DS choisit le type de moyen à traiter.
- 2 Le système affiche la liste des unités du DS.
- 3 Le DS choisit l'unité pour laquelle il veut faire une proposition d'attribution.
- 4 Le système affiche le formulaire de saisie des propositions d'attribution pré rempli avec les demandes du type de moyen sélectionné effectuées par l'unité.
- 5 Le DS renseigne les données de la proposition d'attribution.
- 6 Le système vérifie la présence des données obligatoires.
- 7 Le système enregistre la saisie et affiche les attributions mises à jour.

- Scénarios alternatifs

- 4-a Saisie d'une proposition d'attribution sans demande préalable

- Le système affiche le formulaire de saisie des propositions d'attribution vierge.
 - Le cas d'utilisation reprend à l'action 5 du scénario nominal.

- 4-b Modification d'une proposition d'attribution saisie

- Le système affiche le formulaire de saisie des propositions d'attribution avec les demandes et les propositions d'attribution de l'unité pré-remplies.
 - Le cas d'utilisation reprend à l'action 5 du scénario nominal.

- 4-c Consultation des propositions d'attribution

- Le système affiche les propositions d'attribution de l'unité sélectionnée avec les demandes de moyens associées.
 - Fin du cas.

- 7-a Erreurs détectées dans la saisie

- Le système réaffiche le formulaire de saisie en indiquant les erreurs détectées.
 - L'instructeur corrige les erreurs.
 - Le cas d'utilisation reprend au point 6 du scénario nominal.

Description des diagrammes d'analyse du cas d'utilisation

La suite de l'analyse du cas d'utilisation se poursuit par l'élaboration du diagramme de séquence (fig. 5.17), l'élaboration de l'interface utilisateur (tab. 5.3) et l'élaboration du diagramme de classe (fig. 5.18).

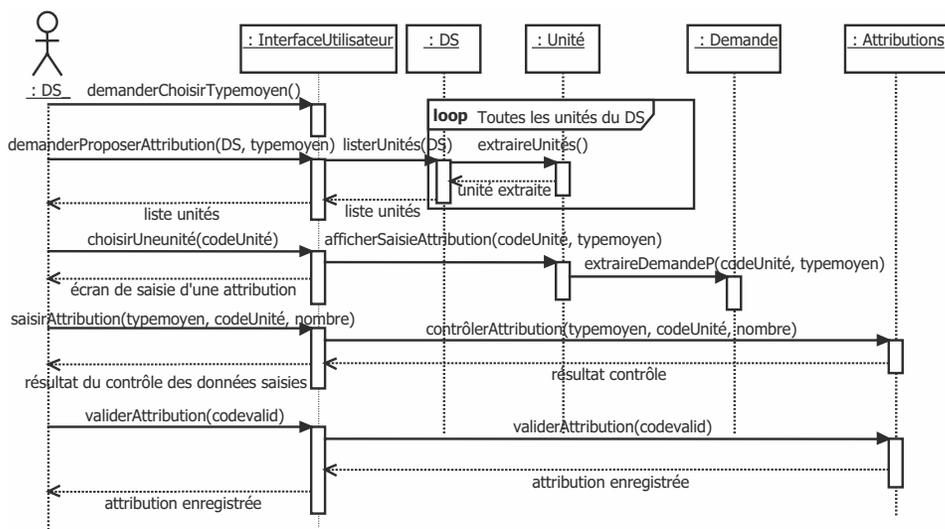
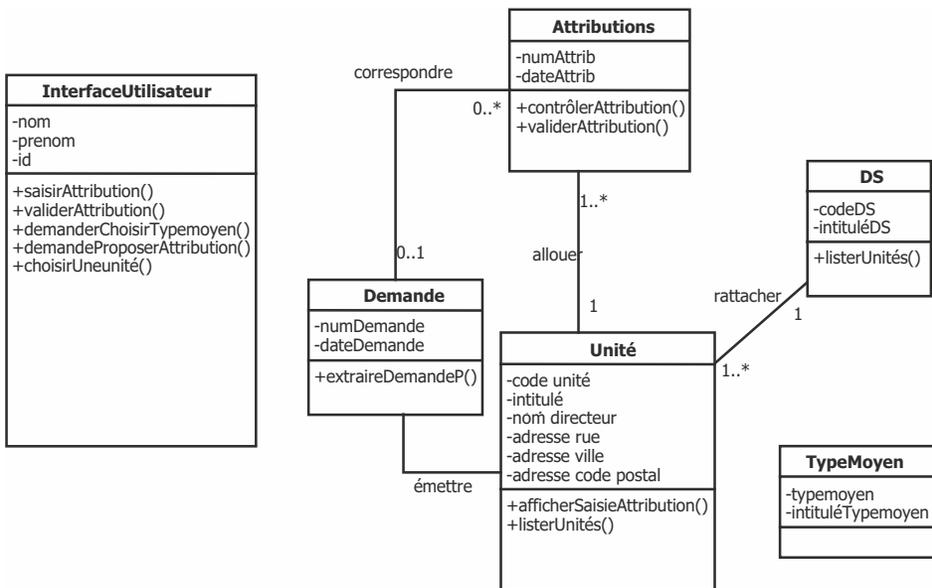


Figure 5.17 – Diagramme de séquence du cas d'utilisation 3

Dans le DSE, seul le scénario nominal est représenté pour le traitement d'une unité. La recherche de l'intitulé DS et type de moyen n'a pas été traitée.

Tableau 5.3 – Données de l'interface utilisateur du cas d'utilisation 3

Données affichées	Données saisies
- Code et intitulé DS	- DS et typemoyen
- Code et intitulé du type de moyen à traiter	- Code de l'unité à traiter
- Demande RH - Demande RF	- Nombre proposé pour le type de moyens traité
- Attribution proposée RH - Attribution proposée RF	- Validation de la saisie

**Figure 5.18** – Diagramme de classe du cas d'utilisation 3

Cas d'utilisation 4- « Gérer les moyens proposés »

Description textuelle du cas d'utilisation

- **Objectif** – Permettre au coordonnateur d'exporter le fichier contenant les éléments relatifs aux demandes et aux propositions d'attribution de moyens pour l'élaboration des documents présentés au DG en vue de leur arbitrage.
- **Acteur concerné** – Coordonnateur.
- **Pré condition** – Aucune.
- **Scénario nominal**
 - 1 Le coordonnateur choisit le type de moyen à traiter.
 - 2 Le coordonnateur demande à extraire le fichier pour la consolidation des propositions d'attribution.

- 3 Le système liste les différentes rubriques des propositions d'attribution dans le fichier
- 4 Le coordonnateur sélectionne les rubriques souhaitées.
- 5 Le système génère le fichier de consolidation des propositions d'attribution pour toutes les unités.

Description des diagrammes d'analyse du cas d'utilisation

La suite de l'analyse du cas d'utilisation se poursuit par l'élaboration du diagramme de séquence (fig. 5.19), l'élaboration de l'interface utilisateur (tab. 5.4) et l'élaboration du diagramme de classe (fig. 5.20).

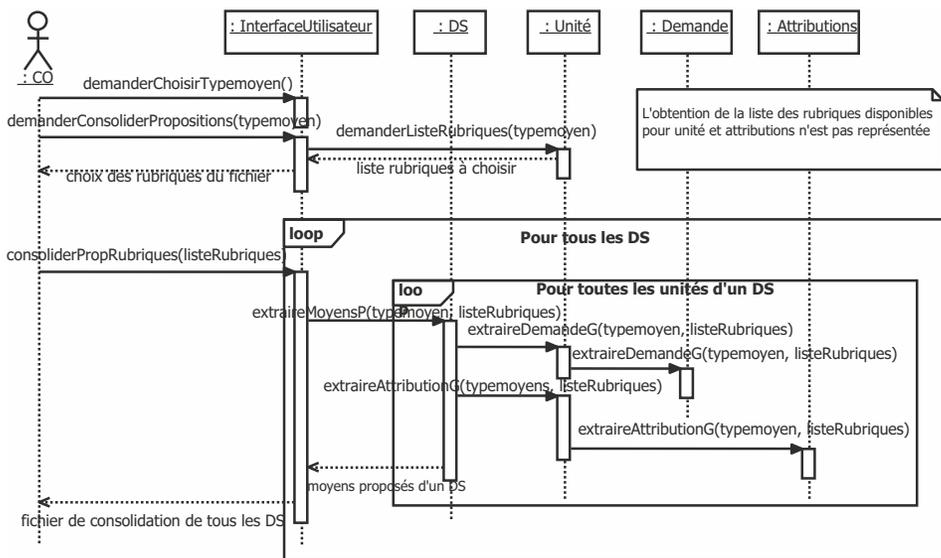


Figure 5.19 – Diagramme de séquence du cas d'utilisation 4

Tableau 5.4 – Données de l'interface utilisateur du cas d'utilisation 4

Données affichées	Données saisies
- Code et intitulé type moyen	- Type moyen
- Liste des rubriques du fichier	- Liste des rubriques choisies
- Nombre correspondant aux demandes et aux moyens proposés	

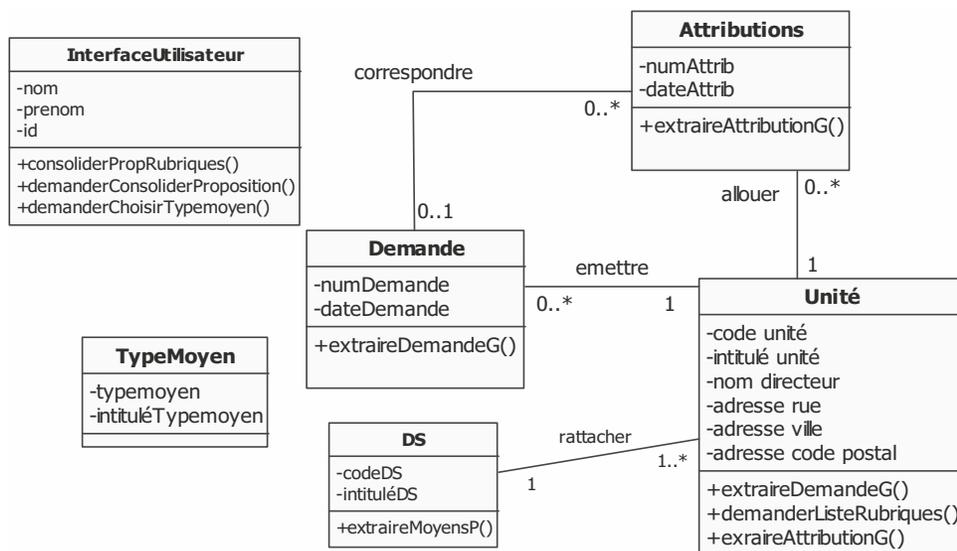


Figure 5.20 – Diagramme de classe du cas d'utilisation 4

Cas d'utilisation 5- « Enregistrer l'arbitrage DG des propositions »

Description textuelle du cas d'utilisation

- **Objectif** – Permettre au coordonnateur de saisir l'arbitrage de la direction.
- **Acteur concerné** – Coordonnateur.
- **Pré condition** – RAS
- **Scénario nominal**
 - 1 Le coordonnateur choisit un type de moyen pour un DS donné.
 - 2 Le système affiche le formulaire de saisie des états d'arbitrage des types de moyen pré rempli.
 - 3 Le coordonnateur renseigne la date d'arbitrage.
 - 4 Le système vérifie la conformité des données saisies.
 - 5 Le système demande la validation des données saisies.
 - 6 Le système enregistre la saisie, après validation, et affiche le résultat de la mise à jour.
- **Scénarios alternatifs**
 - 5-a **Erreurs détectées dans la saisie** :
 - Le système réaffiche le formulaire de saisie en indiquant les erreurs détectées.
 - Le coordonnateur corrige les erreurs.
 - Le cas d'utilisation reprend au point 4 du scénario nominal.

Description des diagrammes d'analyse du cas d'utilisation

La suite de l'analyse du cas d'utilisation se poursuit par l'élaboration du diagramme de séquence (fig. 5.21), l'élaboration de l'interface utilisateur (tab. 5.5) et l'élaboration du diagramme de classe (fig. 5.22).

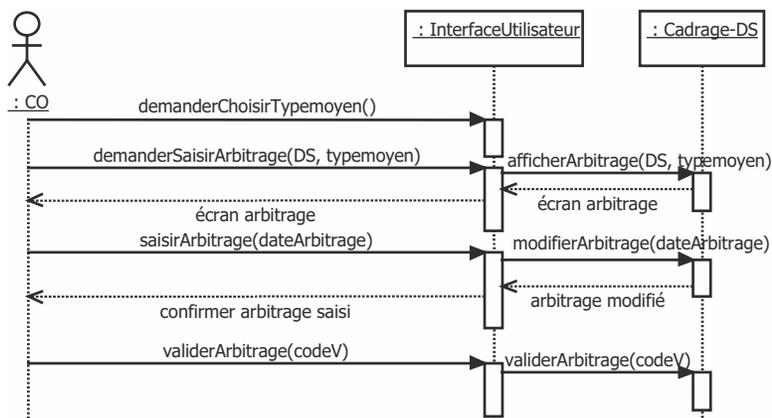


Figure 5.21 – Diagramme de séquence du cas d'utilisation 5

Seul le scénario nominal est représenté dans le DSE. La recherche de l'intitulé type moyen n'est pas aussi représentée.

Tableau 5.5 – Données de l'interface utilisateur du cas d'utilisation 5

Données affichées	Données saisies
- Code et intitulé du type de moyen à traiter	- DS et type de moyen à traiter
- Date de l'arbitrage	- Date d'arbitrage - Code validation

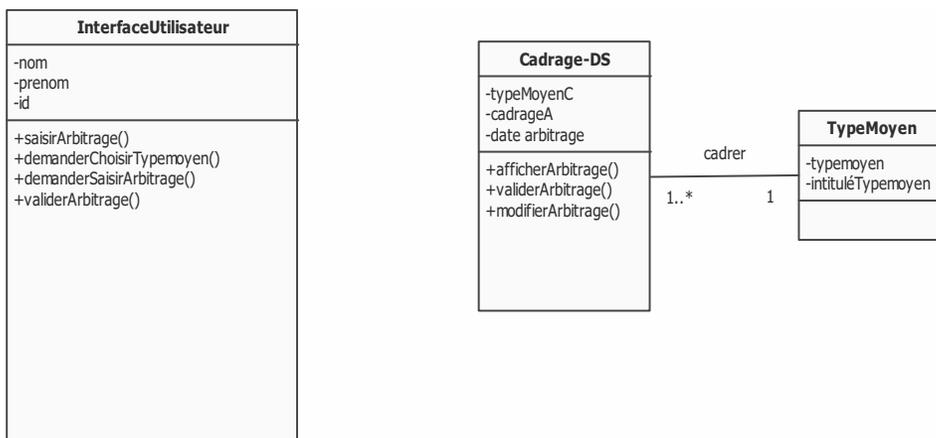


Figure 5.22 – Diagramme de classe du cas d'utilisation 5

Cas d'utilisation 6- « Notifier les moyens arbitrés »

Description textuelle du cas d'utilisation

- **Objectif** – Permettre au DS de produire les lettres informant les directeurs d'unité des moyens qui leur sont alloués.
- **Acteur concerné** – DS.
- **Pré condition** – Aucune.
- **Scénario nominal**
 - 1 Le DS demande l'extraction du fichier pour l'édition des lettres type d'attribution de moyens.
 - 2 Le système génère le fichier des lettres de notification.

Description des diagrammes d'analyse du cas d'utilisation

La suite de l'analyse du cas d'utilisation se poursuit par l'élaboration du diagramme de séquence (fig. 5.23), l'élaboration de l'interface utilisateur (tab. 5.6) et l'élaboration du diagramme de classe (fig. 5.24).

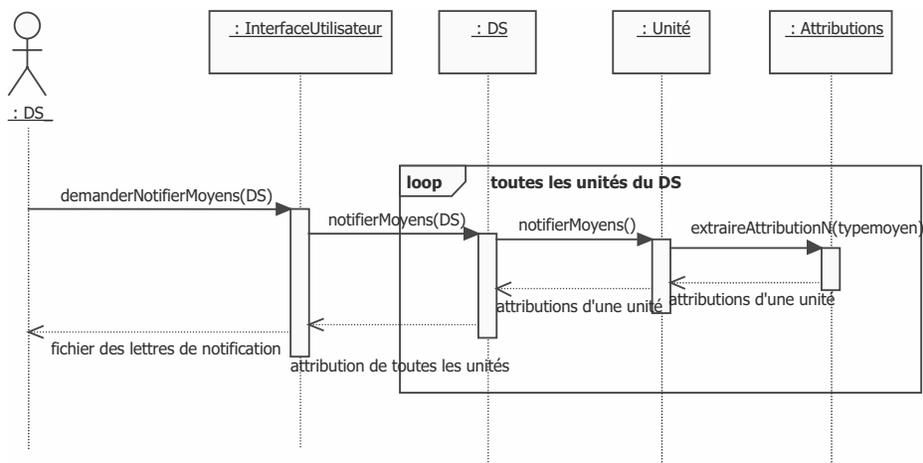


Figure 5.23 – Diagramme de séquence du cas d'utilisation 6

Tableau 5.6 – Données de l'interface utilisateur du cas d'utilisation 6

Données affichées	Données saisies
- Code et intitulé du DS	- Code du DS
Pour chaque unité concernée : - Code - Intitulé - Adresse - Nom du directeur	
- Type moyen et nombre correspondant au moyen attribué (pour toutes les attributions).	

