COURS DE MODELISATION OBJET : UML

Tshikutu Anaclet Bikengela

COURS DE MODELISATION OBJET : UML

A L'usage des étudiants de
TROISIEME GRADUAT EN SCIENCES INFORMATIQUES

Par
CT TSHIKUTU BIKENGELA ANACLET
E-mail : tshikutuanaclet2013@gmail.com
Tél : +243 812382300

UNIVERSITE JOSEPH KASA-BUVU
FACULTE DES SCIENCES INFORMATIQUES

OBJECTIFS DU COURS

Ce cours a pour principal objectif de parcourir les différentes approches, techniques et méthodes utilisées dans la conception Objet de systèmes d'informations (à fort impact sur la numérisation de systèmes d'entreprise, …).

Il doit permettre à l'apprenant de :

- Comprendre les notions approfondies de systèmes d'informations,
- Connaître les différentes méthodes de conception de systèmes d'informations,
- Maîtriser les différents enchaînements d'activités du processus unifié (UML) pour le développement d'un système informatisé,
- Maitriser les outils du marché en passant par les applications concrètes.

MODES D'EVALUATION

Le cours sera évalué de la manière suivante :

- Sous forme de travaux réalisés pendant l'année (Présence aux cours, aux T.P.)
- Sous forme d'exposés (Par groupe d'étudiants, les étudiants choisissent un thème, le développent et l'exposent avant les examens)
- Sous forme d'interrogation et d'examen oral et / ou écrit

CONTENU

REFERENCES BIBLIOGRAPHIQUES

1. C.J. DATE, *Introduction aux bases de données*, International Thomson Publishing, traduction de la septième édition de l'ouvrage d'Addison-Wesley, 2000.
2. C. SOUTOU, « Inference of Aggregate Relationships through Database Reverse Engineering », *Proceedings of the 17th International Conference on Conceptual Modeling (ER'98),* Lecture Note in Computer Science, Editions Springer Verlag, volume 1507, pages 135-149.
3. D. DIONISI, *L'essentiel sur UML*, Eyrolles, 2e édition, 1998.
4. E.F. CODD, « A Relational Model for Large Shared Data Banks », *Communications of the ACM*, Vol 13, N°6, 1970.
5. G. BOOCH, I JACOBSON, J. RUMBAUGH, *Le guide de l'utilisateur UML,* Eyrolles, 2000.
6. H. TARDIEU, D. NANCI, D. PASCOT, « A Method, A Formalism and Tools for Database Design (three years of Experimental Practice) », *Proceedings of the International Conference on Entity-Relationship Approach to Systems Analysis and Design,* 1979.
7. H. TARDIEU, D. NANCI, D. PASCOT, C*onception d'un système d'information, Conception de la base de données*, Les Éditions d'Organisation, 1979.
8. H. TARDIEU, A. ROCHFELD, R. COLLETI, *La méthode MERISE tome 1*, Eyrolles, 1991.
9. J. AKOKA, I. COMYN-WATIAU, Conception des bases de données relationnelles, Vuibert, 2001.
10. P. VALDURIEZ, « Objets complexes dans les systèmes de bases de données relationnels », *Techniques et science informatique*, Vol. 6, N° 5, 1987.
11. Rapport introductif *Modèles de structure de données dans les systèmes d'information*, Séminaire international, Namur 1974.
12. R. ELMASRI, S. NAVATHE, *Conception et architecture des bases de données*, Pearson Education, 2004.
13. T. CONNOLY, C. BEGG, *Systèmes de bases de données*, Reynald Goulet-Eyrolles, 2005.

CHAPITRE I. LES NOTIONS APPROFONDUES DE SYSTEMES D'INFORMATIONS

I.1. INTRODUCTION

L'information comme élément de connaissance susceptible d'être codé pour être conservé, traité ou communiqué, est indispensable au fonctionnement d'une entreprise. Il est nécessaire de traiter les diverses données, d'origine interne et externe, pour les adapter aux besoins des utilisateurs.

A chaque instant, pour l'ensemble de ses activités, l'entreprise utilise des informations. Ces informations doivent être préparées et traitées pour être utilisables.

Le Système d'information a pour objectif de restituer aux différents membres de l'entreprise, les informations sous une forme directement utilisable, au moment opportun, afin de faciliter le déroulement des opérations et la prise de décision aux différents niveaux.

Le Système d'information est en partie formel et en partie informel.

Le système d'information formel est défini dans l'organisation par des procédures codifiées, des documents et des messages rigoureusement spécifiés. L'information y est véhiculée essentiellement sous forme écrite.

Le système d'information informel traite et communique de l'information, souvent importante, sans laisser de traces systématiques et sans appliquer des règles rigoureuses connues et stables. C'est le domaine de l'improvisation, de la relation personnelle… voir du secret.

De plus en plus, les opérations correspondant au système d'information formel sont automatisées grâce à l'utilisation d'ordinateur. Un système informatique comprend des matériels (micro, mini ou gros ordinateurs et périphériques) et des logiciels (programmes nécessaires pour le traitement des données). Ce système informatique est une composante majeure du système d'information formel de l'entreprise.

La notion de système d'information ne fait qu'évoluer. Elle a connu depuis plus de trente ans une évolution très importante et régulière, tout particulièrement sous l'effet des transformations technologiques et l'émergence des moyens informatiques et télématiques qui peuvent être envisagés et mobilisés pour construire et faire fonctionner le nouveau système d'information.

Trois grandes étapes marquent principalement cette évolution. Chacune de ces étapes traduit une importante extension de la compréhension du concept. Ces trois étapes correspondent :

a. Au Système d'information informatisé qui prend en charge les tâches administratives les plus courantes et très répétitives ;
b. Au Système d'information informatisé qui contribue à informer le manager opérationnel sur le fonctionnement courant ;
c. Au Système d'information informatisé qui assiste le manager dans la prise de décisions opérationnelles.

Dans le cadre de ce cours, nous allons nous intéresser plus au type (b).

Relativement à l'organisation et la direction d'un système d'information, celui-ci se construit autour de processus « métiers » et ses interactions, et non simplement autour de bases de données ou de logiciels informatiques. Le système d'information doit réaliser l'alignement stratégique et la stratégie d'entreprise par un management spécifique.

La gouvernance des systèmes d'information ou gouvernance informatique (IT gouvernance) renvoie au moyen de gestion et de régulation des systèmes d'information mis en place par une organisation dans le but d'atteindre ses objectifs. A ce titre, la gouvernance du système d'information fait partie de la gouvernance de l'organisation. Les méthodes ITIL (IT infrastructure Library) et COBIT (Control Objectives for Business & Related Technology) sont par exemple des supports permettant de mettre un système d'information sous contrôle et de le faire évoluer en fonction de la stratégie de l'organisation.

I.2. DEFINITIONS, FONCTIONS ET QUALITES D'UN SI

Une organisation, en tant que système, peut être décomposée en sous-système opérant et sous-système de pilotage reliés par un système d'information faisant le trait d'union. Une telle décomposition prend bien en compte :

- la différence de besoin en matière d'information des modules opérants et pilotes,
- la nécessité pour le système d'information de ne pas se contenter de transmettre les informations mais d'en changer le niveau de synthèse.

Par système d'information d'une organisation, on entend un ensemble formé :

- de procédés pour l'acquisition, la mémorisation, la transformation, la recherche, la communication et la restitution des renseignements.
- de ressources humaines et de moyens techniques intégrés dans un système coopérant et contribuant à son fonctionnement et à la poursuite des objectifs qui lui sont assignés.

Il englobe aussi un modèle de fonctionnement de cette dernière à travers d'une part, les règles qui régissent la vie des renseignements, et d'autre part, les principes administratifs de constitution, de transformation et de diffusion des documents. Il faut le voir comme un miroir sur lequel on peut observer :

- La réalité décrite dans ses multiples facettes et projetée avec des réductions temporelles et spatiales inhérente à la nature du Système d'information ;
- Une abstraction simplificatrice qui analyse la réalité en termes de catégories ou de types.

Le Système d'information est considéré comme un artéfact (phénomène artificiel) greffé sur l'objet réel qui est l'entreprise, volontairement conçu et mis en place pour remplir des fonctions de représentation, de mémorisation et de communication, en vue d'atteindre des objectifs informationnels.

Cet artéfact supporte un réseau de flux d'information nécessaire pour organiser, mettre en œuvre, gérer et maintenir les activités d'une organisation. C'est un instrument de communication qui sert aux échanges informationnels entre les partenaires et l'organisation et accroit leur efficacité.

Un bon système d'information doit remplir comme fonctions : le recueil (Collecte et saisie) des informations, la mémorisation des informations brutes ou résultats de traitement, le traitement et la circulation des informations.

Un bon système d'information doit posséder des qualités ci-après : la fiabilité, la rapidité, la pertinence et la sécurité.

I.3. DEVELOPPEMENT D'UN SYSTEME D'INFORMATION

I.3.1. Problématique de la conception du système d'information

L'organisation qui entame un nouveau cycle de vie pour son système d'information doit tout décider. Dans le processus de développement du système d'information, objet complexe, la conception est la tâche la plus créative mais aussi la plus difficile. Les nombreuses difficultés découlent directement ou indirectement de la nature du travail de conception, du nombre et de la variété des problèmes qu'il faut résoudre et de la diversité des compétences qu'il faut réunir pour les traiter.

Les contraintes de l'information ont au départ et pendant longtemps pesés sur ces choix. C'est ainsi que des systèmes techniquement satisfaisants mal adaptés à l'organisation d'accueil ont vraiment tentés en vain de devenir opérationnels. L'évolution des techniques, la variété de choix qu'elles créent permettent de se libérer de l'emprise technologique et de reconnaitre l'importance de l'examen des besoins, de l'analyse conceptuelle et l'intérêt de susciter une interaction positive entre professionnels de la technologie et les utilisateurs, et en final une participation plus active de ces derniers à l'expression de besoins et au choix des solutions.

Les difficultés de conception font appel à la réponse **méthodologique** ou aux méthodes d'analyse. Certaines méthodes disparaissent laissant la place à d'autres méthodes plus

adaptées, et d'autres évoluent dans le temps en fonction des différentes technologies. Chaque méthode a ses qualités et ses défauts. Il est donc parfois utile et nécessaire en fonction de l'étape d'analyse du projet d'appliquer des méthodes différentes.

Chaque méthode est adaptée au type de projet (objet, industrielle, gestion) et aux outils (SGBD, L3G …).

Quelques méthodes connues sont : MERISE et MERISE/2 des méthodes systémiques, SADT (Structured Analysis and Design Techniques) une méthode cartésienne, SSADM (Structured System Analysis and Design Method) une méthode développée en Grande Bretagne, UML qui n'est pas tout à fait une méthode mais un langage de modélisation unifiée, OOA (Object Oriented Analysis), OMT (Object Modeling Technique), SART …

I.3.2. Approche particulière

Développer un système d'information, c'est mener un ensemble d'activités complexes et variées de décisions et de planifications, d'analyse de la réalité et de recueil de besoins informationnels, de conception et d'évaluation de système, de réalisation, de tests, de mise en œuvre et de contrôle de qualité qui aboutissent à faire passer le système d'information d'une situation initiale de profit, à une situation finale de système concret et opérationnel.

Le travail du concepteur de Système d'information est analogue au travail de l'architecte qui conçoit et dessine les plans d'un immeuble. Comme lui, le concepteur de Système d'information par sa spécification, à la fois, définit le futur Système d'information, décide de la solution technologique et prépare les tâches de réalisation.

Il est désormais professionnellement admis que le processus de développement d'un Système d'information s'organise globalement en deux phases. Première phase de conception qui aboutit à la spécification du Système d'information sous la forme d'un schéma, et une grande phase de réalisation, qui conduit à la fabrication effective du Système d'information, comme l'illustre la figure ci-après.

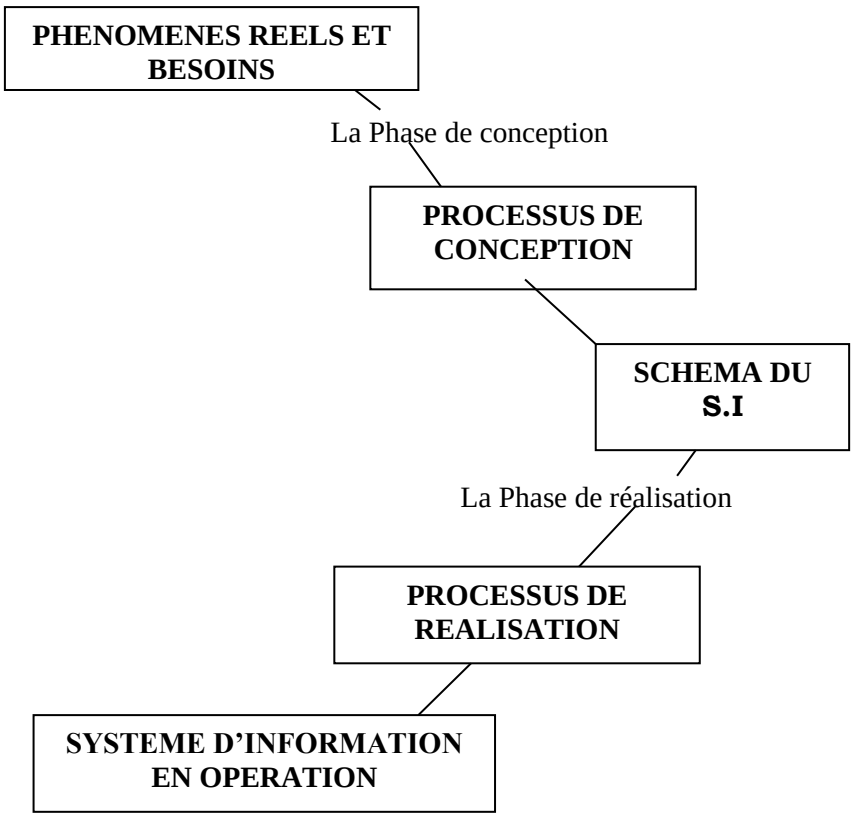


Fig. 2 – Développement d'un Système d'information

I.3.3. La méthode de conception

La méthode de conception est utile et nécessaire pour :

- Aider à clairement et complètement formuler le problème informationnel qui est posé et à maitriser la résolution, en s'appuyant sur des critères objectifs pour évaluer les solutions ;
- Construire des systèmes d'information pertinents, complets, cohérents, fiables, flexibles et adaptatifs ;
- Maitriser la complexité du problème informationnel à résoudre ;
- Permettre d'évaluer le système en tout moment de son cycle de vie, tant sur le plan de son efficacité technique que sur celui de sa pertinence à satisfaire les besoins des gestionnaires ;
- Substituer à la construction trop individuelle des Systèmes d'information, une conception concrète basée sur une coopération efficace entre concepteur, informaticien, gestionnaire et utilisateurs ;
- Permettre la communication entre participants de l'équipe de conception ;
- Maitriser et réduire les coûts et les détails, accroitre la productivité et la qualité des activités de développement.

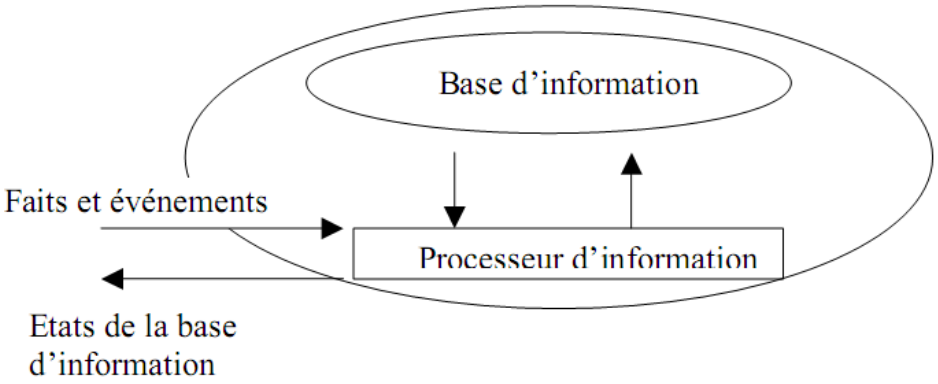
Les entreprises reconnaissent aujourd'hui la nécessité de recourir à une méthode de conception. Les méthodes proposées sont nombreuses et variées. Leur tendance est de

s'intéresser à tout le cycle de vie du Système d'information et non pas seulement à certaines étapes.

En dépit de cette variété, toute méthode doit mettre en œuvre quatre composantes indissociables et complémentaires : Des modèles, des langages, une démarche et des outils.

I.4. ARCHITECTURE D'UN SYSTEME D'INFORMATION

Le système d'information doit décrire le plus fidèlement possible le fonctionnement du système opérant. Pour ce faire, il doit intégrer une base d'information dans laquelle seront mémorisés la description des objets, des règles et des contraintes du système opérant. Cette base étant sujette à des évolutions, le système d'information doit être doté d'un mécanisme (appelé processeur d'information) destiné à piloter et à contrôler ces changements. Le schéma suivant synthétise l'architecture d'un système d'information.



Le processeur d'information produit des changements dans la base d'information à la réception d'un message. Un message contient des informations et exprime une commande décrivant l'action à entreprendre dans la base d'information. Le processeur d'information interprète la commande et effectue le changement en respectant les

00contraintes et les règles.

Si le message exprime une recherche sur le contenu de la base d'information, le processeur interprète la commande et émet un message rendant compte du contenu actuel de la base d'information. Dans tous les cas, l'environnement a besoin de connaître si la commande a été acceptée ou refusée. Le processeur émet, à cet effet, un message vers l'environnement.

Relativement à la conception d'un système d'information, l'architecture présentée ci-dessus induit une double conception :

- celle de la base d'information (aspect statique) ;

- celle du processeur de traitement (aspect dynamique).

Pour aider le concepteur dans ces deux tâches, la méthode Merise pour sa part, propose un ensemble de formalismes et de règles destinés à modéliser de manière indépendante les données et les traitements du système d'information.

I.5 LES MODELES CLASSIQUES DE CYCLE DE VIE D'UN SYSTEME

Comme pour toutes les fabrications, il est important d'avoir un procédé de fabrication du logiciel défini et explicitement décrit et documenté. Il s'agit d'un type de fabrication un peu particulier : en un seul exemplaire, car la production en série est triviale (recopie). Les modèles de cycle de vie du logiciel décrivent à un niveau très abstrait et idéalisé les différentes manières d'organiser la production. Les étapes, leur ordonnancement, et parfois les critères pour passer d'une étape à une autre sont explicités (critères de terminaison d'une étape – revue de document, critères de choix de l'étape suivante, critères de démarrage d'une étape).
Il faut souligner la différence entre étapes (découpages temporel) et activités (découpage selon la nature du travail). Il y a des activités qui se déroulent dans plusieurs étapes (ex : la spécification, la validation et la vérification), voire dans toutes les étapes (ex : la documentation).