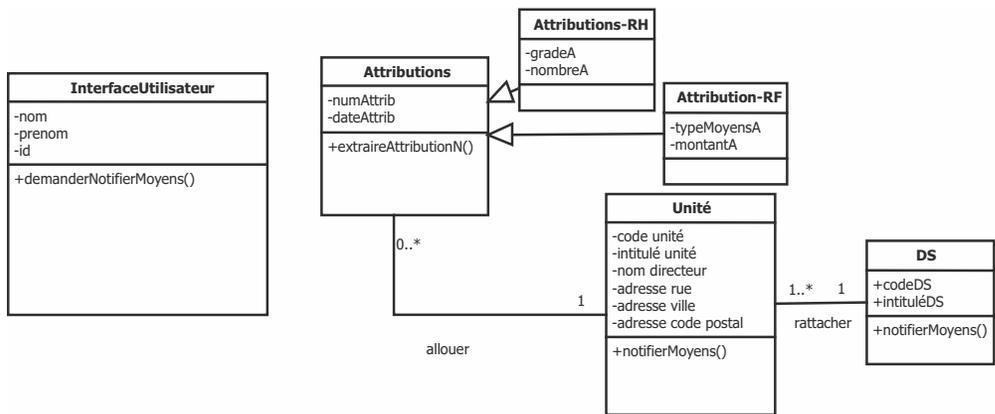


Figure 5.24 – Diagramme de classe du cas d'utilisation 6

Cas d'utilisation 7- « Choisir un type de moyen »

Description textuelle du cas d'utilisation

- **Objectif** – Permettre aux acteurs de choisir le type de moyen qu'ils veulent traiter.
- **Acteurs concernés** – Instructeur DS ou DG ou coordonnateur.
- **Pré condition** – RAS.
- **Scénario nominal**
 - 1 Le système affiche la liste des types de moyen.
 - 2 L'acteur concerné choisit le type de moyen qu'il veut traiter.
- **Scénarios alternatifs** – Aucun.

Description des diagrammes d'analyse du cas d'utilisation

La suite de l'analyse du cas d'utilisation se poursuit par l'élaboration du diagramme de séquence (fig. 5.25), l'élaboration de l'interface utilisateur (tab. 5.7) et l'élaboration du diagramme de classe (fig. 5.26).

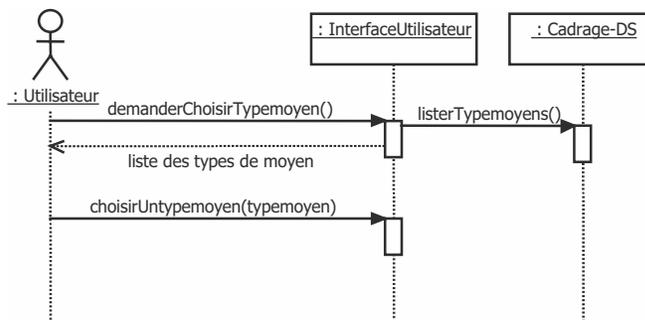
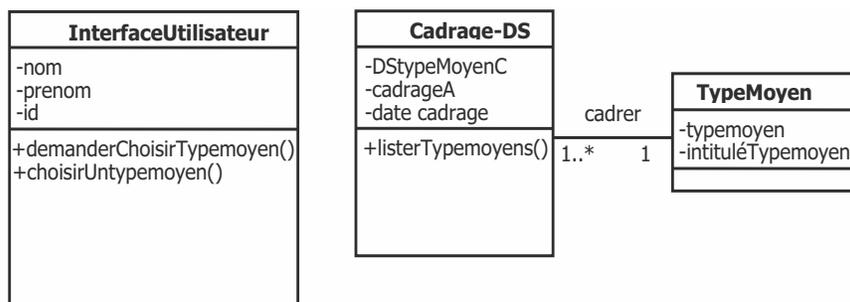


Figure 5.25 – Diagramme de séquence du cas d'utilisation 7

Tableau 5.7 – Données de l'interface utilisateur du cas d'utilisation 7

Données affichées	Données saisies
- Liste des types de moyen proposés	- Code du type de moyen choisi

**Figure 5.26** – Diagramme de classe du cas d'utilisation 7

Cas d'utilisation 8 « Suivre l'avancement des attributions »

Description textuelle du cas d'utilisation

- **Objectif** – Mettre à la disposition des départements scientifiques un ensemble de tableaux de suivi des attributions des moyens aux unités.
- **Acteur concerné** – Coordonnateur.
- **Pré condition** – Être à l'intérieur de l'exécution du cas d'utilisation n° 4 : Gérer les moyens arbitrés.
- **Scénario nominal**
 - 1 Le système affiche la liste des tableaux de suivi.
 - 2 Le coordonnateur saisit le choix correspondant au tableau souhaité.
 - 3 Le système produit le tableau de suivi demandé.

Description des diagrammes d'analyse du cas d'utilisation

La suite de l'analyse du cas d'utilisation se poursuit par l'élaboration du diagramme de séquence (fig. 5.27), l'élaboration de l'interface utilisateur (tab. 5.8) et l'élaboration du diagramme de classe (fig. 5.28).

5.5 SYNTHÈSE DE L'ANALYSE

Le diagramme de classe récapitulatif (FG13) de la figure 5.29 intègre l'ensemble des diagrammes de classe élaborés par cas d'utilisation.

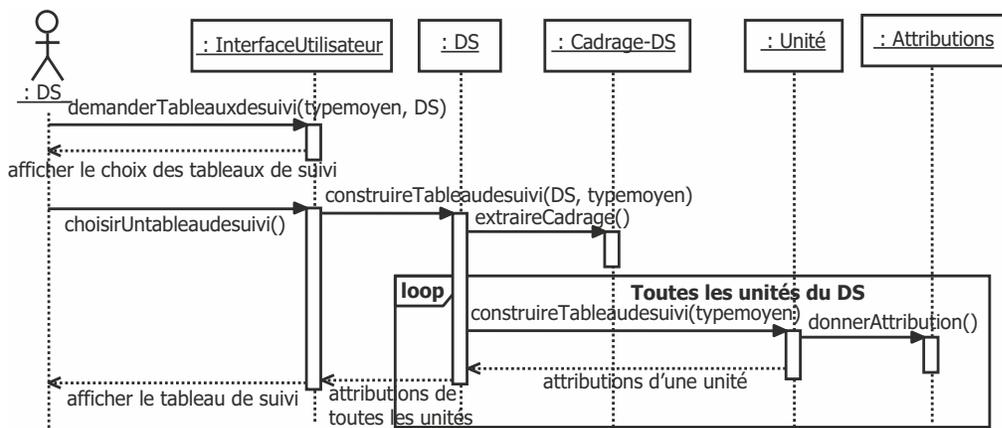


Figure 5.27 – Diagramme de séquence du cas d'utilisation 8

Tableau 5.8 – Données de l'interface utilisateur du cas d'utilisation 8

Données affichées	Données saisies
- Liste des tableaux de suivi	- Choix du tableau de suivi

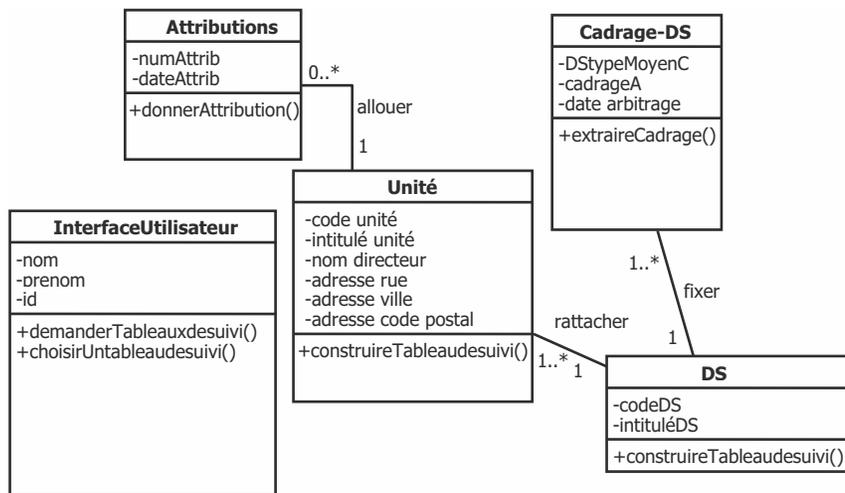


Figure 5.28 – Diagramme de classe du cas d'utilisation 8

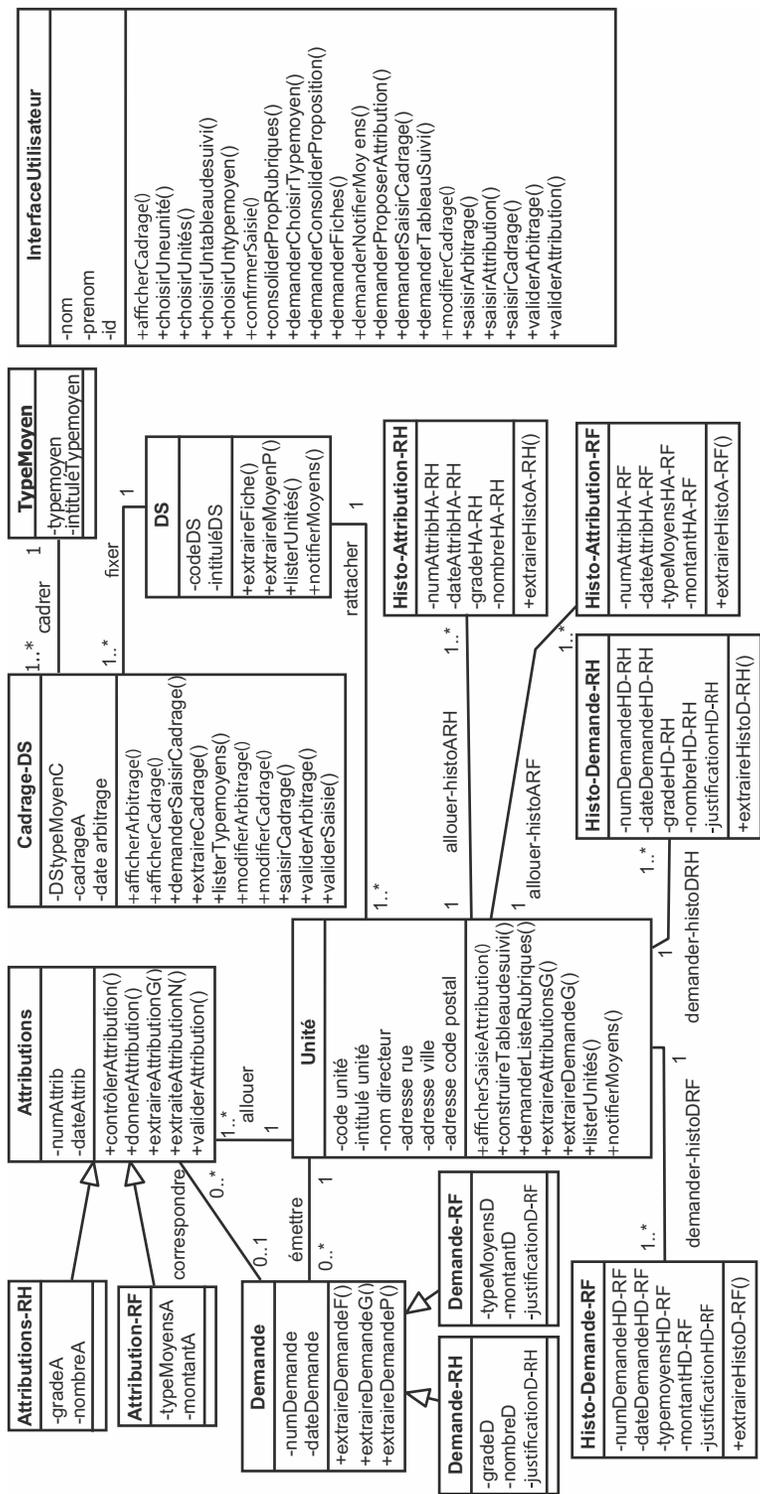


Figure 5.29 — Diagramme de classe de synthèse

6

Étude de cas n° 2 Analyse et conception

6.1 ÉNONCÉ DU CAS GESTION ACTIVITÉ ET FRAIS

La société JeConseille, spécialisée dans le conseil et l'audit auprès de petites et grandes entreprises, souhaite automatiser son système de reporting d'activité et de frais. Elle désire que son nouveau système soit accessible par tous ses employés lors de leurs missions. Des niveaux de performances sont exigés : 100 connexions simultanées et les temps de réponse pour chaque écran ne doivent pas dépasser 5 secondes.

Le fonctionnement actuel du système repose sur la saisie dans un tableur, par les employés, de rapports prévisionnels d'activité et de frais mensuels. Ces rapports contiennent le nombre de jours travaillés prévisionnels dans le mois, la répartition par projet (nombre de jours/par projet), le trajet prévisionnel réalisé durant le mois (km) et un cumul des frais (€) prévisionnel dépensé durant le mois.

Ces rapports prévisionnels sont envoyés à la secrétaire de la division en début de mois par messagerie. La secrétaire relance *via* la messagerie les employés n'ayant pas fourni leurs rapports.

La secrétaire effectue par la suite une consolidation par division de tous les rapports prévisionnels afin d'obtenir une synthèse des activités, des frais par projet et le taux d'activité de la division.

Cette synthèse est consultée par le manager de la division tous les mois.

Une modification de l'activité ou des frais d'un consultant fait l'objet d'une modification du rapport enregistré et d'un nouvel envoi de mail à la secrétaire.

En fin de mois, la secrétaire reporte manuellement les informations nécessaires sur les activités et les frais des employés dans le système de facturation de l'entreprise.

Les fonctionnalités attendues sont décrites dans l'étape d'expression des besoins de la démarche UML.

6.2 MODÉLISATION MÉTIER

6.2.1 Élaboration du schéma de contexte du domaine d'étude (FG1)

Le schéma de contexte du cas « Gestion activité et frais » est présenté à la figure 6.1.

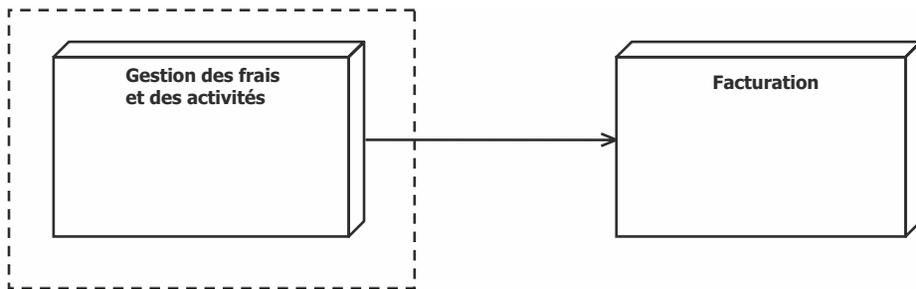


Figure 6.1 – Représentation du schéma de contexte

Deux sous-ensembles et leurs dépendances sont modélisés dans ce diagramme :

- **Gestion des frais et des activités** – Le sous-ensemble étudié tout au long de l'étude de cas.
- **Facturation** – Le sous-ensemble connexe dans lequel chaque mois sont transmises les données des activités et des frais pour établir la facturation de l'entreprise.

6.2.2 Élaboration du diagramme d'activité (FG2)

Le diagramme d'activité est donné à la figure 6.2.

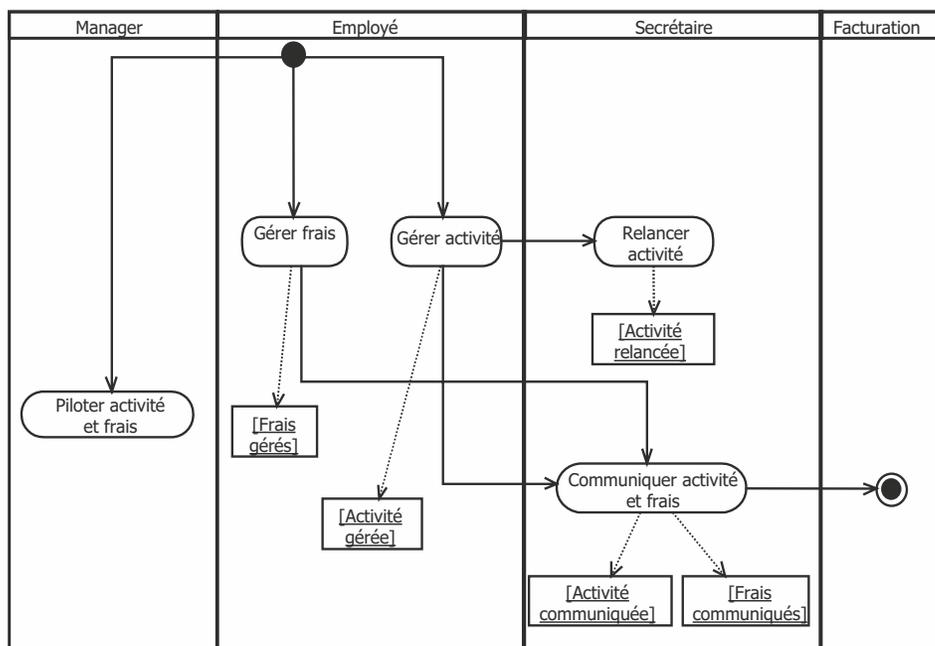


Figure 6.2 — Diagramme d'activité

Quatre acteurs sont identifiés :

- **Employé** – Il gère son activité et ses frais chaque mois (création, modification, consultation).
- **Secrétaire** – Il ou elle¹ relaie les employés n'ayant pas créé leur activité. Elle communique l'activité et les frais des employés au système de facturation en fin de mois.
- **Manager** – Il peut piloter l'activité et les frais de ses employés.
- **Facturation** – Cette entité correspond au système de facturation de l'entreprise qui reçoit les activités et frais des employés en fin de mois.

6.2.3 Élaboration du diagramme de classe métier (FG3)

Le diagramme de classe métier est donné à la figure 6.3.

1. Par simplification d'écriture et en considérant le cas traité, nous avons pris l'hypothèse que le poste de secrétaire est occupé par une femme.