Rappelons aussi la différence entre vérification et validation (B. Boehm, 76) :

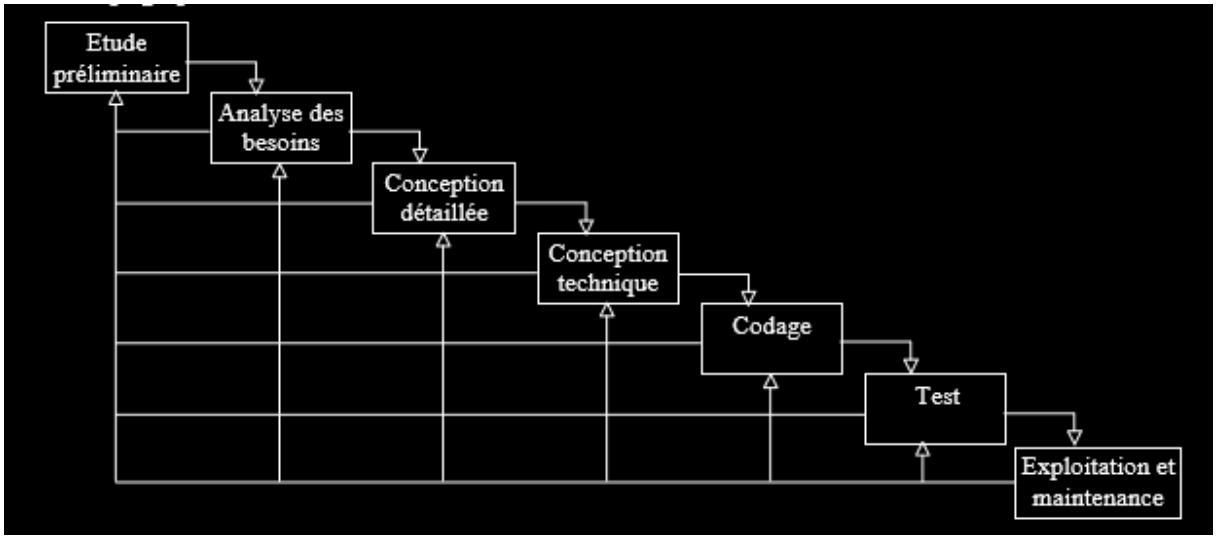
- Vérification: « are we building the product, right? " ("Construisons nous le produit correctement ?", correction interne du logiciel, concerne les utilisateurs),

- Validation : « are we building the rigth product » (« construisons nous le bon produit ? », adaptation vis à vis des besoins des utilisateurs, concerne les utilisateurs).

I.5.1. MODELE EN CASCADE

C'est le tout premier modèle classique du cycle de vie. D'autres noms sont parfois donnés à ses phases et un découpage plus fin est souvent utile. Le cycle de vie dit de la « cascade » date de 1970, il est l'œuvre de ***Royce***. Ce Cycle de vie est linéaire (présentoir) sans aucune évaluation entre le début du projet et la validation. *Ici, Le projet est découpé en phases successives dans le temps et à chaque phase correspond une activité principale bien précise produisant un certain nombre de livrables et On ne passe à l'étape suivante que si les résultats de l'étape précédente sont jugés satisfaisants*.
L'activité d'une étape se réalise avec les résultats fournis par l'étape précédente ; ainsi, chaque étape sert de contrôle du travail effectué lors de l'étape précédente et Chaque phase ne peut remettre en cause que la phase précédente ce qui, dans la pratique, s'avère insuffisant. L'élaboration des spécifications est une phase particulièrement critique : les erreurs de spécifications sont généralement détectées au moment des tests, voire au

moment de la livraison du logiciel à l'utilisateur. Leur correction nécessite alors de reprendre toutes les phases du processus. Ce modèle est mieux adapté aux petits projets ou à ceux dont les spécifications sont bien connues et fixes. Dans le modèle en cascade, on effectue les différentes étapes du logiciel de façon séquentielle. Les interactions ont lieu uniquement entre étapes successives : on s'autorise des retours en arrière uniquement sur l''étape précédente. Par exemple, un test ne doit pas remettre en cause la conception architecturale.



1) Etude préliminaire ou étude de faisabilité : Elle concerne la définition globale du problème. Il faut étudier la stratégie de l'entreprise et décider de la nécessité de fabriquer ou acheter un nouveau produit.
2) Analyse des besoins (quoi faire ?) : identifier les besoins de l'utilisateur (=> produire le cahier de charges)
3) Conception détaillée (comment faire) : organiser les besoins de l'utilisateur dans différents modules du système, faire la description détaillée des fonctions de chaque module en vue de l'implémentation.
4) Conception technique (Avec quoi ?) : étudier le contexte d'implémentation (matériels, outils logiciels, …)
5) Codage : coder chaque module et le tester indépendamment des autres (test unitaires)
6) Test : Intégrer les différents modules et les tester dans l'ensemble par rapport aux besoins de l'utilisateur.
7) Après l'installation suit la phase d'exploitation et de maintenance (operation and maintenance phase). Le logiciel est maintenant employé dans son environnement opérationnel, son comportement est surveillé et, si nécessaire, il est modifié. Cette dernière activité s'appelle la maintenance du logiciel (software maintenance).

Il peut être nécessaire de modifier le logiciel pour corriger des défauts, pour améliorer ses performances ou autres caractéristiques, pour adapter le logiciel à un nouvel environnement ou pour répondre à des nouveaux besoins ou à des besoins modifiés.
On peut donc distinguer entre la maintenance corrective, la maintenance perfective et la maintenance adaptative. Sauf pour des corrections mineures, du genre dépannage, la maintenance exige en fait que le cycle de développement soit réappliqué, en général sous une forme simplifiée.
Voici les différentes formes de maintenance qui peuvent exister en génie logiciel :

- *La Maintenance corrective* : c'est une maintenance qui Corrige les erreurs et les défauts d'utilité, d'utilisabilité, de fiabilité... cette maintenance Identifie également les défaillances, et les dysfonctionnements en localisant la partie du code responsable. Elle Corrige et estime l'impact d'une modification. Attention, La plupart des corrections introduisent de nouvelles erreurs et Les coûts de correction augmentent exponentiellement avec le délai de détection. Bref, la maintenance corrective donne lieu à de nouvelles livraisons (release).
- *La Maintenance adaptative* : c'est une maintenance qui ajuste le logiciel pour qu'il continue à remplir son rôle compte tenu du l'évolution des Environnements d'exécution, des Fonctions à satisfaire et des Conditions d'utilisation. C'est par exemple le changement de SGBD, de machine, de taux de TVA, an 2000 ...
- *La Maintenance perfective et d'extension* : c'est une maintenance qui sert à accroître et améliorer les possibilités du logiciel afin de donner lieu à de nouvelles versions. C'est par exemple ; les services offerts, l'interface utilisateur, les performances...

Une fois qu'une version modifiée du logiciel a été développée, il faut bien entendu la distribuer. De plus, il est en général nécessaire de fournir à l'exploitant du logiciel une assistance technique et un support de consultation.

En résumé, on peut dire que la maintenance et le support du logiciel comprennent les tâches suivantes :

- Effectuer des dépannages pour des corrections mineures ;
- Réappliquer le cycle de développement pour des modifications plus importantes ;
- Distribuer les mises à jour ;
- Fournir l'assistance technique et un support de consultation ;
- Maintenir un journal des demandes d'assistance et de support.

A un moment donné, on décide de mettre le logiciel hors service. Les tâches correspondantes sont accomplies durant la phase de retrait (retirement phase) et comprennent :

- ✓ Avertir les utilisateurs ;
- ✓ Effectuer une exploitation en parallèle du logiciel à retirer et de son successeur ;
- ✓ Arrêter le support du logiciel.

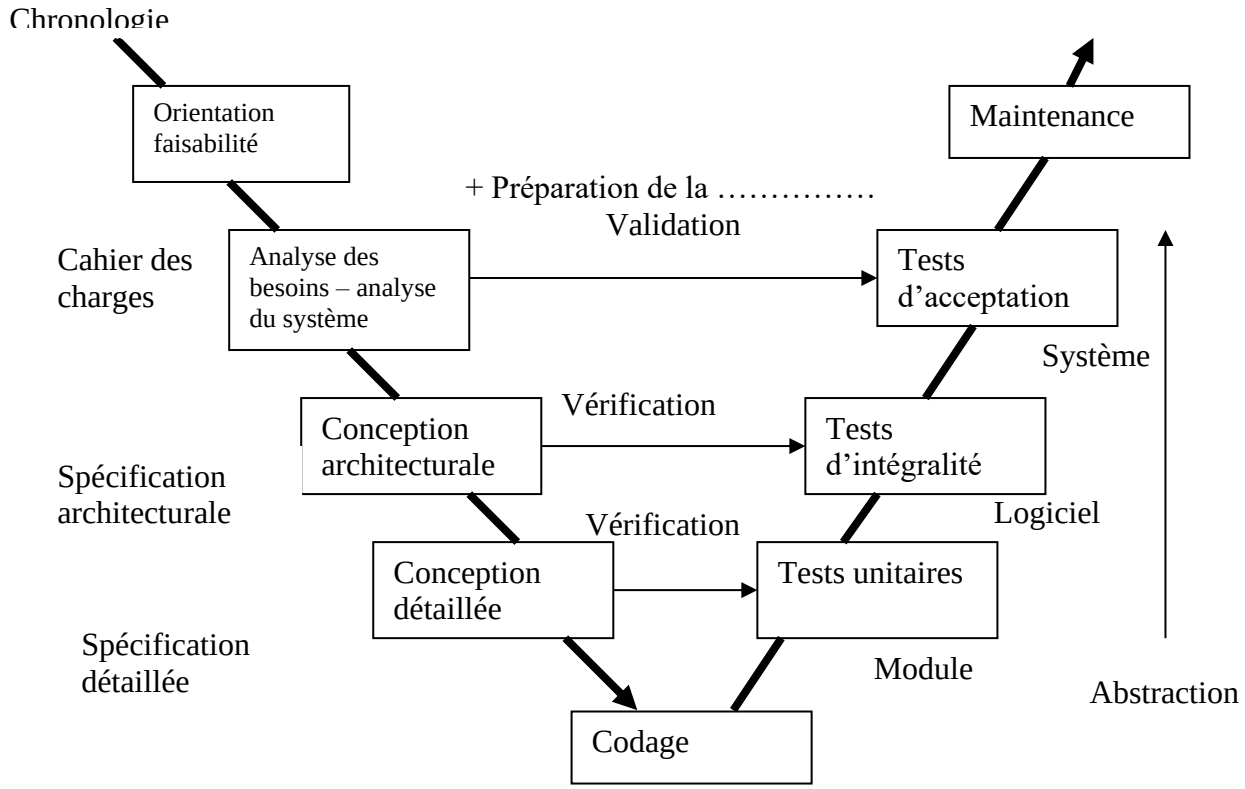
Ces étapes ne doivent pas être vues comme se succédant les unes aux autres de façon linéaire. Il y a en général (et toujours) des retours sur les phases précédentes, en particulier si les tests ne réussissent pas ou si les besoins évoluent.
Des études ont été menées pour évaluer le coût des différentes étapes du développement :

Type de système	Conception	Implantation	Test
Gestion	44	28	28
Scientifique	44	26	30
Industriel	46	20	34

Mais c'est la maintenance qui coûte le plus cher.

I.5.2. MODELE EN V

Dérivé du modèle en cascade, le modèle en V montre non seulement l'enchaînement des phases, mais aussi les relations logiques entre phases plus éloignées : ce sont des liens de validation.
Le modèle en V du cycle de vie du logiciel précise la conception des tests : (les tests systèmes sont préparés à partir de la spécification ; les tests d'intégrations sont préparés à partir de la conception architecturale ; les tests unitaires sont préparés à partir de la conception détaillée des composants). *Dérivé du modèle en cascade, le modèle en V du cycle de développement montre non seulement l'enchaînement des phases successives, mais aussi les relations logiques entre phases plus éloignées.* Ce modèle fait apparaître le fait que le début du processus de développement conditionne ses dernières étapes. Le modèle du cycle de vie en V est souvent adapté aux projets de taille et de complexité moyenne.



- Quoi faire ? Analyse
- Comment faire ? Conception générale
- Comment faire par morceau ? conception détaillée

La première branche correspond à un modèle en cascade classique. Toute description d'un composant est accompagnée de définitions de tests. Avec les jeux de tests préparés dans la première branche, les étapes de la deuxième branche peuvent être mieux préparées et planifiées. La seconde branche correspond à des tests effectifs effectués sur des composants réalisés. L'intégration est ensuite réalisée jusqu'à l'obtention du système logiciel final. *L'avantage d'un tel modèle est d'éviter d'énoncer une propriété qu'il est impossible de vérifier objectivement une fois le logiciel réalisé*. Le cycle en V est le cycle qui a été normalisé, il est largement utilisé, notamment en informatique industrielle et en télécommunication. Ce modèle fait également apparaitre les documents qui sont produits à chaque étape, et les « revues » qui permettent de valider les différents produits.

Comme les phases des modèles précédents sont successives, une difficulté majeure est rencontrée lorsque les besoins exprimés par le client ne sont pas complets au début du cycle. Il est alors possible de construire une maquette (prototype) qui simule le comportement du logiciel tel qu'il sera perçu par l'utilisateur.

NB : d'autres modèles ont étés crées pour le développement de systèmes complexes, par exemple : Modèle spiral, Modèle Incrémental, Modèle en Y, etc.

I.6 DU SYSTEME OPERATIONEL AU SYSTEME DECISIONNEL

Un système décisionnel représente un ensemble des moyens, d'outils et des méthodes permettant de collecter, consolider, modéliser et restituer les données de l'entreprise dans le but d'apporter l'aide à la prise de décision.

Cette nouvelle utilisation continue dans les bases opérationnelles des entreprises, a donné l'issue à l'élaboration des nouveaux systèmes dédiés à l'analyse et à la prise de décision. Ce système regroupe un ensemble d'informations et d'outil mise à la disposition de décideur pour supporter des manières efficaces la prise de décision.

Ainsi, un système décisionnel est un système des informations dédiées aux applications décisionnelles et a pour atout de permettre au responsable des entreprises d'avoir une vie d'ensemble de l'activité traitée.

La maturité des systèmes décisionnels peut être illustrée en cinq étapes, à savoir :

- ✓ *Les tableaux de bord (Que s'est-il passé) ?*
- ✓ *L'analyse (pourquoi) ?*
- ✓ *La prédiction (Que va-t-il s'est passé) ?*
- ✓ *L'aide opérationnelle (Qu'est-il en train de se passer) ?*
- ✓ *L'entrepôt Actif (Que faire) ?*

CHAPITRE II. PRESENTATION DU LANGAGE UML ET SES DIAGRAMMES DE BASE

II.1. LES METHODES D'ANALYSE ET DE CONCEPTION

Une méthode :

- Propose une démarche, distinguant les étapes du développement dans le cycle du vue du logiciel et exploitant au mieux les principes fondamentaux : modularité, réduction de la complexité, réutilisation, abstraction, etc.,
- Propose des formalismes (langages) et des types de documents (modèles), qui facilitent la communication, l'organisation et la vérification,

Il existe de nombreuses méthodes :

- Méthodes fonctionnelles de décomposition hiérarchique (dites de première génération) comme SADT, SA-SD, … : application du principe diviser pour régner (problème → sous problèmes) ;
- Méthodes systémiques (dites de deuxième génération) comme MERISE, SSADM, … : séparation données/traitements, niveaux conceptuels, organisationnels, physiques.
- Méthodes objets (dites de troisième génération) comme OMT, OOA, OOD, Hood, OOSE, OOM, Fusion, …

Parmi les principaux objectifs des méthodes objets, on peut noter la volonté de :

- Regrouper l'analyse des données et des traitements,
- Etablir un couplage explicite entre les concepts du monde réel et les composants exécutables (« réduire la distance sémantique entre le langage des concepteurs et celui des utilisateurs »),
- Faciliter la réutilisation,
- Simplifier les transformations entre le niveau conceptuel et l'implantation.

Au début des années 90, il existait une cinquantaine de méthodes objet, liées par un certain consensus autour d'idées communes : objets, classes, sous-systèmes, etc. mais chacune possédait sa propre notation, ses points forts et ses manques, et son orientation privilégiée vers un domaine d'application. L'idée d'une certaine unification des méthodes objet est née de ce constant. La communauté informatique, autour de l'OMG ('Objet Management Group') qui standardise les technologies de l'objet, s'est attachée à la définition d'un langage commun unique, utilisable par toutes les méthodes objets, dans toutes les phases du cycle de vie et compatible avec les techniques de réalisation du moment.

II.2. UML

Ce langage commun s'appelle UML ('Unified Modeling Language'). UML est une notation basée principalement sur les méthodes OOD (de Booch), OMT (de Rumbaugh) et OOSE (de Jacobson).

UML a été proposé afin de standardiser les produits du développement (modèles, notations, diagrammes) sans standardiser le processus de développement. Il est en effet très difficile de standardiser le processus de développement qui dépend des personnes, des applications, des cultures, etc. UML se propose de créer un langage de modélisation utilisable à la fois par les humains (forme graphique) et les machines (syntaxe précise). La figure 1 résume UML avec ses 5 vues et ses 9 diagrammes.

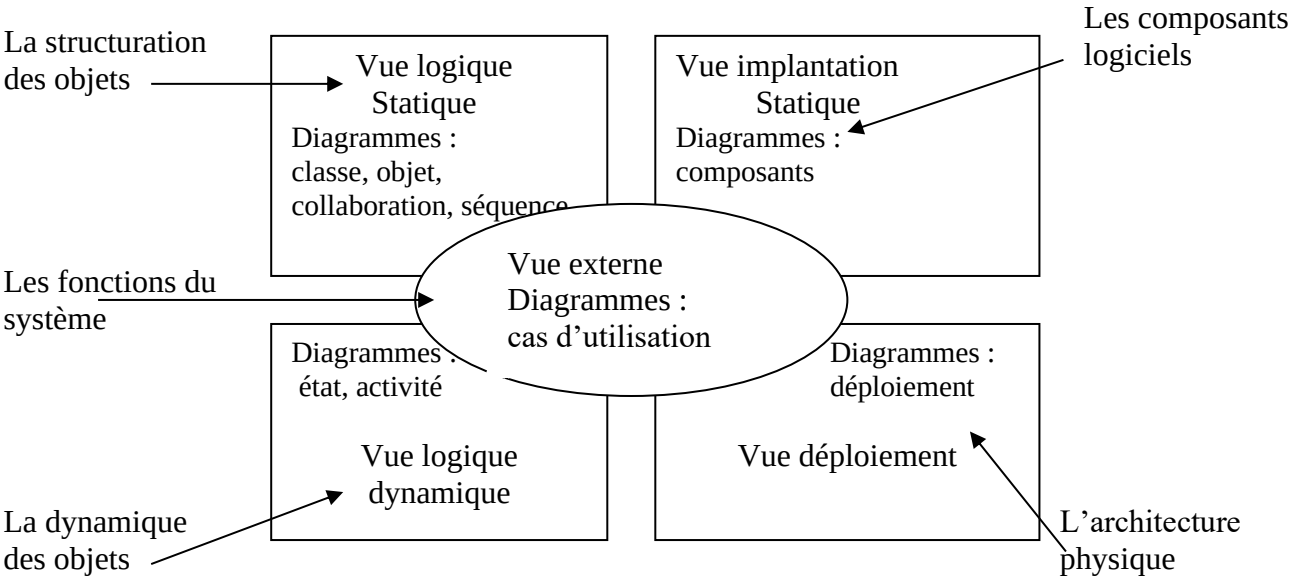


Figure 1 : les vues et les diagrammes UML

En résumé :

- UML est une notation, pas une méthode.
- UML est un langage de modélisation objet.
- UML convient pour toutes les méthodes objets.
- UML est dans le domaine public.