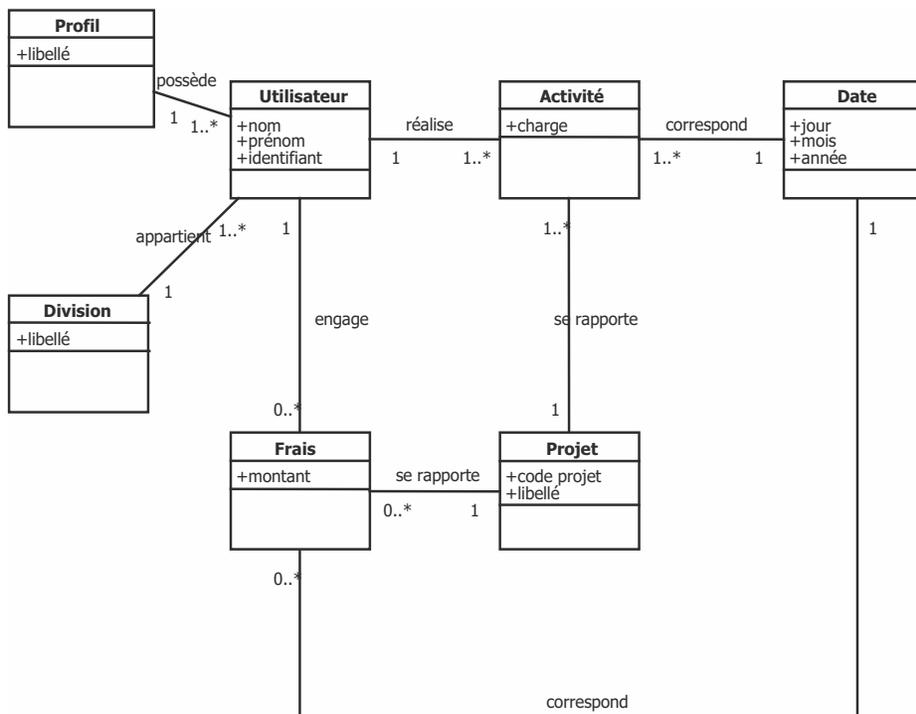


Figure 6.3 – Diagramme de classe métier

Définition des concepts du domaine (glossaire métier)

- **Utilisateur** – Ce concept englobe tous les acteurs utilisant l'application.
- **Profil** – Les utilisateurs appartiennent à différents profils (employé, secrétaire, manager) qui ont des habilitations différentes sur l'application.
- **Division** – L'entreprise est structurée en plusieurs divisions, chacune ayant sa spécificité (marketing, comptabilité...).
- **Activité** – Ce concept correspond à la quantité de travail fournie par chaque employé de l'entreprise.
- **Frais** – Ce concept correspond aux différentes dépenses survenues pour chaque employé dans le cadre de son activité.
- **Projet** – Ce concept correspond au moyen de mettre en œuvre l'activité des employés au sein de l'entreprise.
- **Date** – Ce concept permet d'archiver l'activité et les frais des employés de l'entreprise.

Les concepts métiers décrits sont illustrés sur les figures 6.4 et 6.5. Il s'agit des formulaires utilisés avant informatisation : le relevé mensuel d'activité et le relevé mensuel de frais.

JeCounselle		RAPPORT MENSUEL D'ACTIVITE																																
MOIS : Octobre		NOM PRENOM : DUPONT JEAN														DIVISION : Marketing																		
PROJET	COEF PROJET	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	TOTAL JOURS	
		Lu	Ma	Me	Je	Ve	Sa	Di	Lu	Ma	Me	Je	Ve	Sa	Di	Lu	Ma	Me	Je	Ve	Sa	Di	Lu	Ma	Me	Je	Ve	Sa	Di	Lu	Ma	Me		
PROJET1	P1	1	1	1	1	1			1	1	0.5	1				1	1	1	1	1			1	1	1	1	1	1	1		1	1	1	11.50
PROJET2	P2										1	0.5																						11.50

6.3 EXIGENCES FONCTIONNELLES

6.3.1 Élaboration du diagramme des cas d'utilisation système (FG4)

À partir du diagramme d'activité et de la connaissance des besoins des acteurs, nous élaborons une vision générale des cas d'utilisation métiers du futur système (fig. 6.6).

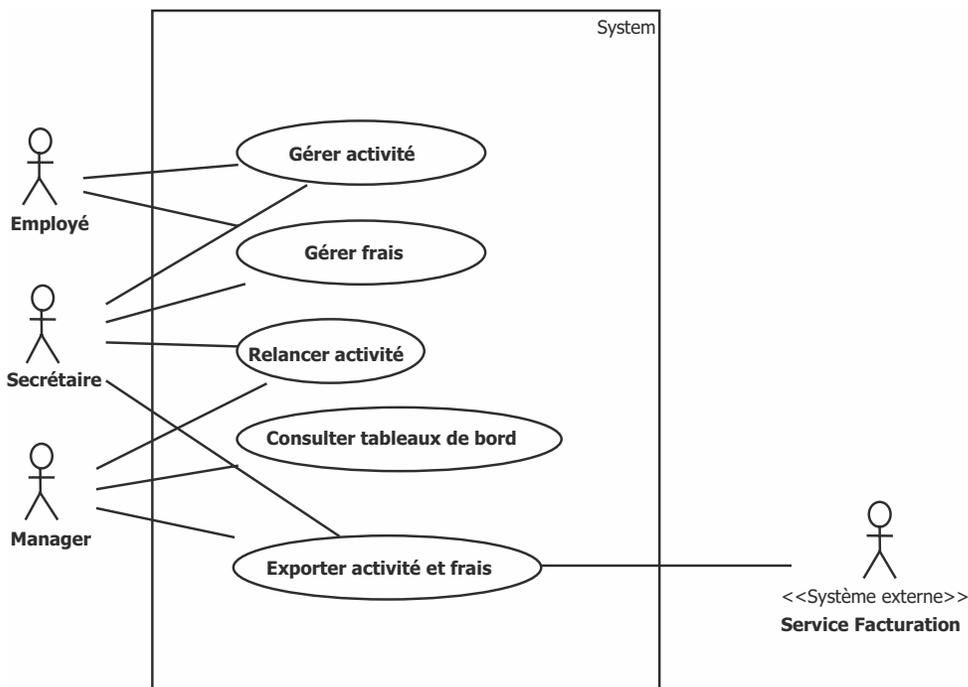


Figure 6.6 – Diagramme des cas d'utilisation système

Description générale des cas d'utilisation métiers

- **Cas d'utilisation 1- « Gérer activité »** – L'employé renseigne son activité en début de mois pour les projets sur lesquels il va travailler. La description de l'activité est réalisée par journée. La secrétaire peut ainsi retrouver, pour une journée donnée, l'activité réalisée par l'employé. Un récapitulatif de son activité sur le mois en cours ou sur les mois précédents est proposé à l'employé. Si l'employé ne peut renseigner son activité (maladie, absence), la secrétaire est habilitée à renseigner son activité.
- **Cas d'utilisation 2- « Gérer frais »** – L'employé renseigne ses frais prévisionnels (frais kilométriques, frais divers) en début de mois pour les projets sur lesquels il va travailler. La description des frais est réalisée par journée. La secrétaire peut ainsi retrouver pour une journée donnée, les frais prévisionnels d'un employé.

Un récapitulatif de ses frais sur le mois en cours ou sur les mois précédents est proposé à l'employé. Si l'employé ne peut renseigner ses frais pour cause de maladie ou absence, la secrétaire est habilitée à renseigner ses frais.

- **Cas d'utilisation 3- « Relancer activité »** – La secrétaire relance par messagerie, à partir du 10 de chaque mois, les employés n'ayant pas ou partiellement renseigné leurs activités mensuelles. Si la secrétaire ne peut relancer l'activité, le manager est habilité à le faire à sa place.
- **Cas d'utilisation 4- « Consulter tableaux de bord »** – Le manager visualise des tableaux de bord sur l'activité, les frais, et le taux d'activité pour un projet donné ou une division donnée et sur une période donnée.
- **Cas d'utilisation 5- « Exporter activités et frais »** – À la fin de chaque mois, la secrétaire exporte les activités et les frais de tous les employés vers le service facturation. Si la secrétaire ne peut exporter les activités et les frais, le manager est habilité à le faire à sa place.

6.3.2 Élaboration des diagrammes de séquence système (FG5)

Chaque cas d'utilisation décrit ci-dessus donne lieu à un diagramme de séquence système :

- DSE du CU Gérer activités (fig. 6.7)
- DSE du CU Gérer frais (fig. 6.8)
- DSE du CU Consulter tableaux de bord (fig. 6.9)
- DSE du CU Exporter activités et frais (fig. 6.10)
- DSE du CU Relancer activité (fig. 6.11)

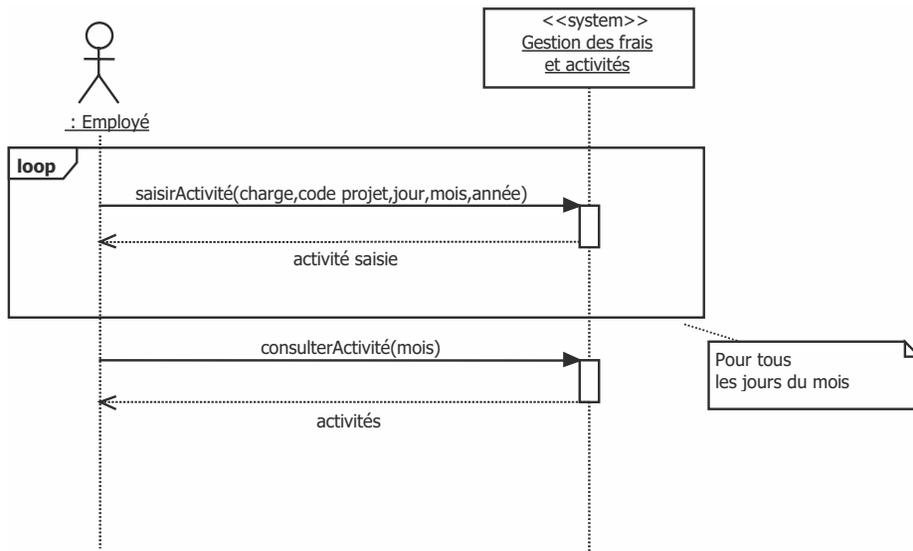


Figure 6.7 – DSE du CU Gérer activités

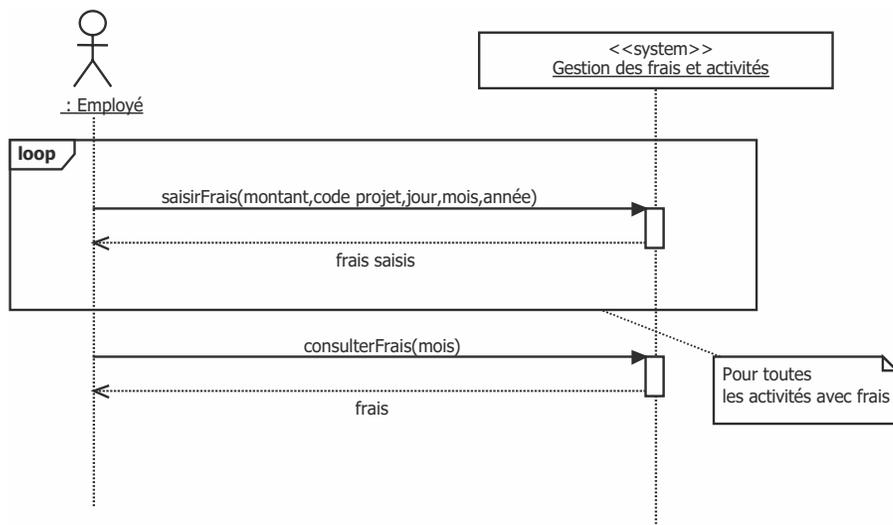


Figure 6.8 – DSE du CU Gérer frais

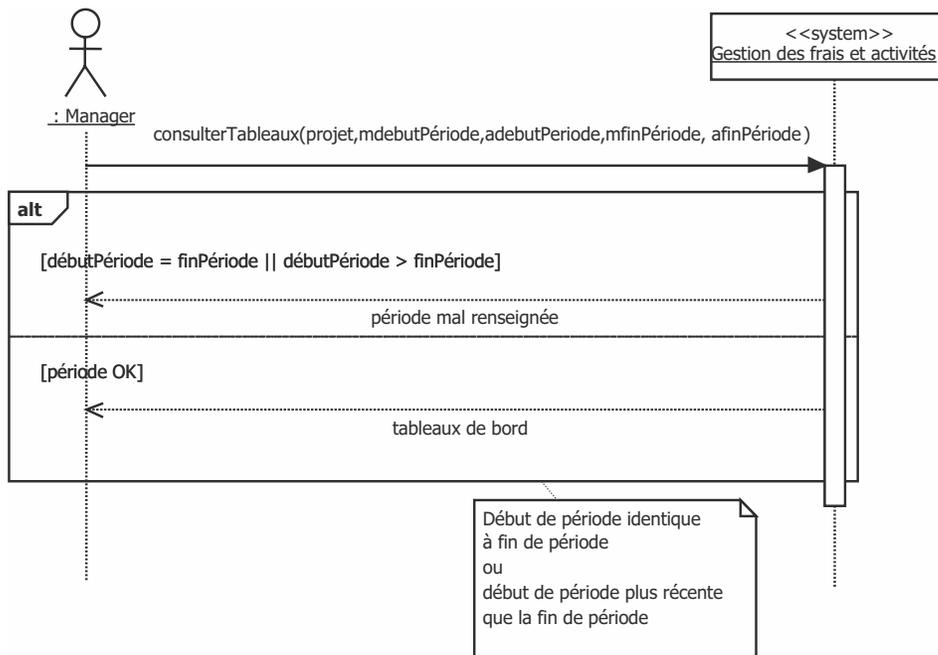


Figure 6.9 – DSE du CU Consulter tableaux de bord

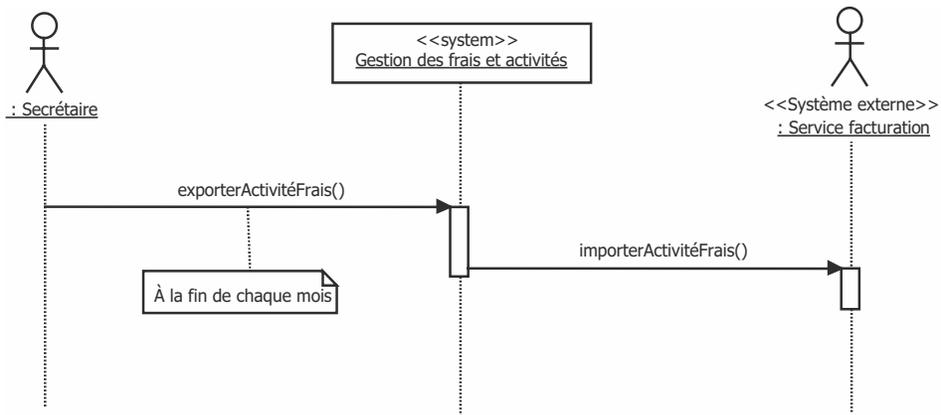


Figure 6.10 — DSE du CU Exporter activités et frais

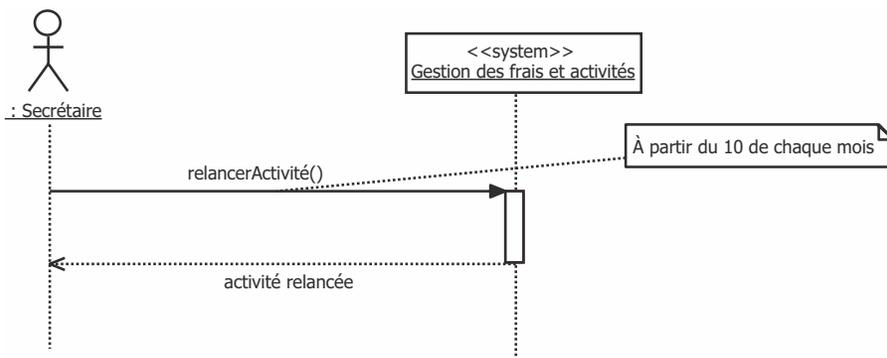


Figure 6.11 — DSE du CU Relancer activité

6.3.3 Élaboration du schéma de navigation générale (FG6)

Dans le cadre de l'expression des exigences fonctionnelles, l'enchaînement des futurs écrans de l'application avec les habilitations des différents acteurs est présenté sur la figure 6.12.

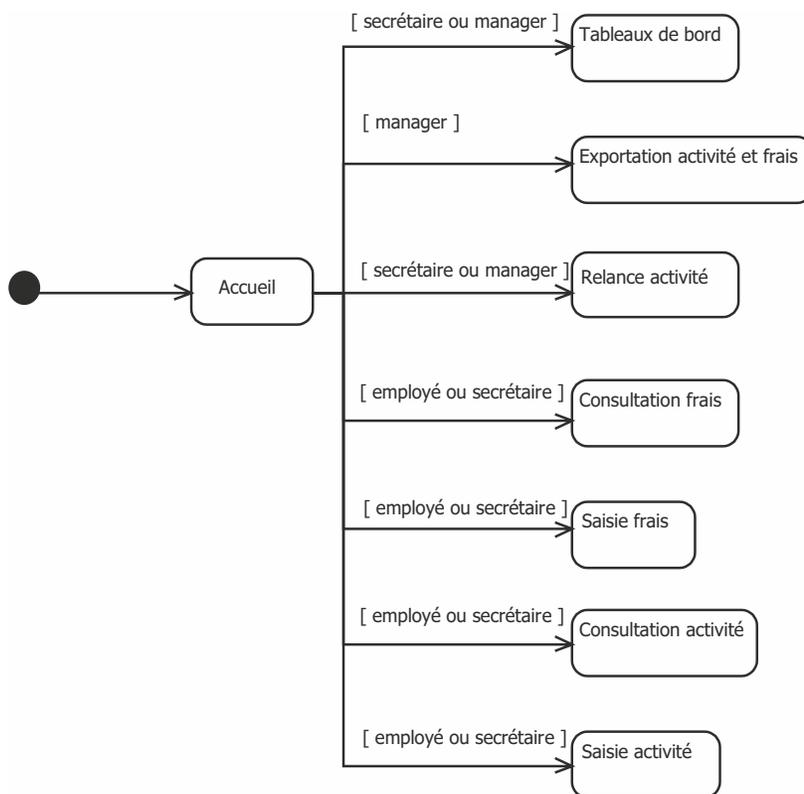


Figure 6.12 — Schéma de navigation générale

6.4 ANALYSE DES CAS D'UTILISATION

6.4.1 Élaboration du diagramme des cas d'utilisation (FG7)

Ce diagramme de cas d'utilisation (fig. 6.13) est plus détaillé que celui présenté au paragraphe précédent. En effet, nous sommes passés dans une phase d'analyse qui correspond à une vue informatique du système et nous avons identifié 13 cas d'utilisation.

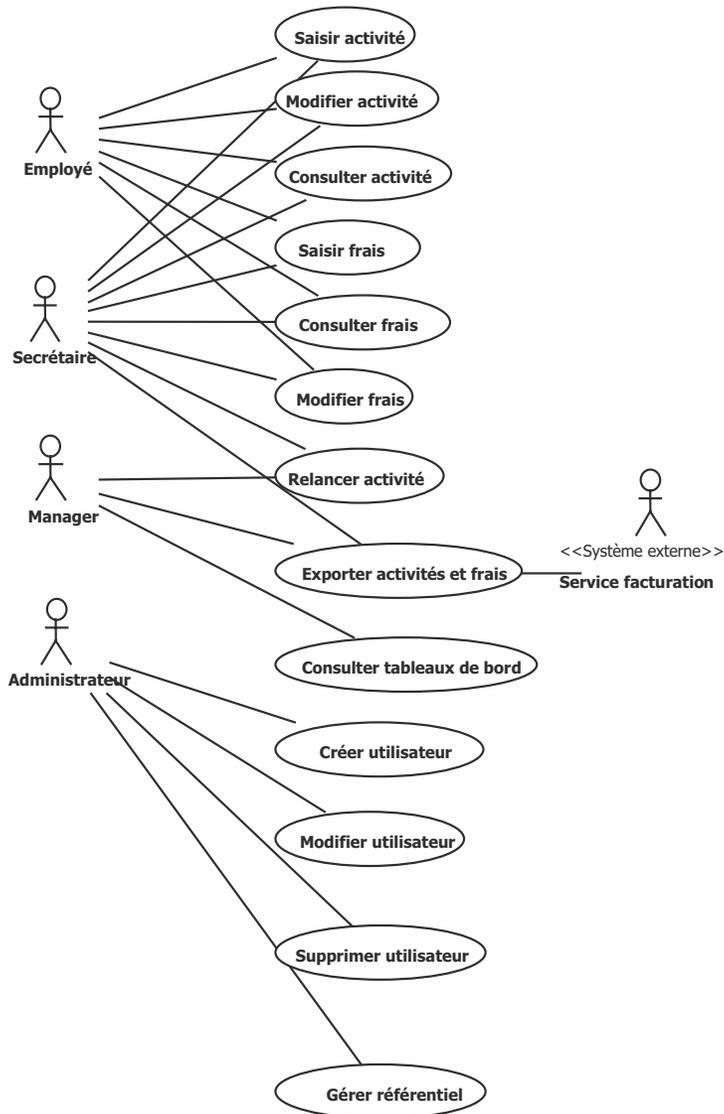


Figure 6.13 – Diagramme des cas d'utilisation

6.4.2 Description des cas d'utilisation (FG8, FG9, FG11, FG12)

Pour la suite de l'étude de cas, nous allons poursuivre l'analyse sur les cinq cas d'utilisation suivants seulement :

- Saisir activité
- Consulter activité
- Relancer activité

- Consulter tableaux de bord
- Créer utilisateur

Pour chaque cas d'utilisation, les sous-activités suivantes de l'activité « Analyse des cas d'utilisation » sont réalisées :

- Description du cas d'utilisation (FG8)
- Élaboration du diagramme de séquence (FG9)
- Élaboration de l'interface utilisateur (FG11)
- Élaboration du diagramme de classe (FG12)

Les scénarios alternatifs des cas d'utilisation ont été modélisés uniquement sur le cas d'utilisation « Relancer activité ».

Cas d'utilisation « Saisir Activité »

Description textuelle du cas d'utilisation

- **Objectif** - Saisir l'activité mensuelle de l'employé.
- **Acteur concerné** – L'employé.
- **Pré condition** – L'employé s'est authentifié correctement à l'application
- **Scénario nominal**
 - 1 L'application propose la saisie de la première semaine du mois en cours.
 - 2 L'employé sélectionne un ou plusieurs projets correspondant à l'activité de la semaine, puis saisit la charge consommée estimée sur chaque projet. Cette charge consommée est indiquée en jour.
 - 3 L'employé valide l'activité de la semaine.
 - 4 L'employé répète les actions 1 à 3 pour toutes les semaines du mois à traiter.
- **Scénario alternatif**
 - 2a L'employé saisit une charge négative pour un projet pour une journée donnée.
 - L'employé saisit une charge négative pour un projet sur une journée donnée.
 - L'employé valide l'activité de la semaine.
 - L'application avertit l'employé qu'une charge négative a été saisie.
 - Le cas d'utilisation reprend au point 2.

Description des diagrammes d'analyse du cas d'utilisation

La suite de l'analyse du cas d'utilisation se poursuit par l'élaboration du diagramme de séquence (fig. 6.14), l'élaboration de l'interface utilisateur (fig. 6.15) et l'élaboration du diagramme de classe (fig. 6.16).

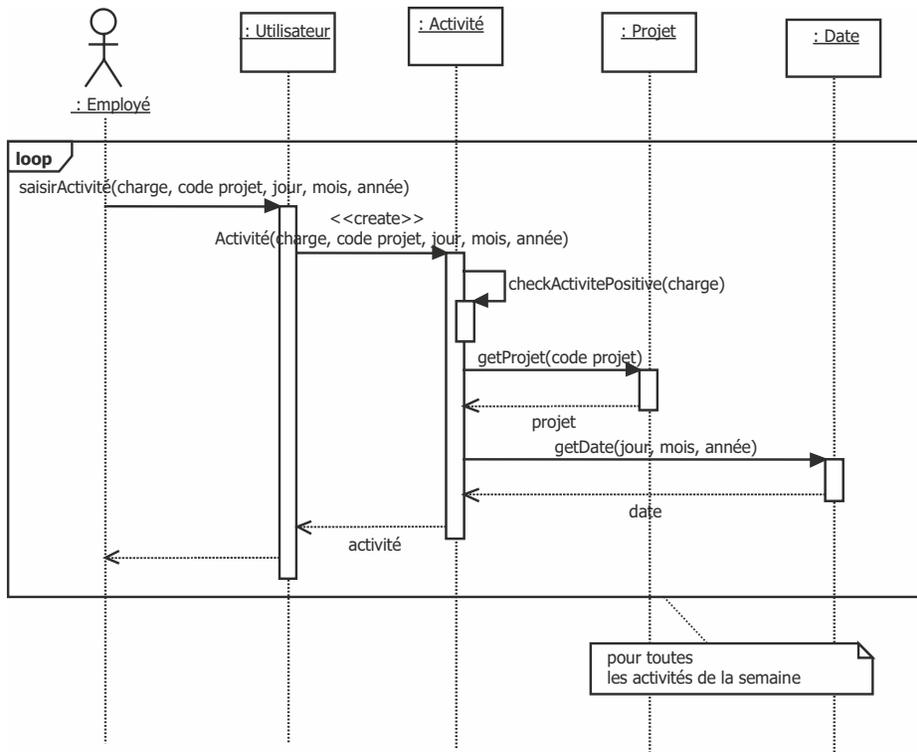


Figure 6.14 – Diagramme de séquence du CU Saisir activité

Saisir activité février

	S 1	S 2	S 3	S 4							
					Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche
Code projet 1 <input type="button" value="↓"/>		0.5	0.5	1	1	1					4 j
Code projet 2 <input type="button" value="↓"/>		0.5	0.5								1 j
										Total : 5 j	
<input type="button" value="Ajouter un code"/>											
<input type="button" value="Valider"/>											

Figure 6.15 – Écran Saisir Activité

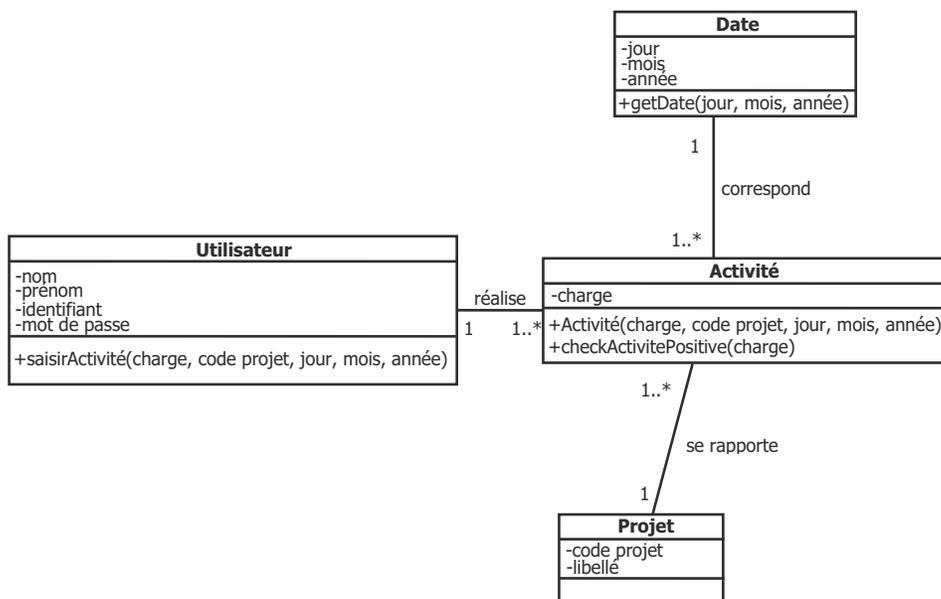


Figure 6.16 – Diagramme de classe du CU Saisir activité

Cas d'utilisation « Consulter activité »

Description textuelle du cas d'utilisation

- **Objectif** – Pour un employé donné, avoir un récapitulatif de l'activité sur le mois en cours ou sur les mois précédents.
- **Acteur concerné** – L'employé.
- **Pré condition** – L'employé s'est authentifié correctement à l'application.
- **Scénario nominal**
 - 1 L'employé sélectionne un mois.
 - 2 L'application présente l'activité mensuelle de l'employé avec la répartition entre les différents projets (si l'employé a participé à plusieurs projets au cours du mois).
 - 3 L'application fait aussi un cumul global de l'activité mensuelle de l'employé, puis des cumuls mensuels de l'activité par projet.
- **Scénario alternatif**
 - 2a : L'application ne peut présenter l'activité car elle n'a pas été encore saisie pour le mois sélectionné.
 - L'application affiche un message indiquant qu'il n'y a pas d'activité pour le mois sélectionné.
 - L'employé revient sur la sélection du mois. Le cas d'utilisation reprend au point 1.

Description des diagrammes d'analyse du cas d'utilisation

La suite de l'analyse du cas d'utilisation se poursuit par l'élaboration du diagramme de séquence (fig. 6.17), l'élaboration de l'interface utilisateur (fig. 6.18), et l'élaboration du diagramme de classe (fig. 6.19).

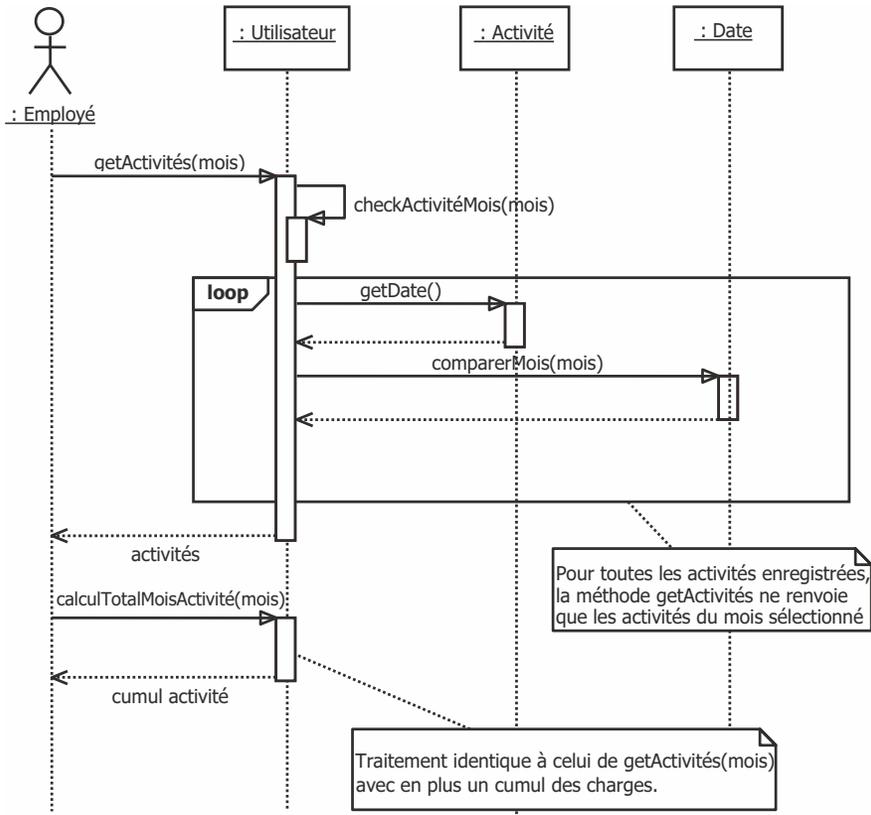


Figure 6.17 — Diagramme de séquence du CU Consulter activité

Consulter activité

Sélectionner un mois pour afficher le récapitulatif de votre activité :

Mois : Mois ↓

Valider

Consulter activité février

	L1	M2	M3	J4	V5	S5	D6	L7	M8	M9	J10	V11	S12	D13	L14	M15	M16	J17	V18	S19	D20	L21	M22	M23	J24	V25	S26	D27	L28
CP1	1	1	1	1	1																								
CP2								1	1	1	1	1				1	1	1	1	1			1	1	1	1	1		1

Total : 21 j

Figure 6.18 — Écran Consulter sélection activité

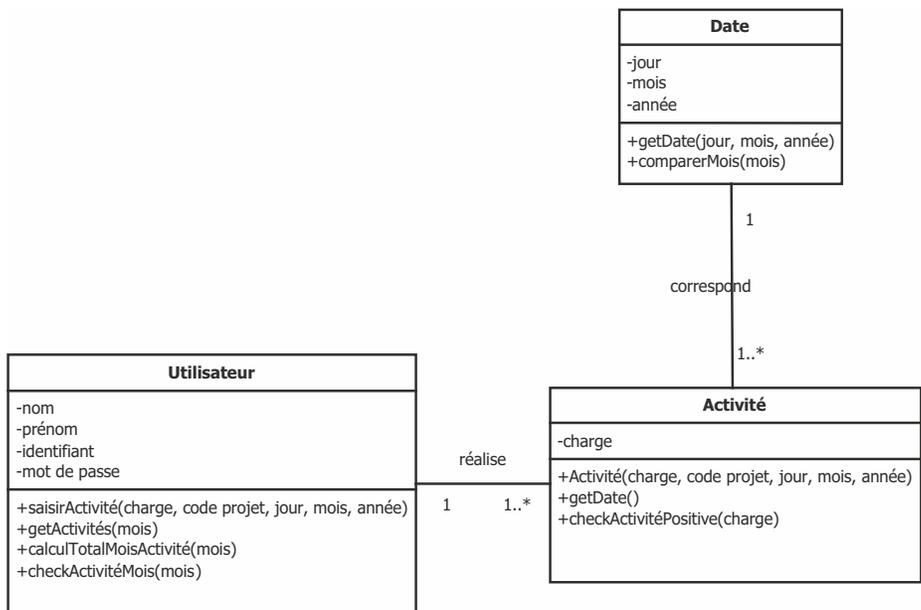


Figure 6.19 — Diagramme de classe du CU Consulter activité

Cas d'utilisation « Relancer activité »

Description textuelle du cas d'utilisation

- **Objectif** – Relancer, à partir du 10 de chaque mois, les employés n'ayant pas saisi leur activité du mois.
- **Acteur concerné** – La secrétaire.
- **Pré condition** – La secrétaire s'est authentifiée correctement à l'application.
- **Scénario nominal**
 - 1 L'application présente la liste des employés qui n'ont pas saisi (ou partiellement) leur activité mensuelle.
 - 2 La secrétaire valide cette liste.
 - 3 L'application envoie un courriel de relance aux employés concernés.
- **Scénarios alternatifs**
 - 1a Relance effectuée avant le 10 du mois en cours.**
 - L'application affiche un message indiquant que la relance ne peut avoir lieu qu'à partir du 10 du mois. Le cas d'utilisation se termine en échec.
 - 1b Tous les employés ont rempli leur activité.**
 - L'application affiche un message indiquant que tous les employés ont saisi leur activité. Le cas d'utilisation se termine en échec.

Description des diagrammes d'analyse du cas d'utilisation

La suite de l'analyse du cas d'utilisation se poursuit par l'élaboration de diagrammes de séquence (fig. 6.20 et 6.21), l'élaboration de l'interface utilisateur (fig. 6.22), et l'élaboration du diagramme de classe (fig. 6.23).

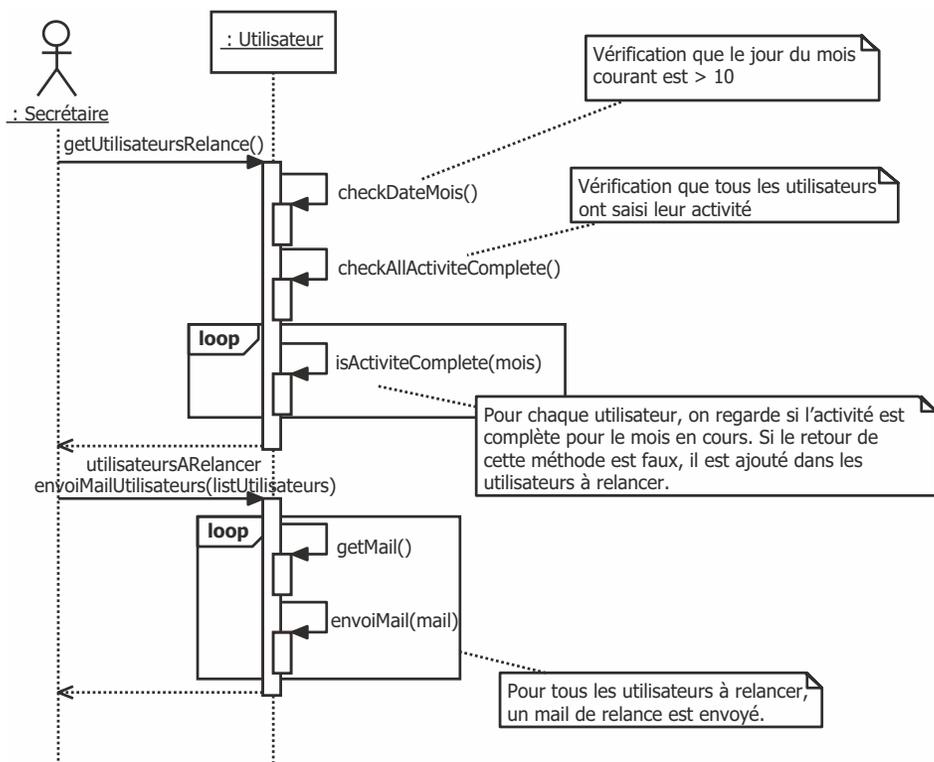


Figure 6.20 — Diagramme de séquence du CU Relancer activité (scénario nominal)

Sur le même DSE (fig. 6.21) sont représentés le scénario alternatif « 1a Relance effectuée avant le 10 du mois en cours » et le scénario alternatif « 1b Tous les employés ont rempli leur activité ».