UML a pour but de documenter les modèles objets.

II.3. LES CAS D'UTILISATION

Formalisés par Ivar jacobson dans Objet-Oriented Software Engineering (Addison-Wesley, 1992), les cas d'utilisation ('use cases') servent à exprimer le comportement du système en termes d'actions et de réactions, selon le point de vue de chaque utilisateur (« approche centrée utilisateur »).

Les cas d'utilisation délimitent le système, ses fonctions (ses arcs), et ses relations avec son environnement. Ils constituent un moyen de déterminer les besoins du système. Ils permettent d'impliquer les utilisateurs dès les premiers stades du développement pour exprimer leur attente et leurs besoins (analyse des besoins). Ils constituent un fil conducteur pour le projet et la base pour les tests fonctionnels (cf.Figure 2).

Un acteur est une personne ou un système qui interagit avec le système étudié, en échangeant de l'information (en entrée et en sortie). On trouve les acteurs en observant les utilisateurs directs du système, les responsables de sa maintenance, ainsi que les autres systèmes qui interagissent avec lui. Un acteur représente un rôle joué par un utilisateur qui interagit avec le système. La même personne physique peut jouer le rôle de plusieurs acteurs. D'autres part, plusieurs personnes peuvent jouer le même rôle, et donc agir comme un même acteur.

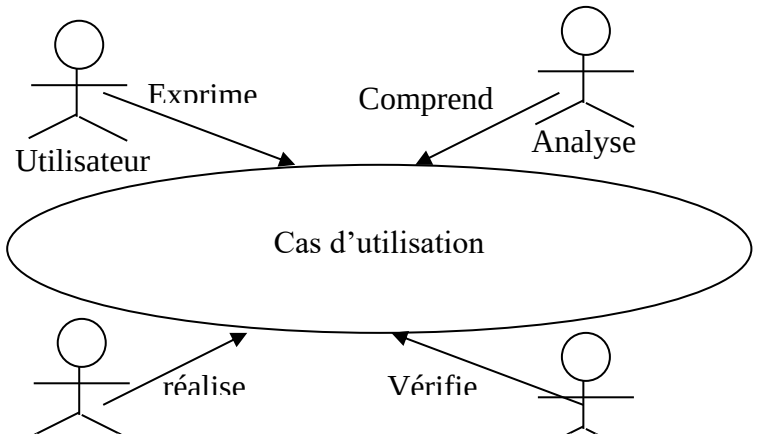
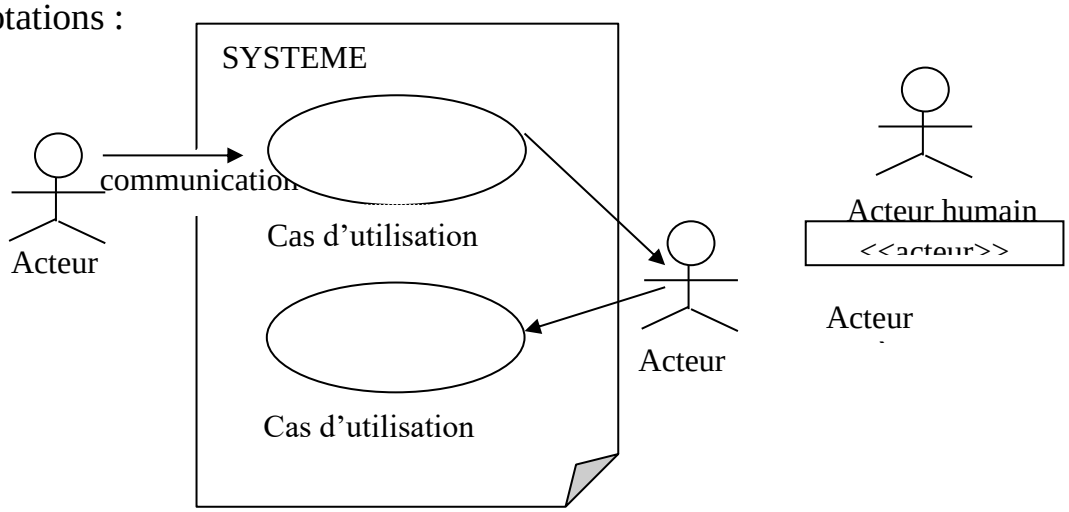


Figure 2 : les cas d'utilisation, fil conducteur du projet

Notations :



L'acteur est source ou destination d'une

Figure 3 : notation des cas d'utilisation

Les cas d'utilisations représentent le dialogue entre l'acteur et le système de manière abstraite. Les communications sont orientées (avec une flèche) ou non. Les cas d'utilisations peuvent aussi être vus comme des classes de scénarios. Enfin, les cas peuvent être structurés par les relations <<inclut>> (un cas inclut obligatoirement un autre cas) et <<Etend>> (un cas est une variante d'un autre) (cf. Figure 4). On peut aussi avoir de la généralisation/spécialisation entre acteurs et entre cas (cf. Paragraphe 6).

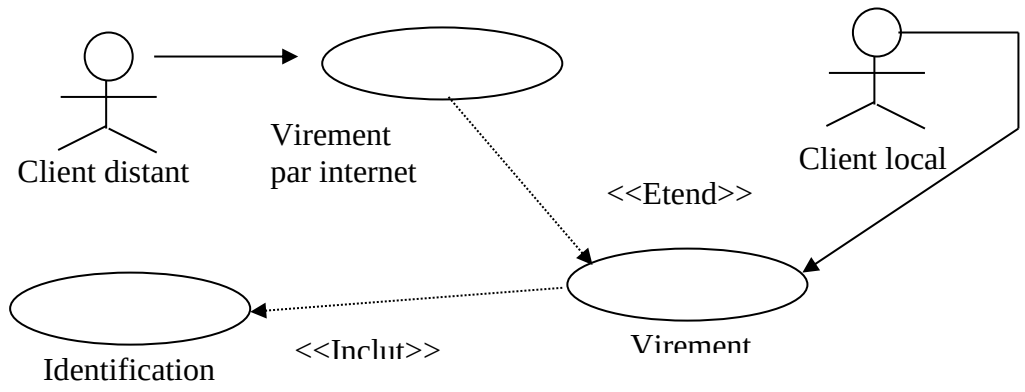


Figure 4 : structuration des cas d'utilisation

Démarche d'utilisation des cas d'utilisation :

1) Identifier les acteurs et les cas.
2) Décrire les cas en écrivant des scénarios sous forme textuelle.
3) Dessiner les diagrammes de cas.
4) Récapituler les cas, les structures et s'assurer de la cohérence d'ensemble.

II.4. LES OBJETS.

Les objets informatiques définissent une représentation simplifiée des entités du monde réel, qu'il s'agisse d'entités concrètes (avec une « réalité physique ») ou conceptuelles. Les objets sont des abstractions qui mettent en avant les caractéristiques essentielles et dissimulent les détails. Une abstraction se définit par rapport à un point de vue (tout dépend de l'intérêt que l'on porte à l'objet).
Exemples d'objets : une transaction bancaire, un client, un pop-up menu.

Les trois caractéristiques fondamentales des objets sont l'état, le comportement, l'identité. L'état correspond aux valeurs instantanées de tous les attributs (ou données membres) de l'objet. L'état évolue au cours du temps. L'état d'un objet à un instant donné est la conséquence de ses comportements passés.

Exemples : pour un signal électrique : valeurs de l'amplitude, de la pulsation, de la phase, … ; pour une voiture ; valeurs de la marque, de la puissance, de la couleur, du nombre de places assises, de la quantité d'essence, …

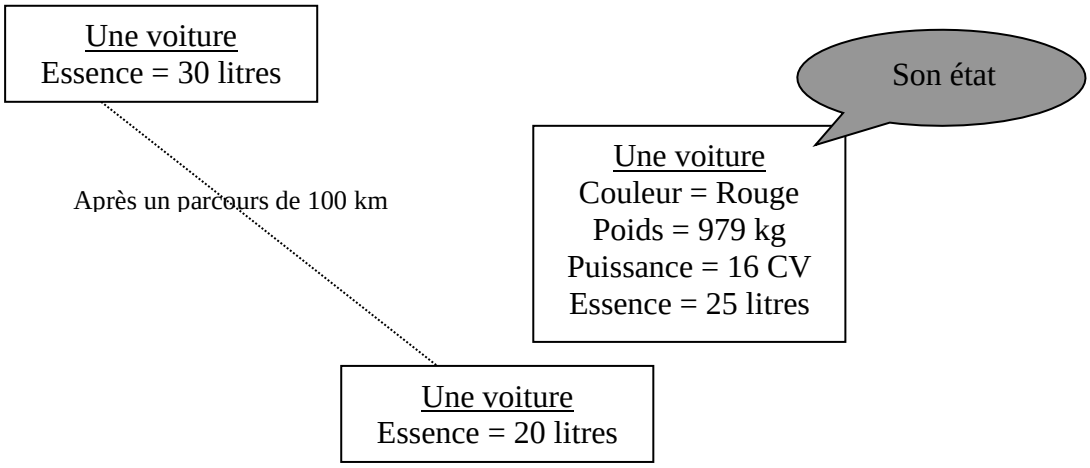


Figure 5 : objet et état

Le comportement décrit les actions et les réactions d'un objet (ou 'compétences' d'un objet). Le comportement se matérialise sous la forme d'un ensemble d'opérations (ou méthodes ou fonctions membres). On distingue souvent les opérations selon leur finalité de constructeur, accesseur, modificateur, destructeur et itérateur.