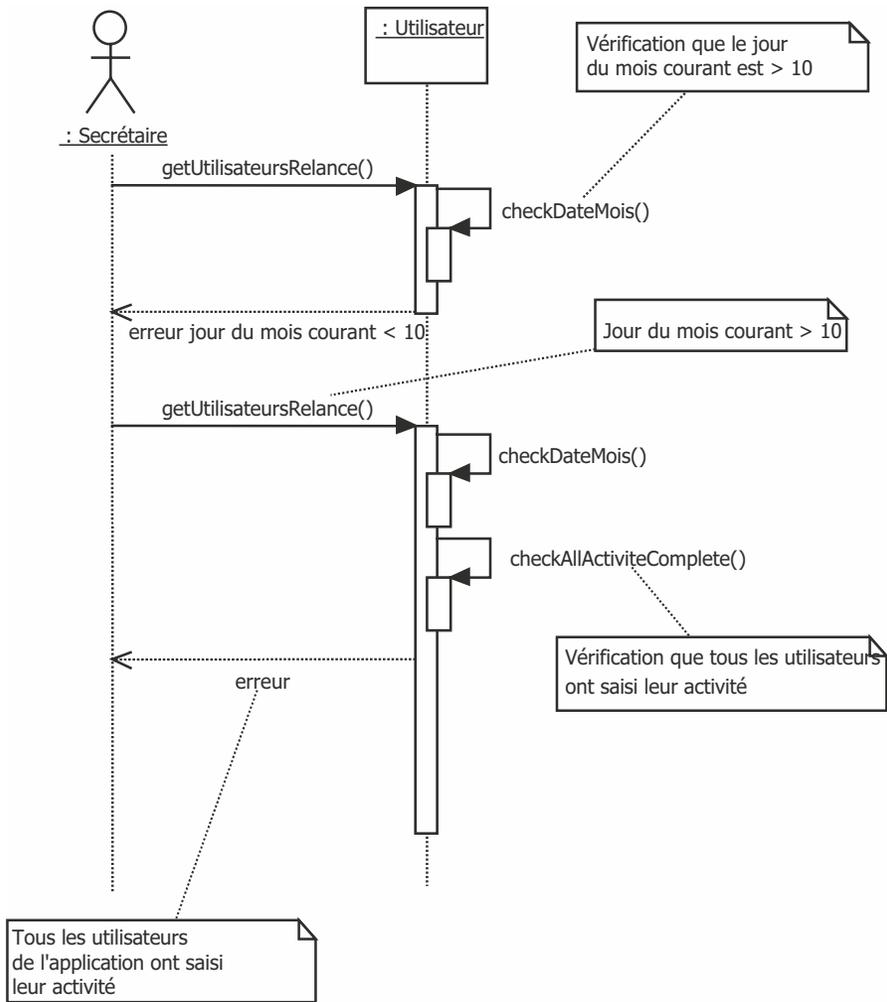


Figure 6.21 — Diagramme de séquence du CU Relancer activité (scénarios alternatifs)

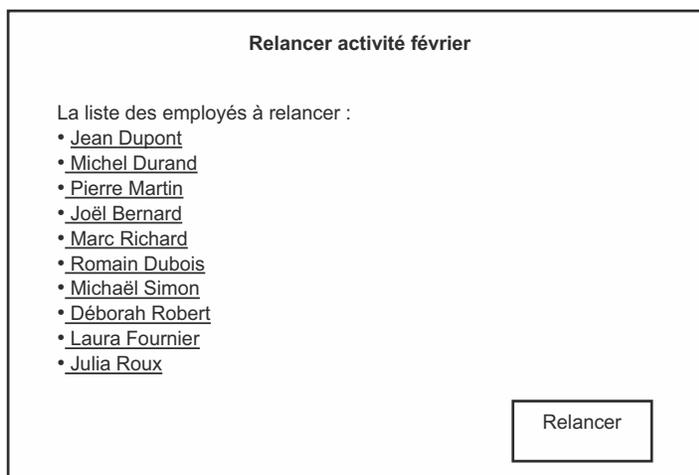


Figure 6.22 — Écran Relancer activité

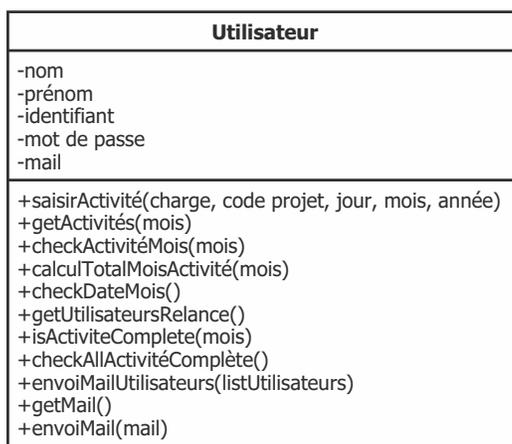


Figure 6.23 — Diagramme de classe du CU Relancer activité

Cas d'utilisation « Consulter tableaux de bord »

Description textuelle du cas d'utilisation

- **Objectif** – Fournir au manager des tableaux de bord sur l'activité, les frais, le taux d'activité pour un projet donné ou une division donnée et sur une période donnée.
- **Acteur concerné** – Le manager.
- **Pré condition** – Le manager s'est authentifié correctement à l'application.

- **Scénario nominal**
 - 1 Le manager sélectionne une période (mois, année).
 - 2 L'application propose des tableaux de bord sur la période et la division du manager concernant l'activité par projet (nombre de jours/h par projet), le taux d'activité (nombre de jour/h sur un projet / nombre de jour/h total de la période).
 - 3 L'application propose aussi un affichage de graphiques.
- **Scénarios alternatifs**
 - 1a **Le manager sélectionne une période erronée** (date début de période est supérieure à la date de fin de période ou date de début de période est identique à la date de fin de période).
 - L'application présente un message d'erreur sur la période sélectionnée non valide au manager. Le cas d'utilisation reprend au point 1.
 - 2a **Aucun employé n'a rempli son activité et ses frais sur la période sélectionnée.**
 - L'application affiche un message indiquant qu'aucun employé n'a saisi son activité ou ses frais. Le cas d'utilisation se termine en échec.

Description des diagrammes d'analyse du cas d'utilisation

La suite de l'analyse du cas d'utilisation se poursuit par l'élaboration de l'élaboration du diagramme de séquence (fig. 6.24), l'élaboration de l'interface utilisateur (fig. 6.25, fig. 6.26), et l'élaboration du diagramme de classe (fig. 6.27).

Par simplification, l'affichage des graphiques n'est pas traité sur le DSE figure 6.24 ni dans le reste de l'étude de cas.

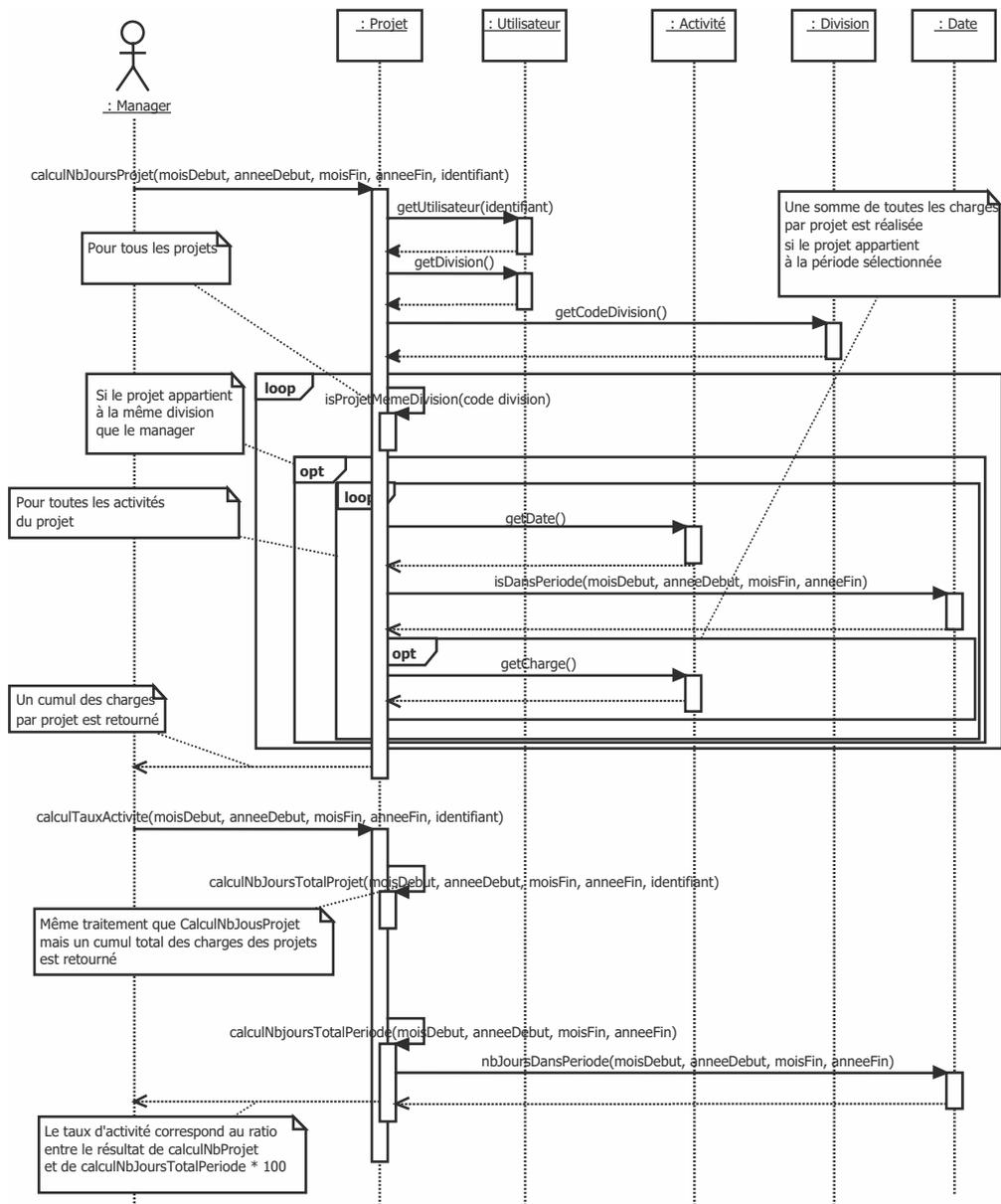


Figure 6.24 — Diagramme de séquence du CU Consulter tableaux de bord

Sélection tableaux de bord

Sélectionner une période pour afficher les tableaux de bord pour la division **Industrie** :

Début période : ↓ ↓

Fin période : ↓ ↓

Figure 6.25 – Écran Sélection tableaux de bord

Consulter tableaux de bord

La liste des projets et leurs activités pour la division Industrie sur la période novembre/décembre (effectif total : 10 personnes, durée de la période : 40 j) :

Projet	Activité (j/h)
Code projet 1	100
Code projet 2	80
Code projet 3	80
Code projet 4	30
Code projet 5	70

Taux d'activité de la division Industrie pour la période novembre/décembre :

Période	Taux d'activité de la division (%)
Novembre/décembre	90

[graphiques](#)

Figure 6.26 – Écran Consulter tableaux de bord

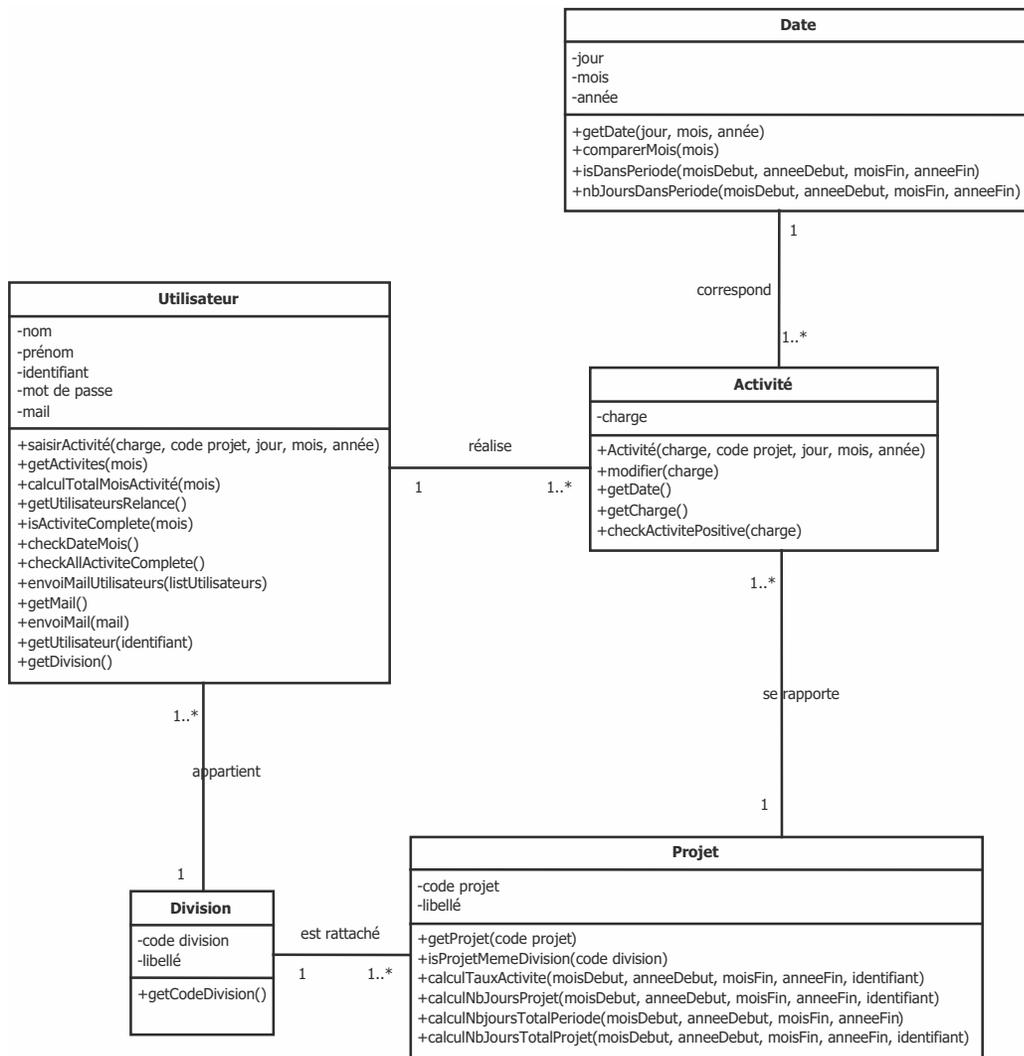


Figure 6.27 – Diagramme de classe du CU Consulter tableaux de bord

Cas d'utilisation « Créer utilisateur »

Description textuelle du cas d'utilisation

- **Objectif** – Initialiser l'application avec les utilisateurs.
- **Acteur concerné** – Administrateur.
- **Pré condition** – L'administrateur s'est authentifié correctement à l'application.
- **Scénario nominal**
 - 1 L'administrateur crée un utilisateur avec ses caractéristiques : prénom, nom, mail, profil.
 - 2 L'application enregistre les informations renseignées et affiche un message de confirmation.
 - 3 Un mail est envoyé à l'utilisateur créé avec son identifiant/password.
- **Scénarios alternatifs**
 - 1a L'administrateur saisit un nom ou un prénom avec des chiffres.**
 - L'application affiche un message d'erreur précisant que le nom ou prénom sont des chaînes de caractères uniquement. Le cas d'utilisation reprend au point 1.
 - 1b L'administrateur saisit un nom ou prénom supérieur à 50 caractères.**
 - L'application affiche un message d'erreur précisant que le nom ou prénom sont des chaînes de caractères inférieures à 50 caractères. Le cas d'utilisation reprend au point 1.
 - 1c L'administrateur saisit un mail erroné (contrôle sur le @).**
 - L'application affiche un message d'erreur précisant que le mail n'est pas valide. Le cas d'utilisation reprend au point 1.

Description des diagrammes d'analyse du cas d'utilisation

La suite de l'analyse du cas d'utilisation se poursuit par l'élaboration du diagramme de séquence (fig. 6.28), l'élaboration de l'interface utilisateur (fig. 6.29) et l'élaboration du diagramme de classe (fig. 6.30).