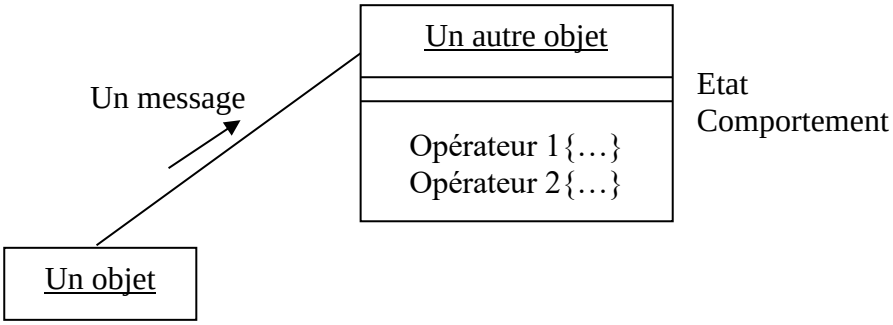


Figure 6 : comportement d'un objet

Un objet peut faire appel aux compétences d'un autre objet par invocation d'une opération (ou 'envoi de message). Les messages servent à implanter les flots de données, les flots de contrôle, les événements, etc.

L'état et le comportement sont liés :
-le comportement dépend de l'état (cf. la pré condition de l'opération) ; le message peut être accepté ou refusé ;
-l'état est modifié par le comportement (cf. la post condition de l'opération).

Tout objet possède une identité interne (non modifiable) qui lui est propre et qui le caractérise. L'identité permet de distinguer tout objet de façon non ambiguë, indépendamment de son état (de ses attributs). Les langages objets utilisent des identifiants internes. Les identifiants et clés primaires des bases de données sont inutiles en objet.

Intérêt des objets :

- Concurrence : les objets sont autonomes et évoluent indépendamment les uns des autres.
- Maîtrise de la complexité :

-l'encapsulation permet de se concentrer sur un objet et un seul,
-il est possible de tester de façon quasi exhaustif un objet.

- Evolution facilitée : rajouter de nouveaux objets est simple.
- Réutilisation possible.

Difficulté des objets :

- Tout n'est pas naturellement objet.
- Tester un système OO est difficile.
- Penser objet oblige à changer de mentalité.

II.5. LES COLLABORATIONS

Une application est une société d'objets qui collaborent afin de réaliser les fonctions de l'application. Le comportement global d'une application repose donc sur la communication entre les objets qui la composent (échanges de messages).

5.1. Les diagrammes de collaboration

Les diagrammes de collaboration mettent l'accent sur les relations « spatiales » entre objet. Les messages peuvent être numérotés pour introduire une dimension temporelle.

De nombreuses notations annexes permettent de caractériser les messages. Ils sont souvent utilisés pour décrire grossièrement la réalisation des cas d'utilisation.

Exemple : un objet A envoie un message X à un objet B, puis l'objet B envoie un message Y à un objet C, et enfin C s'envoie à lui-même message Z.

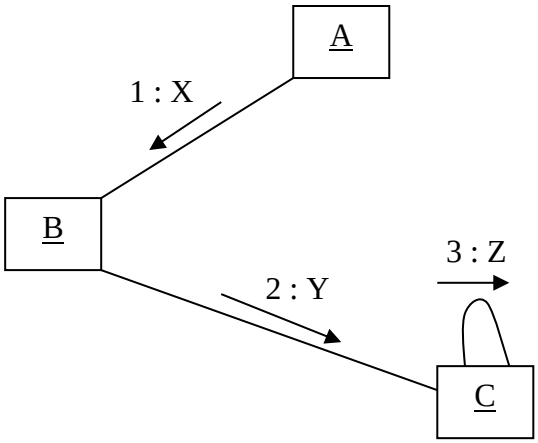


Figure 7 : un diagramme de collaboration

5.2. Les diagrammes de séquence

Ils mettent l'accent sur les relations temporelles. De nombreuses notations annexes permettent de préciser la nature des messages (appel de procédure, simple signal, message répétitif, conditionnel, réflexif, récursif, etc.) et les données véhiculées. Ils peuvent être utilisés aussi bien pour préciser la réalisation des cas d'utilisation que pour spécifier de manière très détaillée la dynamique d'un ensemble d'objets (appels de méthodes).

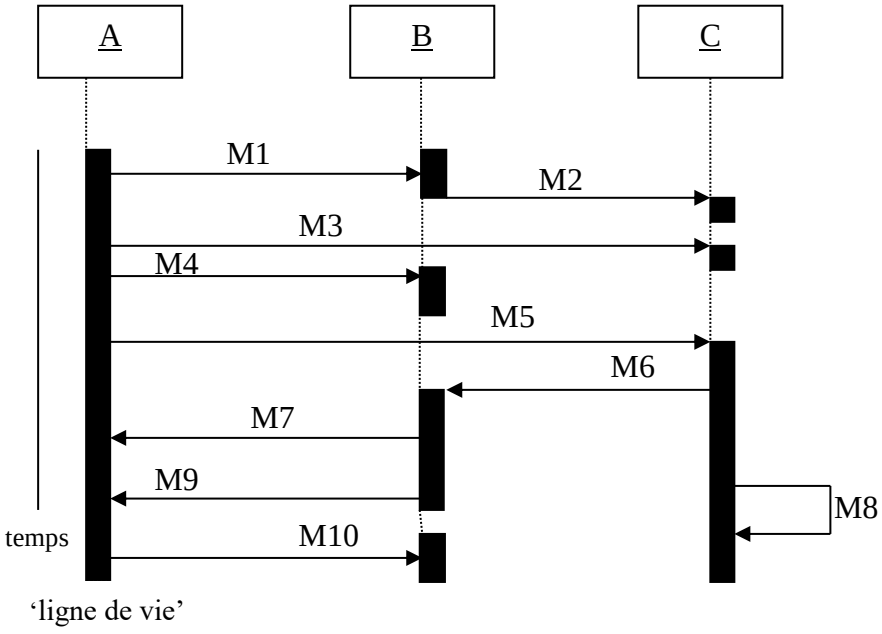


Figure 8 : un diagramme de séquence

II.6. LES CLASSES

Le monde qui nous entoure est constitué de très nombreux objets. Pour comprendre le monde, l'être humain a tendance à regrouper les éléments qui se ressemblent. Regrouper des objets suivant des critères de ressemblance s'appelle classer. Les humains ont classé les animaux, les plantes, les champignons, les atomes, …

La classe est la description abstraite d'un ensemble d'objets. Elle peut être vue comme la factorisation des éléments communs à un ensemble d'objets.

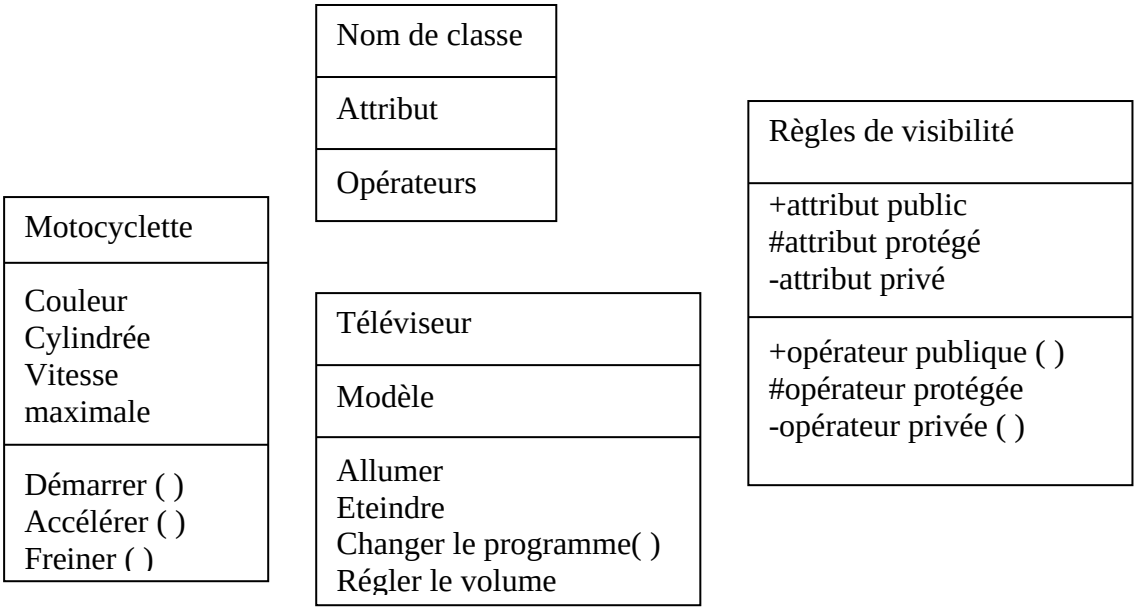


Figure 9 : des exemples de classe

La classe intègre les concepts de type (en tenant que « moule à instances ») et de module (avec l'idée d'interface + corps).