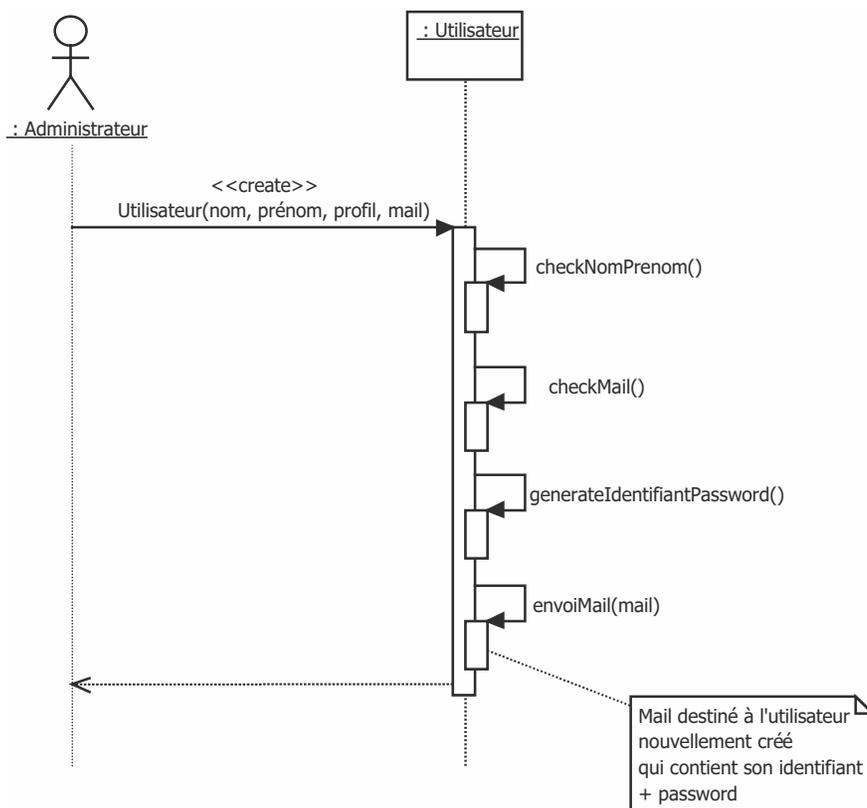


Figure 6.28 – Diagramme de séquence du CU Créer utilisateur

Créer utilisateur

Saisissez les informations relatives au nouvel utilisateur :

Prénom :

Nom :

Mail :

Profil :

Figure 6.29 – Écran Créer utilisateur

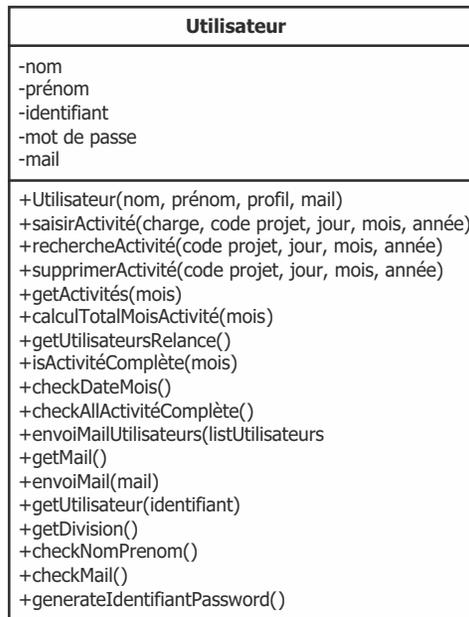


Figure 6.30 — Diagramme de classe du CU Créer Utilisateur

6.5 SYNTHÈSE DE L'ANALYSE

6.5.1 Élaboration du diagramme de classe récapitulatif (FG13)

Ce diagramme de classe (fig. 6.31) intègre l'ensemble des diagrammes de classe élaborés par cas d'utilisation.

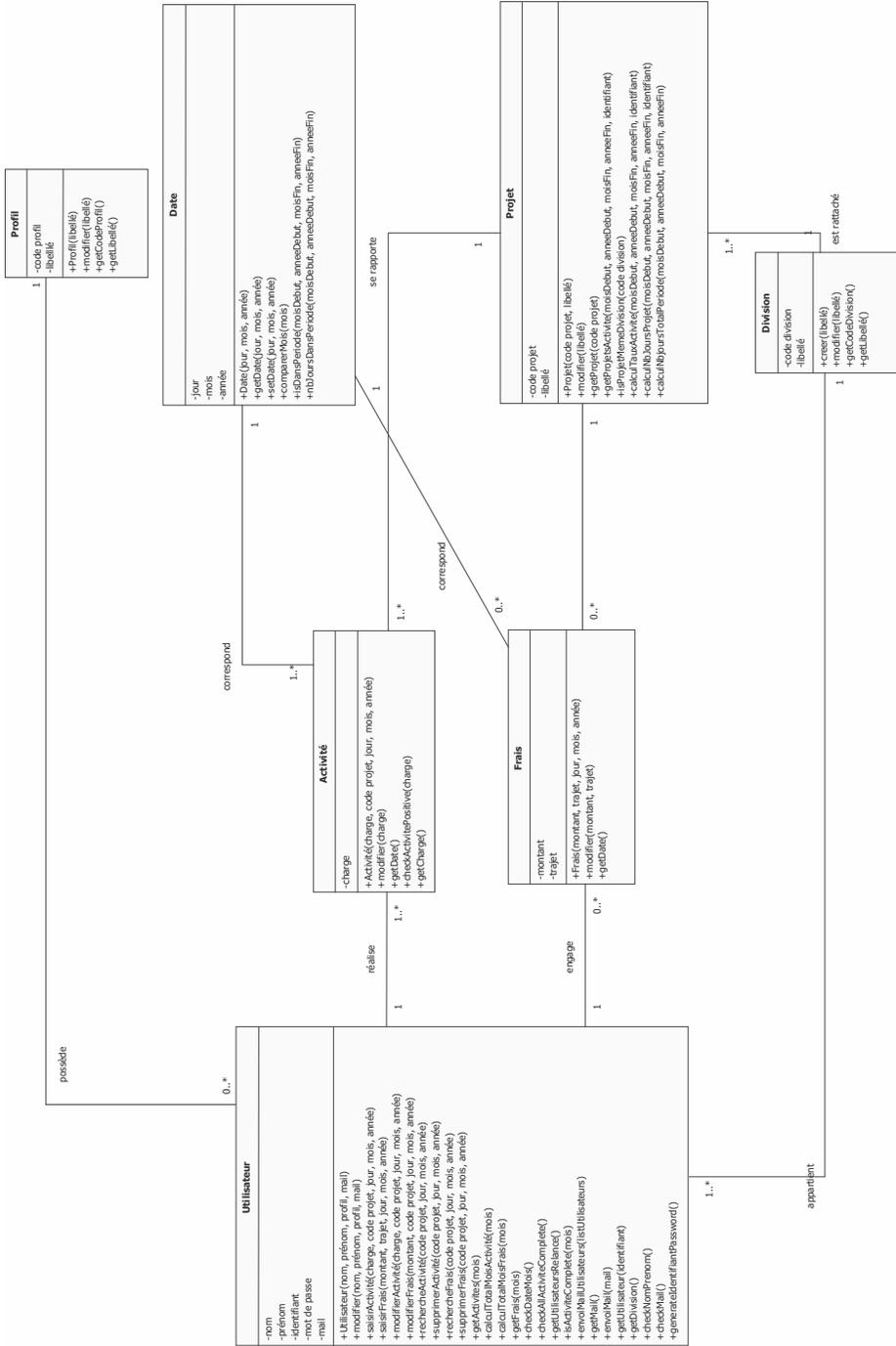


Figure 6.31 — Diagramme de classe récapitulatif

6.5.2 Élaboration de la matrice de validation (FG14)

La matrice de validation (fig. 6.32) permet de vérifier que l'analyse du cas est complète, c'est-à-dire que tous les cas d'utilisation métier ont été intégrés. Elle permet aussi d'établir une correspondance entre les cas d'utilisation métier et les cas d'utilisation d'analyse.

Cas d'utilisation métier	Cas d'utilisation analyse
Gérer activités	Saisir activité
Gérer activités	Modifier activité
Gérer activités	Consulter activité
Gérer frais	Saisir frais
Gérer frais	Modifier frais
Gérer frais	Consulter frais
Relancer activité	Relancer activité
Consulter tableaux de bord	Consulter tableaux de bord
Exporter activités et frais	Exporter activités et frais
	Créer utilisateur
	Modifier utilisateur
	Supprimer utilisateur
	Gérer référentiel

Figure 6.32 — Matrice de validation

6.6 CONCEPTION

Pour répondre aux contraintes techniques de l'entreprise JeConseille (« Elle désire que son nouveau système soit accessible par tous ses employés lors de leurs missions »), la conception d'un site web est nécessaire.

6.6.1 Réalisation des choix techniques et élaboration des diagrammes techniques (FG15, FG16, FG17)

Pour répondre aux contraintes de performances (« 100 connexions simultanées et les temps de réponse pour chaque écran ne doivent pas dépasser 5 secondes ») de l'énoncé de l'étude de cas, notre choix technique portera ainsi sur une architecture orientée service (SOA) pour la conception du site web. En effet, comme précisé dans la présentation des architectures, les temps de réponse sont meilleurs du fait de l'abstraction introduite par la couche service. Le coût des appels aux objets métiers à partir de la couche service est très faible.

Concernant, la technologie pour cette architecture, J2EE sera mise en place. Cette solution ne sera pas détaillée dans cet ouvrage.

Pour chaque cas d'utilisation, les sous-activités suivantes de l'activité « Conception » sont réalisées :

- élaboration du diagramme de séquence technique (FG16),
- élaboration du diagramme de classe technique (FG17).

Cas d'utilisation « Saisir activité »

L'élaboration du diagramme de séquence technique (fig. 6.33), et l'élaboration du diagramme de classe technique (fig. 6.34) sont réalisées.

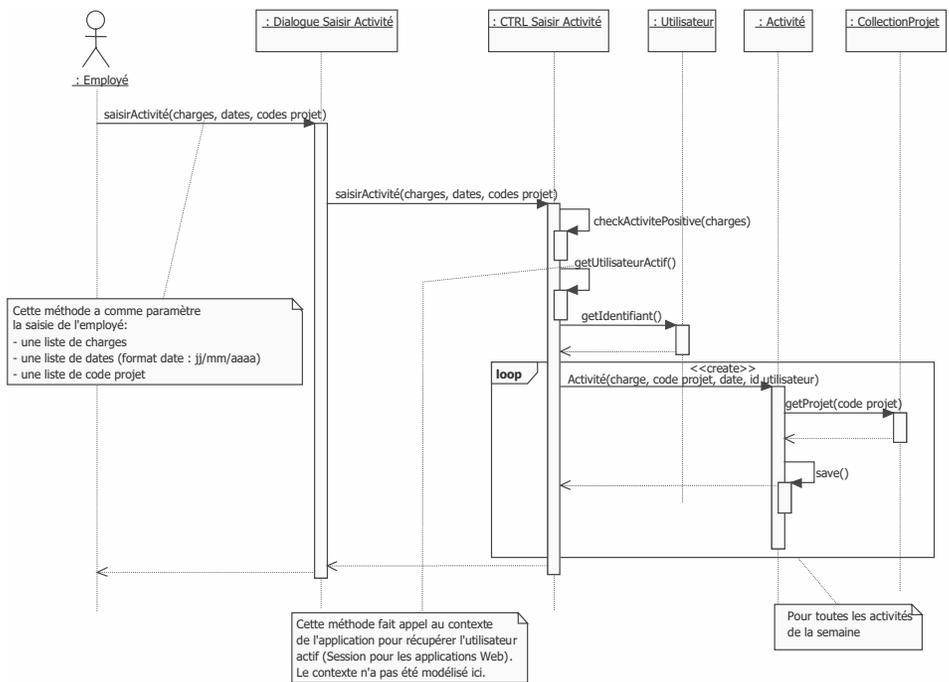


Figure 6.33 – Diagramme de séquence technique du CU Saisir activité

La navigabilité des associations est donnée par le sens de circulation des opérations du diagramme de séquence (fig. 6.33). Ainsi, la classe « Dialogue Saisir Activité » accède à la classe « CTRL Saisir Activité » par le biais du message *saisirActivité*. La navigabilité des autres associations est déterminée sur le même principe.

Une relation d'agrégation existe entre la classe « CollectionProjet » et « Projet » de type « ensemble/élément », de plus une contrainte « ordered » indique que les projets sont triés par code projet.

La contrainte « ordered » entre « Utilisateur » et « Activité » exprime la relation de tri par date croissante des « Activité » dans la collection « Utilisateur ».

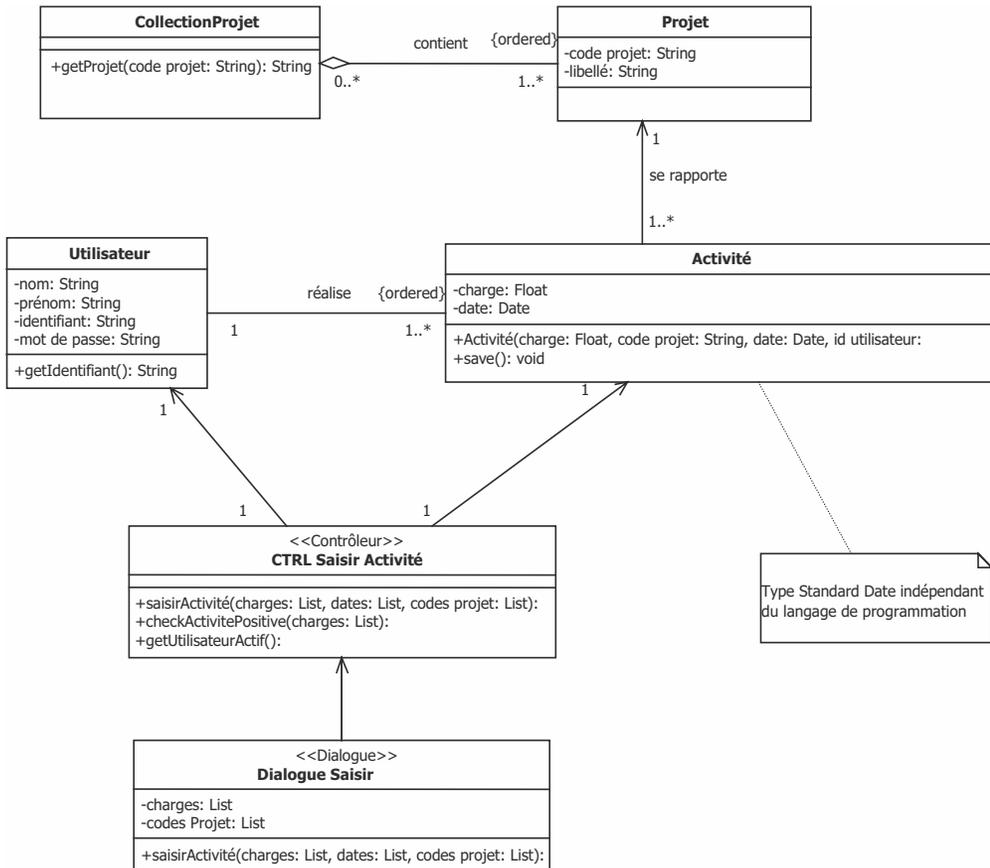


Figure 6.34 — Diagramme de classe technique du CU Saisir activité

Cas d'utilisation « Consulter Activité »

L'élaboration du diagramme de séquence technique (fig. 6.35), et l'élaboration du diagramme de classe technique (fig. 6.36) sont réalisées.

La navigabilité des associations est donnée par le sens de circulation des opérations du diagramme de séquence (fig. 6.34). Ainsi, la classe « Dialogue Consulter Activité » accède à la classe « CTRL Consulter Activité » par le biais du message *consulterActivité*. La navigabilité des autres associations est déterminée sur le même principe.

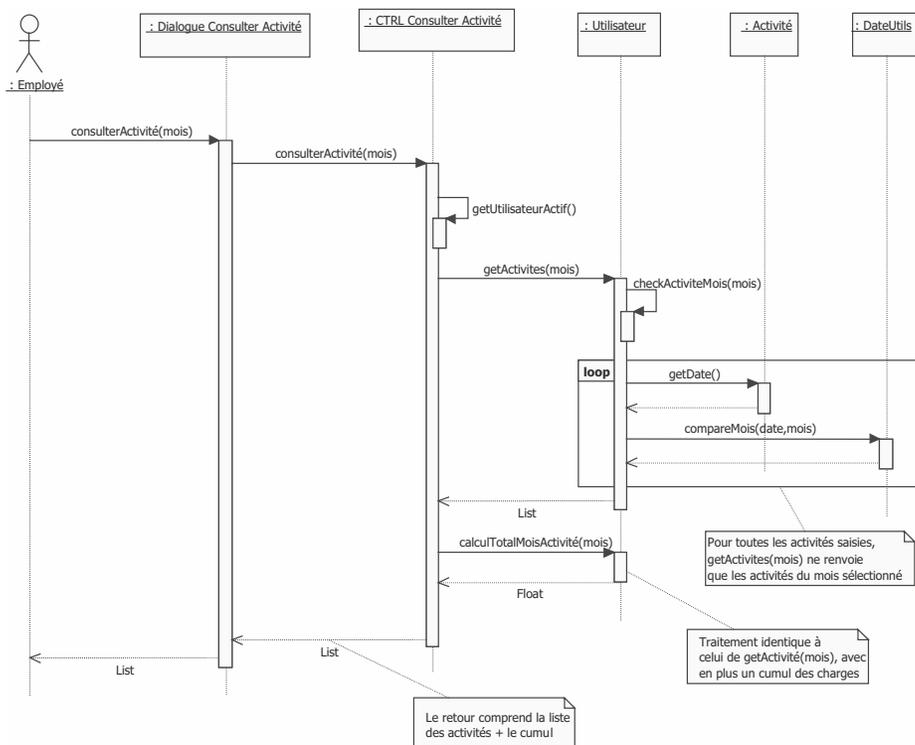


Figure 6.35 – Diagramme de séquence technique du CU Consulter activité

Une relation de dépendance est modélisée entre la classe « Utilisateur » et la classe technique « DateUtils ». On remarquera que le lien entre un objet « Utilisateur » et « DateUtils » est momentané, il ne dure que le temps d'exécution de la méthode *compareMois*. Ainsi ce n'est pas une association, mais une simple dépendance.