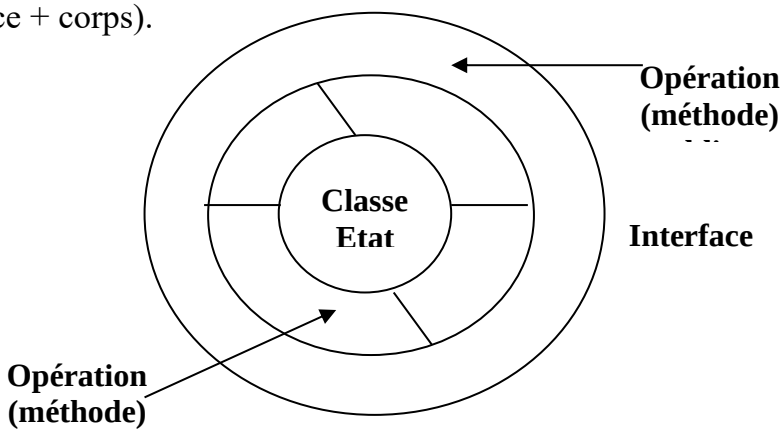


Figure 10 : interface et corps d'une classe

La hiérarchisation des classes permet de gérer la complexité. Dans un sens, la généralisation correspond à la factorisation des éléments communs de classes (attributs, opérations) ce qui favorise la réduction de la complexité et la généricité. Dans l'autre sens, la spécification permet d'adapter une classe générale à un cas particulier ce qui favorise la réutilisation et la modification incrémentielle.

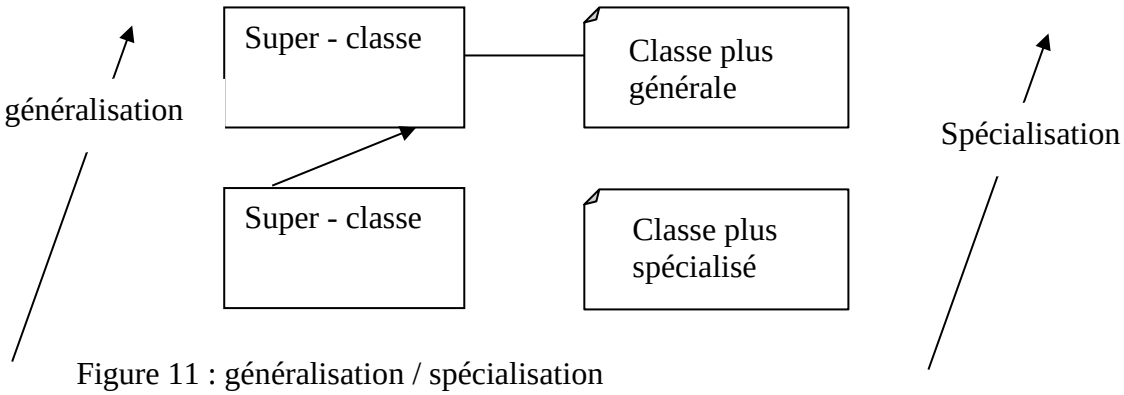


Figure 11 : généralisation / spécialisation

Exemple :

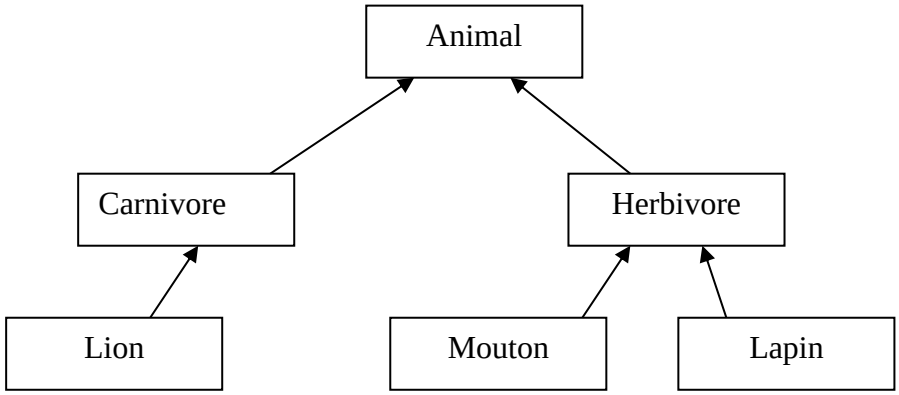


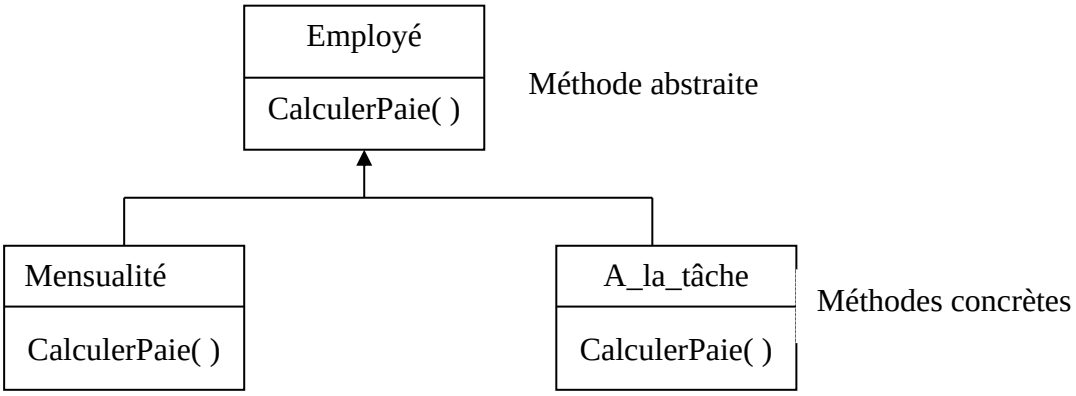
Figure 12 : un exemple de hiérarchisation

Les instances de la classe spécialisée héritent de la description des attributs (variables) et des opérations (méthodes) de la super-classe. Elles peuvent en ajouter d'autres et /ou en redéfinir certaines.
On peut dire aussi que le type Lion est un sous type du type Carnivore. Tout lion est un carnivore (on parle aussi de relation 'est – un' ou 'isa' en anglais). L'ensemble des lions est un sous ensemble des carnivores. Mais l'ensemble des attributs d'un lion est un sur ensemble de l'ensemble des attributs d'un carnivore (d'où le mot extends qu'utilise java pour lier sous –classe et super –classe).

La relation de spécification / généralisation est une relation non réflexive, non symétrique et transitive. L'héritage multiple (plusieurs super classes) est possible.

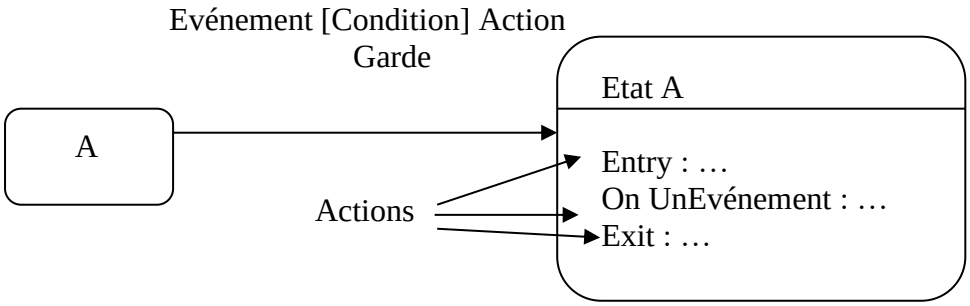
Un même message peut être interprété de manière différente selon la nature de l'objet receveur. On parle de polymorphisme. L'émetteur n'a pas besoin de connaître la classe du receveur.

Exemple : on paye des employés de 2 types ('mensualisés' et 'à la tâche'). Il suffit d'envoyer la méthode calculerPaie ( ) à tous les employés. Si un nouveau type d'employé apparaît le programme de paie n'a pas à être modifié.

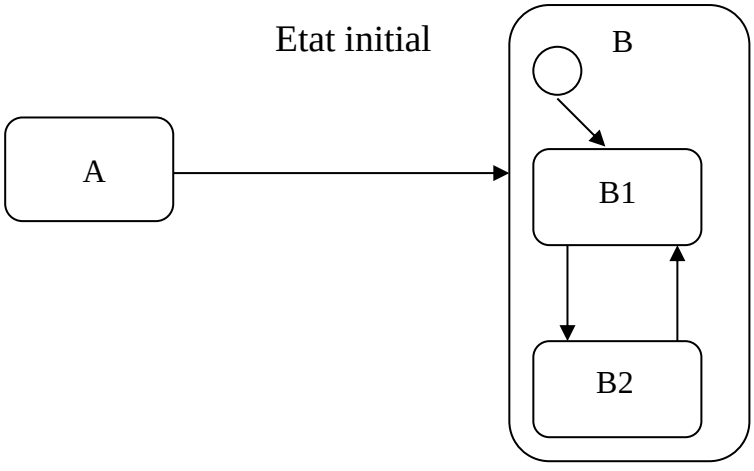


II.7. LES DIAGRAMMES D'ETAT

Les diagrammes d'état servent à décrire le comportement des objets (classes) complexes. Ce sont des diagrammes d'état étendus avec des conditions (gardes) sur les transitions et des actions sur les transitions et les états.



Ces diagrammes permettent aussi la décomposition d'états en sous états. On parle aussi d'état composite.



Enfin, il est possible de décrire du parallélisme entre sous-systèmes, ce qui rapproche les diagrammes d'états des réseaux de Pétri.

II.8. LES DIAGRAMMES D'ACTIVITE

Ils permettent de décrire un flot de contrôle entre opérations (calculs). Il s'agit d'un gros ordinogramme incluant éventuellement du parallélisme.
A un niveau macroscopique, les diagrammes d'activité permettent de décrire des enchaînements de fonctionnalités. Ils complètent donc bien les cas d'utilisation au niveau de l'analyse des besoins.
A niveau microscopique, les diagrammes d'activité permettent, par exemple, de décrire l'algorithme d'une action d'un diagramme d'états.