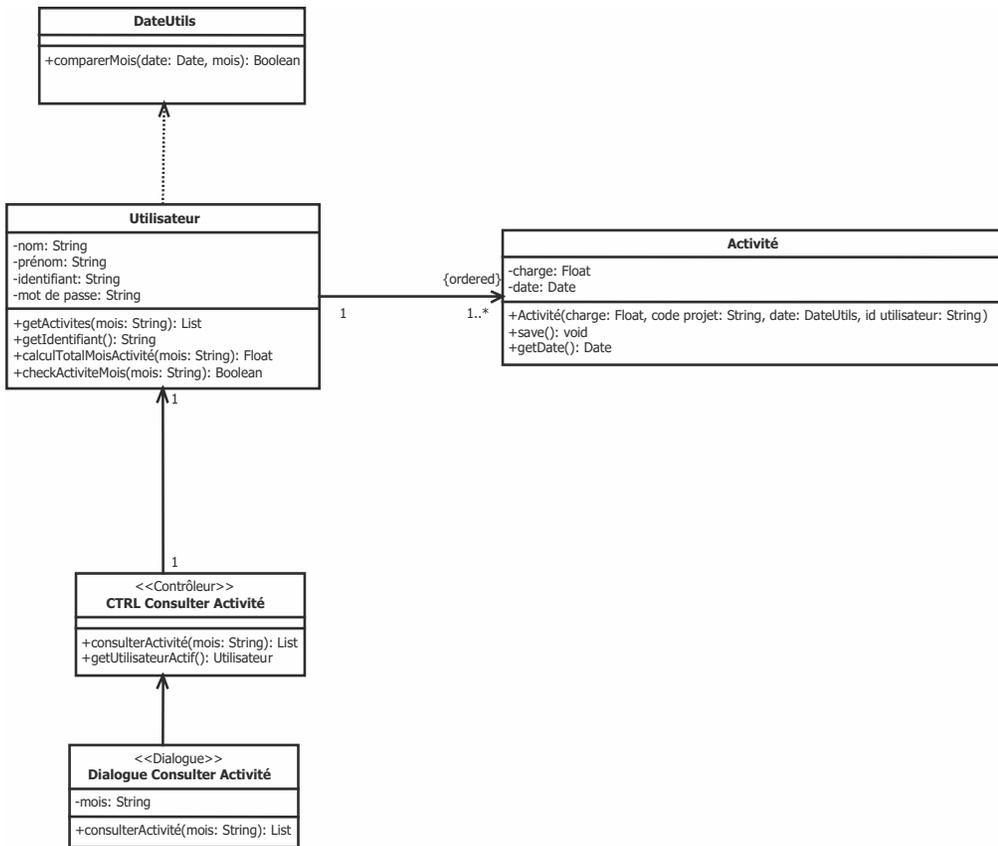


Figure 6.36 — Diagramme de classe technique du CU Consulter activité

Cas d'utilisation « Relancer activité »

L'élaboration des diagrammes de séquence technique (fig. 6.37, fig. 6.38), et l'élaboration du diagramme de classe technique (fig. 6.39) sont réalisées.

Sur le même DSE (fig. 6.38) sont représentés le scénario alternatif « 1a : Relance effectuée avant le 10 du mois en cours » et le scénario alternatif « 1b : Tous les employés ont rempli leur activité ».

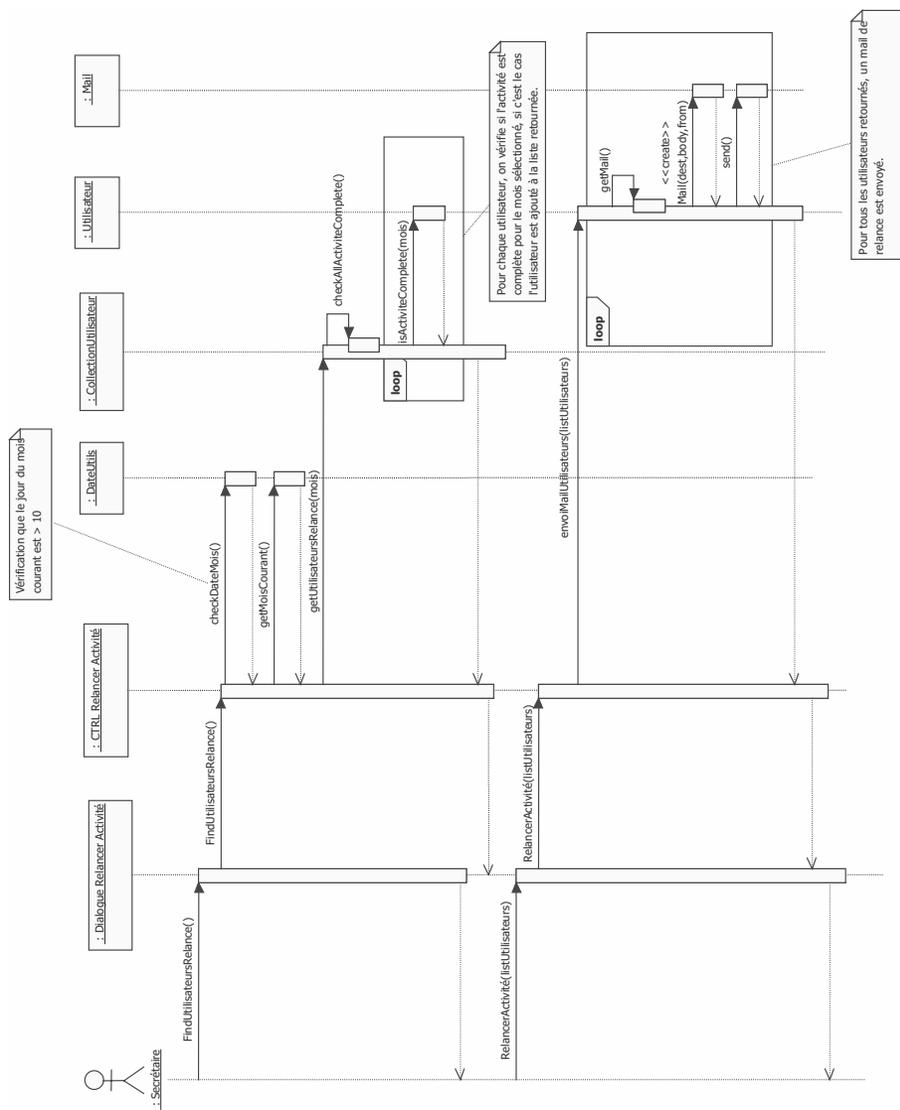


Figure 6.37 — Diagramme de séquence technique du CU Relancer activité (scénario nominal)

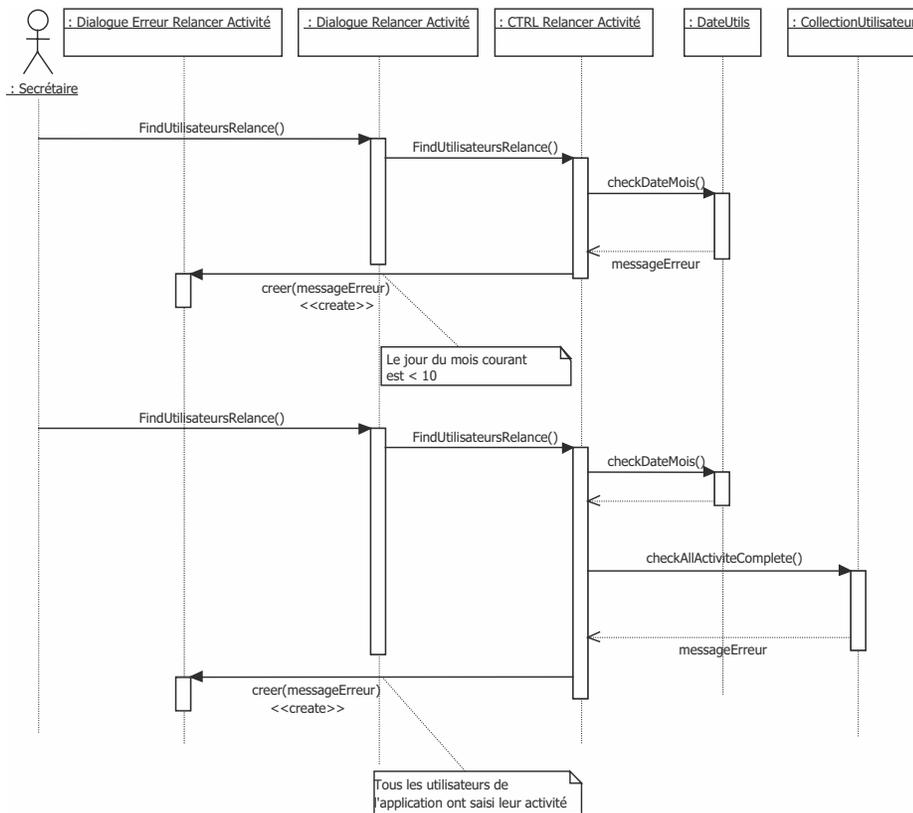


Figure 6.38 — Diagramme de séquence technique du CU Relancer activité (scénarios alternatifs)

La navigabilité des associations est donnée par le sens de circulation des opérations du diagramme de séquence (fig. 6.38). Ainsi, la classe « CTRL Relancer Activité » accède à la classe « Dialogue Erreur Relancer Activité » par le biais du message *creer*. La navigabilité des autres associations est déterminée sur le même principe.

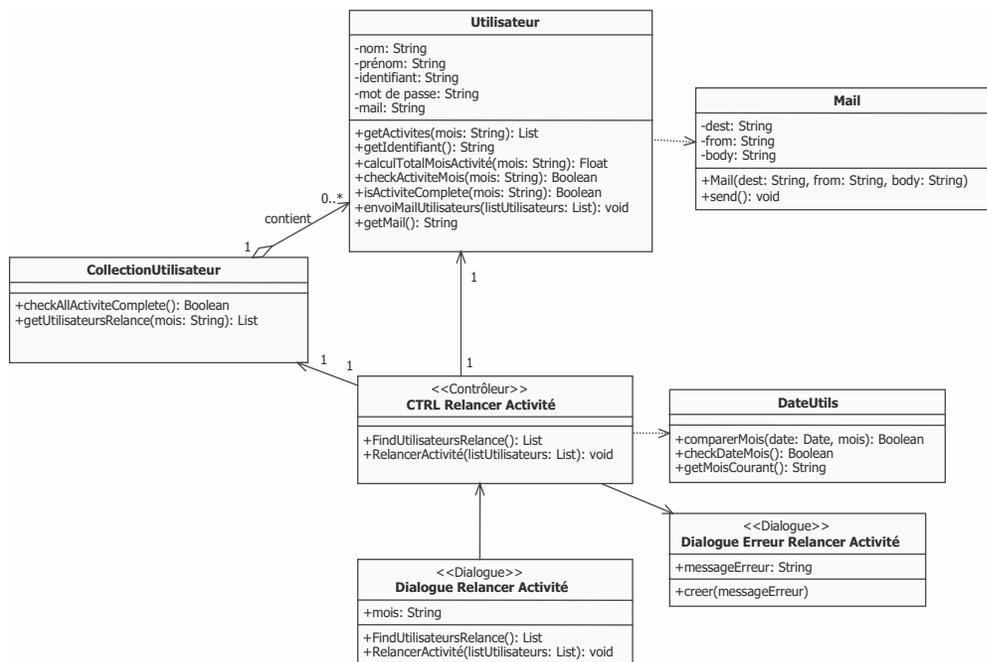


Figure 6.39 – Diagramme de classe technique du CU Relancer activité

Une relation de dépendance est modélisée entre la classe « Utilisateur » et la classe technique « Mail ». On remarquera que le lien entre un objet « Utilisateur » et « Mail » est momentané, il ne dure que le temps d'exécution de la méthode *Mail* et *send*. Ainsi ce n'est pas une association, mais une simple dépendance.

Cas d'utilisation « Consulter tableaux de bord »

L'élaboration des diagrammes de séquence technique (fig. 6.40) et l'élaboration du diagramme de classe technique (fig. 6.41) sont réalisées.

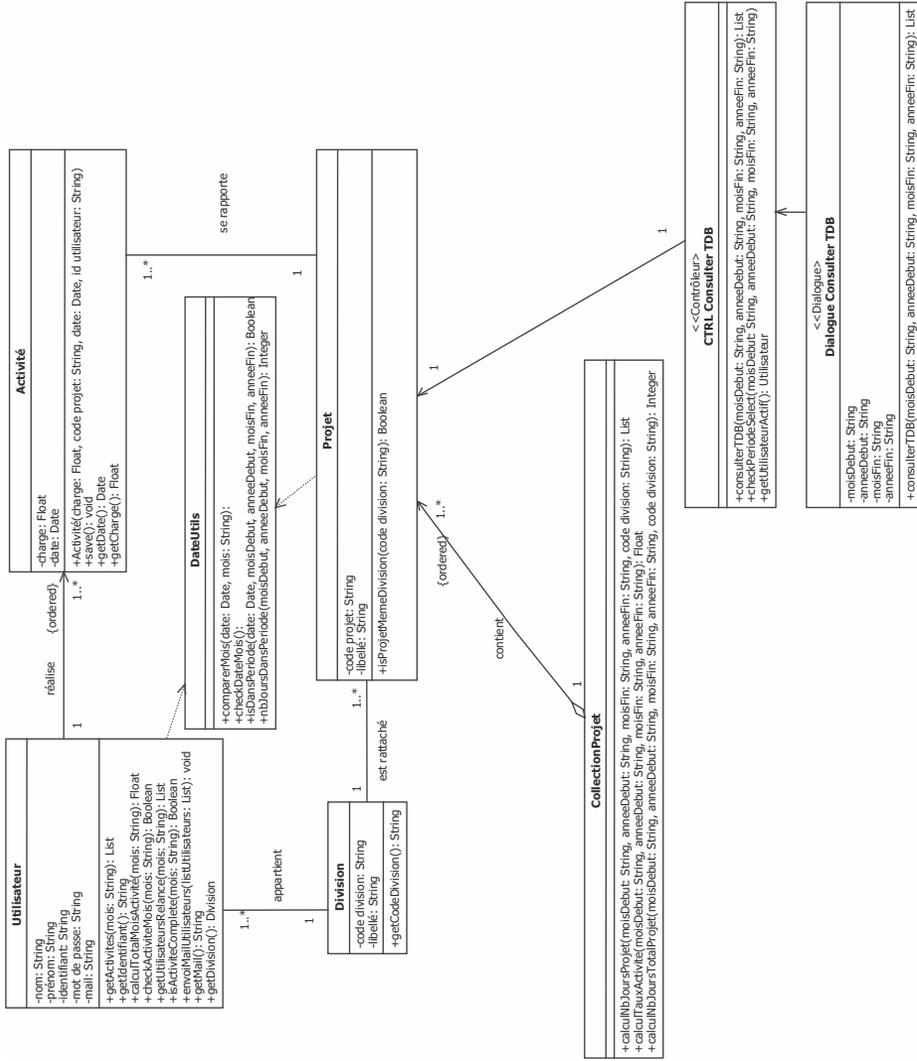


Figure 6.41 — Diagramme de classe technique du CU Consulter tableaux de bord

La navigabilité des associations est donnée par le sens de circulation des opérations du diagramme de séquence précédent. Ainsi, la classe « Dialogue Consulter TDB » accède à la classe « CTRL Consulter TDB » par le biais du message *consulterTDB*. La navigabilité des autres associations est déterminée sur le même principe.

Une relation de dépendance est modélisée entre la classe « Projet » et la classe technique « DateUtils ». On remarquera que le lien entre un objet « Projet » et « DateUtils » est momentané, il ne dure que le temps d'exécution de la méthode *isDansPeriode*. Ainsi ce n'est pas une association, mais une simple dépendance.

Cas d'utilisation « Créer utilisateur »

L'élaboration des diagrammes de séquence technique (fig. 6.42), et l'élaboration du diagramme de classe technique (fig. 6.43) sont réalisées.

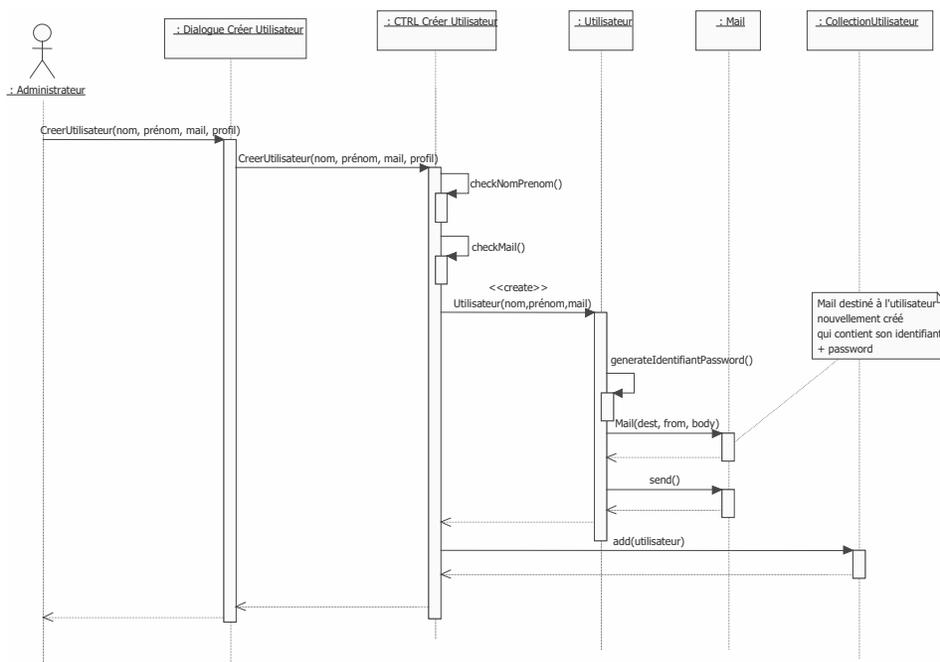


Figure 6.42 — Diagramme de séquence technique du CU Créer utilisateur

La navigabilité des associations est donnée par le sens de circulation des opérations du diagramme de séquence précédent. Ainsi, la classe « Dialogue Créer Utilisateur » accède à la classe « CTRL Créer Utilisateur » par le biais du message *creerUtilisateur*. La navigabilité des autres associations est déterminée sur le même principe.

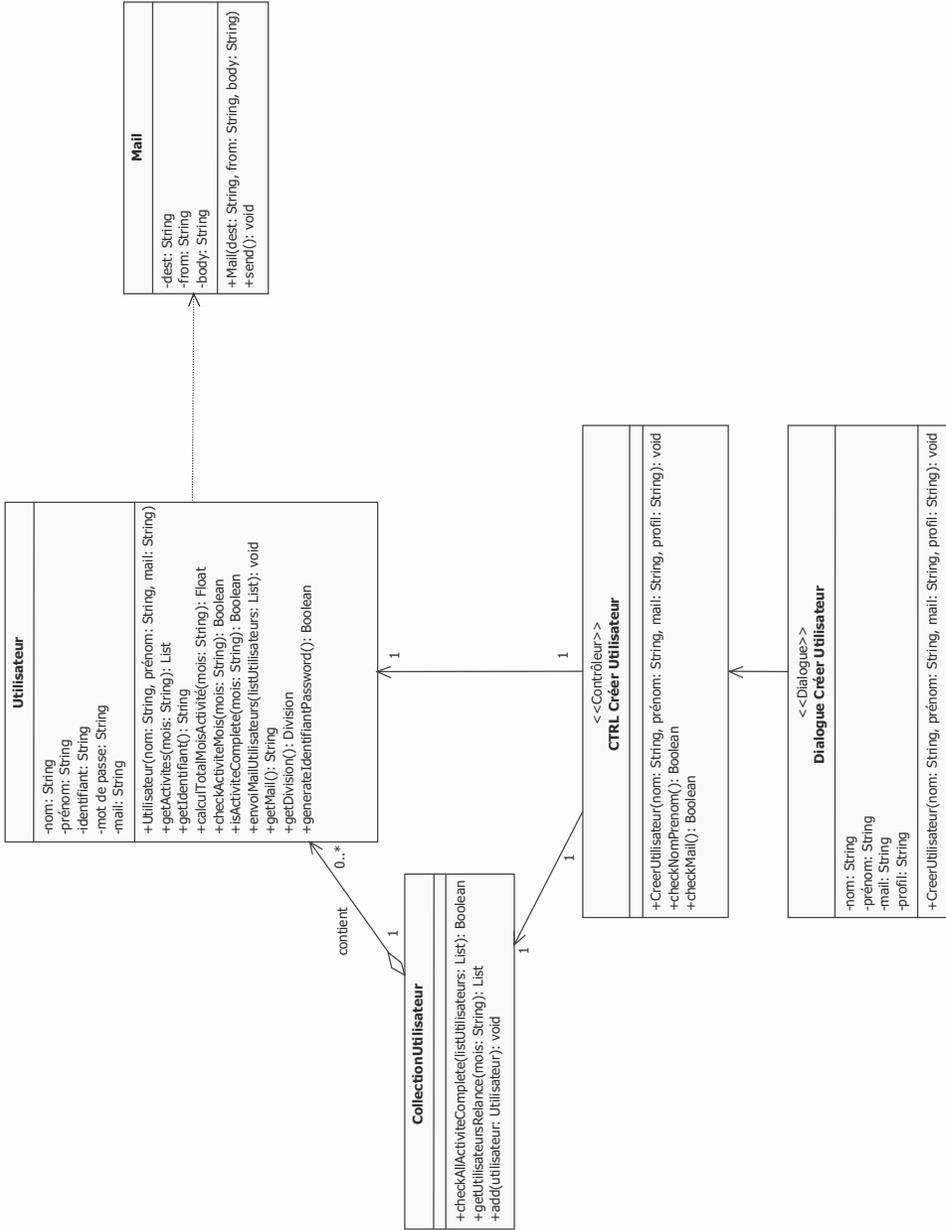


Figure 6.43 — Diagramme de classe technique du CU Créer utilisateur

6.6.2 Élaboration du diagramme de paquetage (FG18)

Tous les paquetages de l'application sont représentés sur la figure 44 :

- **Dialogue** – Paquetage regroupant l'ensemble des classes permettant la gestion des dialogues de l'application :
 - Dialogue Saisir Activité.
 - Dialogue Consulter Activité.
 - Dialogue Relancer Activité.
 - Dialogue Erreur Relancer Activité.
 - Dialogue Consulter TDB.
 - Dialogue Créer Utilisateur.
- **Contrôleur** – Paquetage regroupant l'ensemble des classes permettant la gestion des contrôleurs de l'application :
 - CTRL Saisir Activité.
 - CTRL Consulter Activité.
 - CTRL Relancer Activité.
 - CTRL Consulter TDB.
 - CTRL Créer Utilisateur.
- **Entité** – Paquetage regroupant l'ensemble des classes métiers de l'application. À l'intérieur de ce paquetage, une division en trois sous-paquetages est présente qui correspond à un découpage fonctionnel.

Ces trois sous-paquetages sont les suivants.

- **Gestion des utilisateurs** – Paquetage regroupant l'ensemble des classes permettant la gestion des données de l'utilisateur :
 - CollectionUtilisateur.
 - Utilisateur.
- **Gestion activités et frais** – Paquetage regroupant l'ensemble des classes permettant la gestion des données relatives aux activités et des frais :
 - Frais.
 - Activités.
- **Gestion du référentiel** – Paquetage regroupant l'ensemble des classes pouvant être administrées :
 - CollectionProjet.
 - Projet.
 - Division.
 - Profil.

Deux paquetages techniques sont aussi représentés sur le diagramme.

- **Gestion mail** – Paquetage regroupant l'ensemble des classes techniques permettant la gestion des mails : Mail.
- **Gestion utilitaire** – Paquetage regroupant l'ensemble des classes techniques utilitaires : DateUtils.

Les relations « access » décrites figure 6.44 entre le paquetage Dialogue et Contrôleur et entre le paquetage Contrôleur et Entité illustrent l'indépendance des couches du modèle. En effet, les classes du paquetage Dialogue ont accès à l'espace de nommage du paquetage Contrôleur, mais cet accès ne se transmet pas par transitivité. Les classes du paquetage ont accès à l'espace de nommage du paquetage Entité, mais cet accès ne se transmet pas par transitivité au paquetage Dialogue.

Les relations « import » entre le paquetage Entité et GestionUtilitaire illustrent bien le concept de transitivité. En effet, le paquetage Contrôleur ayant accès à l'espace de nommage du paquetage Entité a lui aussi accès au paquetage Gestion utilitaire.

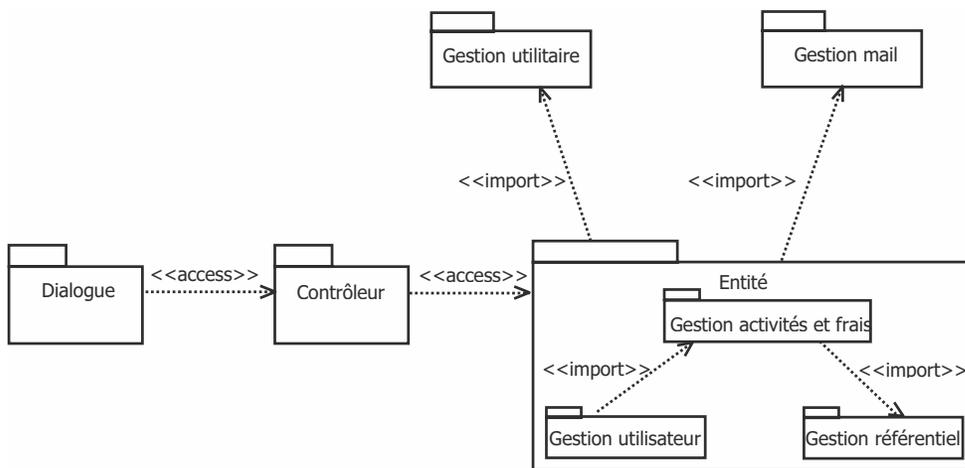


Figure 6.44 – Diagramme de paquetage

Annexes

A. RÉCAPITULATIF DES CONCEPTS D'UML 2

À titre de synthèse, il est proposé dans cette annexe une liste récapitulative des concepts d'UML 2 présentés dans cet ouvrage.

Ce récapitulatif est donné par diagramme après un rappel des concepts communs. Il peut aider le lecteur à mémoriser les différents concepts mis en œuvre dans chaque diagramme.

- **Concepts communs à tous les diagrammes**
 - stéréotype,
 - mot-clé,
 - note,
 - contrainte.
- **Concepts du diagramme de classe (DCL)**
 - attribut,
 - opération,
 - visibilité,
 - association,
 - navigabilité,
 - classe et classe abstraite,
 - rôle,
 - multiplicité,
 - dépendance,

- agrégation et composition
- qualification,
- généralisation et spécialisation,
- héritage.
- **Concepts du diagramme d'objet (DOB)**
 - objet,
 - lien,
 - valeur d'attribut.
- **Concepts du diagramme de composant (DCP)**
 - composant,
 - connecteur,
 - port,
 - interface requise, interface fournie,
 - dépendance,
 - compartiment.
- **Concepts du diagramme de déploiement (DPL)**
 - nœud,
 - unité de traitement (« device »),
 - environnement d'exécution (« executionEnvironment »),
 - artefact,
 - spécification de déploiement.
- **Concepts du diagramme de paquetage (DPA)**
 - espace de nommage,
 - dépendance de type « import »,
 - dépendance de type « access »,
 - dépendance de type « merge ».
- **Concept du diagramme de structure composite (DSC)**
 - collaboration d'instances,
 - rôle (dans la collaboration).
- **Diagramme des cas d'utilisation (DCU)**
 - acteur,
 - interaction,
 - scénario nominal,
 - scénario alternatif,
 - pré et post condition,
 - inclusion, extension, généralisation de cas d'utilisation.
- **Concept du diagramme état-transition (DET)**
 - état,
 - transition,
 - événement,
 - composition d'état (état composite),

- point d'entrée et de sortie,
- point de jonction,
- point de choix,
- état historisé.
- **Concepts du diagramme d'activité (DAC)**
 - action,
 - activité,
 - flot de contrôle,
 - flot de données,
 - nœud de bifurcation,
 - nœud de jonction,
 - nœud de test-décision,
 - nœud de fusion-test,
 - nœud de données,
 - pin d'entrée et de sortie,
 - partition (couloir d'activités).
- **Concepts du diagramme de séquence (DSE)**
 - ligne de vie,
 - message synchrone et asynchrone,
 - fragment d'interaction,
 - opérateur,
 - interaction,
 - opérateur alt,
 - opérateur opt,
 - opérateur loop,
 - opérateur par,
 - opérateurs strict/weak,
 - opérateur break,
 - opérateurs ignore/consider,
 - opérateur critical,
 - opérateur negative,
 - opérateur assertion,
 - opérateur ref.
- **Concepts du diagramme de communication (DCO)**
 - rôle,
 - état,
 - message.
- **Concepts du diagramme global d'interaction (DGI)**
 - ceux du DAC et les fragments d'interaction du DSE.
- **Concepts du diagramme de temps (DTP)**
 - ligne de temps,
 - états (temporels).

B. RÉCAPITULATIF DE LA DÉMARCHE UP7

Un récapitulatif des activités et des fiches guides de la démarche UP7 proposées dans cet ouvrage est donné dans cette annexe.

- **Liste des activités de la démarche**
 - 1 – Modélisation métier
 - 2 – Exigences fonctionnelles
 - 3 – Analyse des cas d'utilisation
 - 4 – Synthèse de l'analyse
 - 5 – Conception
 - 6 – Implémentation
 - 7 – Test

- **Liste des fiches guides associées aux activités**
 - FG1 – Élaboration du schéma de contexte du domaine d'étude
 - FG2 – Élaboration du diagramme d'activité
 - FG3 – Élaboration du diagramme de classe métier
 - FG4 – Élaboration du diagramme des cas d'utilisation système
 - FG5 – Élaboration des diagrammes de séquence système
 - FG6 – Élaboration du schéma de navigation générale
 - FG7 – Élaboration du diagramme des cas d'utilisation
 - FG8 – Description des cas d'utilisation
 - FG9 – Élaboration des diagrammes de séquence
 - FG10 – Élaboration des diagrammes d'état-transition
 - FG11 – Élaboration des interfaces utilisateur
 - FG12 – Élaboration des diagrammes de classe
 - FG13 – Élaboration du diagramme de classe récapitulatif
 - FG14 – Élaboration de la matrice de validation
 - FG15 – Réalisation des choix techniques
 - FG16 – Élaboration des diagrammes de séquence technique
 - FG17 – Élaboration des diagrammes de classe technique
 - FG18 – Élaboration du diagramme de paquetage

Bibliographie

- [Barbier2005] Franck BARBIER, *UML 2 et MDE*, Dunod, 2005
- [Bersini2004] Hughes BERSINI, *L'orienté Objet*, Eyrolles, 2004.
- [Blaha2002] Michael BLAHA, James RAMBAUGH – *Modélisation et conception orientées objet avec UML 2*, Pearson Education, 2005.
- [Booch1994] Grady BOOCH, *Object-Oriented Analysis and Design with Application*, Addison-WESLEY, 1994.
- [Cockburn2001] Alistair COCKBURN, *Rédiger des cas d'utilisation efficaces*, Eyrolles, 2001.
- [Fowler2004a] Martin FOWLER, *UML 2.0*, Campus Press, 2004.
- [Fowler2004b] Martin FOWLER, *Initiation aux aspects essentiels de la notation*, Campus Press, 2004.
- [Jacobson1992] Ivar JACOBSON, Grady BOOCH, James RUMBAUGH, *The Unified Modeling Language, Reference Manual*, Addison-WESLEY, 1992.
- [Jacobson2000a] Ivar JACOBSON, Grady BOOCH, James RUMBAUGH, *Le Processus unifié de développement logiciel*, c/o Eyrolles, 2000.
- [Jacobson2000b] Ivar JACOBSON, Grady BOOCH, James RUMBAUGH, *Le guide utilisateur UML*, Eyrolles 2000.
- [Kruchten2000] Philippe KRUCHTEN, *The rational Unified Process An introduction*, Addison-WESLEY 2000.
- [Muller2000] Pierre-Alain MULLER, Nathalie GAETNER, *Modélisation objet avec UML*, Eyrolles, 2000.

- [Meyer2000] Bertrand MEYER, *Conception et programmation orientées objet*, Eyrolles, 2000.
- [OMG2007] OMG (Object Management Group), *UML 2.0 Superstructure*, <http://www.uml.org>, 2007.
- [Rumbaugh1991] James RUMBAUGH, Michael BLAHA, William PREMERLANI, Frederick EDDY, William LORENSEN, *Object-Oriented Modeling and Design*, Prentice Hall International, 1991.
- [Rumbaugh2004] James RUMBAUGH, Ivar JACOBSON, Grady BOOCH, *UML 2.0 Guide de référence*, CampusPress, 2004.
- [Scott2004] Ambler SCOTT, *The elements of UML 2.0 style*, Cambridge Press, 2004.

Index

A

acteur 62
action 73, 81
activité 73, 82, 113, 128
agrégation 3, 27, 28
analyse 116
 des cas d'utilisation 125, 134
artefact 51, 53, 114
association 3, 23
attribut 19, 22

B-C

boîte
 blanche 48
 noire 47
cas d'utilisation 62
classe 2, 18
 abstraite 33
classe-association 27
collaboration 58
compartiment 48
composant 46
composition 28
conception 117, 126, 142, 146
connecteur
 d'assemblage 47
 d'interfaces 47
construction 116
contrainte 10, 26
 temporelle 92

D

dépendance 30
 d'interfaces 47
 entre paquetages 56
déploiement 121
diagramme
 d'activité 11, 80
 d'état-transition 11, 72, 73
 d'objet 11
 de classe 11, 17
 de communication 12, 104
 de comportement 11
 de composant 11, 46
 de déploiement 11, 50
 de paquetage 11, 54
 de séquence 11, 90

de structure composite 11, 58
de temps 12, 109
des cas d'utilisation 11, 61
global d'interaction 12, 106
structurel 11

E

élaboration 115
encapsulation 3
environnement 122
état 2, 72
 composite 75
 historique 77
exigences fonctionnelles 125, 131
expression des besoins 116

F

fiche guide 128
flot
 de contrôle 81
 de données 86
fragment d'interaction 93

G

généralisation 4, 32
gestion
 d'un projet 122
 de l'environnement 122
 de la configuration et des changements
 122
 des exigences 121

H-I

héritage 32
implémentation 117
inception 115
incrémental 112
instance 18
interaction 63
interface 31, 47
itératif 112
itération 116

J-L

jalon 119
ligne de vie 91

M

message 91, 104
 méta-modèle 8
 modélisation métier 121, 124, 128
 modularité 6
 multiplicité 24

N

navigabilité 25
 nœud 50, 53
 de bifurcation 82
 de fusion-test 85
 de jonction 83
 de test-décision 84
 note 10

O

objet 2, 17, 21, 72, 105
 OCL 10
 opérateur
 alt 94
 assertion 100
 break 97
 consider 98
 critical 98
 ignore 98
 loop 94
 negative 100
 opt 94
 par 96
 ref 100
 strict 97
 weak 97
 opération 20, 22

P

paquetage 22, 54
 partition 87
 persistance 5
 phase 114
 pin 86
 point
 de choix 76

 de jonction 76
 polymorphisme 4
 port 49
 post condition 67
 pré condition 67
 privé 22
 processus 112, 119
 profil 9
 protégé 22
 public 22

Q-R

qualification 30
 qualité 118
 relation
 d'extension 65
 d'inclusion 64
 de généralisation 65
 réutilisabilité 6
 risques 113
 rôle 24, 58, 104, 113
 RUP 117

S

scénario
 alternatif 67
 nominal 67
 spécialisation 4
 spécification de déploiement 51
 stéréotype 9, 36
 synthèse de l'analyse 125, 140

T-U

tests 117
 transistion 116
 transition 72, 80, 81
 UML 7
 UP 8, 111
 UP7 123

V-W

valeur marquée 9
 visibilité 22
 workflow 114, 119



MANAGEMENT DES SYSTÈMES D'INFORMATION

APPLICATIONS MÉTIERS

▶ ÉTUDES, DÉVELOPPEMENT, INTÉGRATION

EXPLOITATION ET ADMINISTRATION

RÉSEAUX & TÉLÉCOMS

Joseph Gabay
David Gabay

UML 2 ANALYSE ET CONCEPTION

Mise en œuvre guidée avec études de cas

Cet ouvrage s'adresse à tous les professionnels, concepteurs et développeurs, qui souhaitent mieux maîtriser UML 2 et acquérir une démarche pratique de mise en œuvre ainsi qu'aux étudiants en informatique.

Il propose une approche pédagogique de l'aspect normatif d'**UML 2** et une démarche d'élaboration des diagrammes couvrant l'analyse et la conception des systèmes d'information. Le lecteur suit un apprentissage progressif fondé sur de nombreux **exemples, exercices corrigés** et de véritables **études de cas** se rapprochant de projets réels d'entreprise.

Cette édition sert trois objectifs :

- présenter les **treize diagrammes d'UML 2** en conciliant le respect strict de la norme avec une application centrée sur les SI des entreprises ;
- décrire l'**analyse** et la **conception** des SI à l'aide des diagrammes d'UML 2 en s'appuyant sur des exemples et des exercices **adaptés au contexte professionnel** ;
- proposer une **démarche de mise en œuvre** d'UML 2 structurée en phases et activités, décrite à l'aide de **fiches guides** et illustrée par deux études de cas détaillées.

JOSEPH GABAY

est directeur de projet informatique au CNRS. Il assure également des enseignements d'UML à l'université de Paris-Dauphine ainsi qu'en IUT en cycle BTS. Il est l'auteur de plusieurs ouvrages sur UML.

DAVID GABAY

est ingénieur et chef de projet chez CapGemini.