II.9. AUTRES ELEMENTS

9.1. Les diagrammes de composants

Les composants sont des codes (sources, exécutables), des scripts, des fichiers, des tables, etc.

9.2. Les diagrammes de déploiement

Ces diagrammes illustrent (cf. Figure 25) :

- La disposition physique des différents matériels (ou nœuds) qui entrent dans la composition du système,
- La répartition des composants (cf. diagrammes de composants) au sein des nœuds,
- Les supports de communication entre noeuds.

II.10. ELEMENTS D'UNE DEMARCHE

UML n'impose pas de processus. La 'démarche naturelle', impliquée par la notation UML, part des cas d'utilisation qui expriment un point de vue fonctionnel sur le système.

Puis les diagrammes de collaboration et de séquence associées aux cas font apparaître les classes d'objets impliquées dans le système et donc passer à une vue 'objet' (cf. Figure 26).
Mais on peut aussi passer à une vue 'hiérarchie de fonctions' à partir des cas d'utilisation, comme le montre la Figure 27.

Diagrammes essentiels par phase :

- Analyse des besoins : cas d'utilisation.
- Analyse du système : diagrammes de classes, de collaboration, d'activités, (enchaînement des cas).
- Conception : diagrammes de classes, de séquences, d'activités (conception méthodes), d'états, de composants, de déploiement.

CHAPITRE III. NIVEAU CONCEPTUEL : FACE A FACE MERISE ET UML

III.1. INTRODUCTION

Avant la phase de conception, il est indispensable d'analyser le système existant en faisant l'inventaire le plus exhaustif possible des échanges d'information entre intervenants du domaine d'étude et des traitements réalisés. Cette analyse se clôture par la décision d'informatiser ou non, et le cas du oui, un cahier des charges et éventuellement un plan directeur de réalisation sont établis.

III.1.1. DEMARCHE DE LA METHODE MERISE

Parmi les informations qui appartiennent au système d'information, certaines doivent ou peuvent faire l'objet d'un traitement automatisé grâce aux outils informatiques. Pour assurer la cohérence du système d'information, la méthode Merise propose une démarche d'informatisation comportant comme étapes :

- le **schéma directeur** : dont le rôle est de définir, de manière globale, la politique d'organisation et d'automatisation du système d'information. Pour ce faire, il est nécessaire de répertorier l'ensemble des applications informatiques existantes à modifier et à développer. Pour rendre contrôlable et modulable ce développement, il est nécessaire de découper le système d'information en sous-ensembles homogènes et relativement indépendants. Ces sous-ensembles sont appelés domaines. Par exemple, on peut trouver le domaine « Approvisionnement », le domaine « Personnel ». Les résultats attendus à la fin de cette étape sont une définition précise des domaines, une planification du développement de chaque domaine et un plan détaillé, année par année, des applications qui doivent être réalisées. Cette étape concerne les décideurs.

- **l'étude préalable par domaine** : qui doit aboutir à une présentation générale du futur système de gestion (modèles des données et des traitements) en indiquant les principales novations par rapport au système actuel, les moyens matériels à mettre en œuvre, les bilans coût – avantage. Cette étude est réalisée en 4 phases :

- Une **phase de recueil** qui a pour objectif d'analyser l'existant afin de cerner les dysfonctionnements et les obsolescences les plus frappantes du système actuel.
- Une **phase de conception** qui a pour objectif de formaliser et hiérarchiser les orientations nouvelles en fonction des critiques formulées sur le système actuel et d'autre part des politiques et des objectifs de la direction générale. Cela revient à modéliser le futur système avec une vue pertinente de l'ensemble.
- Une **phase d'organisation** dont l'objectif est de définir le système futur au niveau organisationnel : qui fait quoi ?
- Une **phase d'appréciation** dont le rôle est d'établir les coûts et les délais des solutions définies ainsi que d'organiser la mise en œuvre de la réalisation.

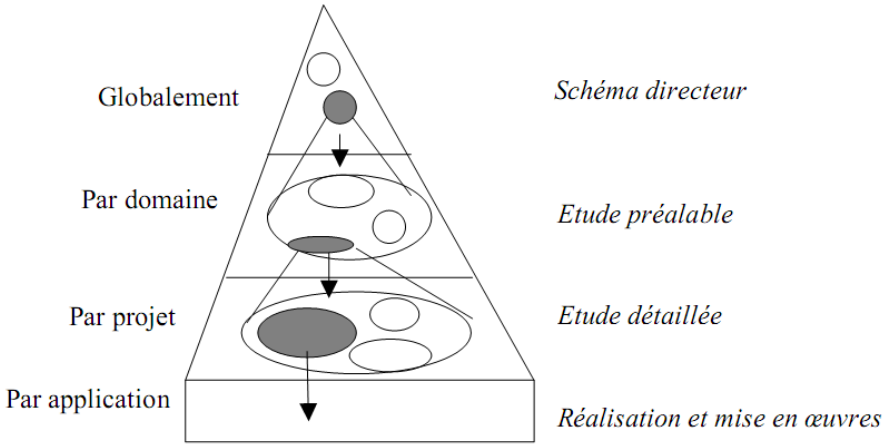
A cet effet un découpage en projets est effectué.

- **l'étude détaillée** par projet qui consiste d'une part à affiner les solutions conçues lors de l'étude préalable et d'autre part à rédiger, pour chaque procédure à mettre en œuvre, un dossier de spécifications détaillé décrivant les supports (maquettes d'états ou d'écran) ainsi que les algorithmes associés aux règles de gestion… A l'issue de cette étude, il est possible de définir le cahier des charges utilisateurs qui constitue la base de l'engagement que prend le concepteur vis à vis des utilisateurs. Le fonctionnement détaillé du futur système, du point de vue de l'utilisateur, y est entièrement spécifié.

- la **réalisation** dont l'objectif est l'obtention des programmes fonctionnant sur un jeu d'essais approuvés par les utilisateurs.

- la **mise en œuvre** qui se traduit par un changement de responsabilité : l'équipe de réalisation va en effet transférer la responsabilité du produit à l'utilisateur. Cette étape intègre en particulier la formation des utilisateurs. Après une période d'exploitation de quelques mois, la recette définitive de l'application est prononcée.

- la **maintenance** qui consiste à faire évoluer les applications en fonction des besoins des utilisateurs, de l'environnement et des progrès technologiques. Le schéma suivant, extrait de l'ouvrage « La méthode Merise » reprend les étapes décrites ci-dessus.



III.1.2. TROIS CYCLES DE LA METHODE MERISE

La démarche de MERISE se fait selon trois cycles suivants : le cycle d'abstraction, avec les formalismes à 4 niveaux (conceptuel, organisationnel, logique, opérationnel ou physique) pour modéliser un système d'information ; le cycle de vie qui comporte 3 grandes périodes dont la conception, la réalisation et la maintenance ; et le cycle de décision qui va de l'étude à la maintenance.

Le cycle d'abstraction de MERISE distingue quatre niveaux de description des systèmes d'information :

- **Niveau conceptuel** *: « Décrit le QUOI faire et avec QUELLES données ? »*

Son rôle est d'exprimer les choix fondamentaux de gestion et les objectifs (tenant naturellement compte des contraintes) ou les finalités de l'entreprise. Il décrit les invariants de l'organisation et le métier de l'organisation indépendamment des aspects organisationnels et des aspects techniques de mise en œuvre (on ne se préoccupe pas de l'organisation du travail ni du matériel utilisé). Il se traduit en termes de modèle conceptuel des données (MCD) avec entité, association, propriété et de modèle conceptuel des traitements (MCT). Il produit la représentation abstraite des données et des traitements.

Du point de vue des traitements, il définit : objectif, résultat, règles de gestion, enchaînement ; et du point de vue des données : signification, structure, liens. Bref, c'est la description la plus stable du système.

- Niveau organisationnel : *« Décrit QUI fait, QUAND et OÙ ? »*

Son rôle est de définir l'organisation de travail qu'il est souhaitable de mettre en place dans l'entreprise pour atteindre les objectifs souhaités.

Il définit la répartition géographique et fonctionnelle des sites de travail (du point de vue des données et des traitements), le mode de fonctionnement (temps réel ou temps différé), la répartition du travail homme/machine (degré et type d'automatisation), les postes de travail et leur affectation, la volumétrie des données, la sécurité des données, indépendamment des moyens de traitement et de stockage de données actuels ou futurs. Université

Les opérations conceptuelles vont être décomposées au niveau organisationnel en une ou plusieurs opérations organisationnelles. C'est la description des postes de travail de l'entreprise et des informations qu'elle traite.

Il se traduit en termes de modèle organisationnel des données (MOD) et de modèle organisationnel des traitements (MOT).

- Niveau logique : *« Décrit AVEC QUOI ? QUELS OUTILS* ? »

Son rôle est de définir l'organisation qu'il est souhaitable de mettre en place dans l'entreprise pour atteindre les objectifs souhaités. C'est à cette étape que l'on précise l'emploi des bases de données ou des fichiers…

Il exprime la forme que doit prendre l'outil informatique pour être adapté à l'utilisateur, à son poste de travail, cela indépendamment de l'informatique spécifique et des langages de programmation ou de gestion des données. Il décrit le schéma de la base de données (relationnel, hiérarchique ou réseau), c'est-à-dire les caractéristiques du mode de gestion des données, la répartition des données sur les différentes unités de stockage, les volumes par unité de stockage et l'optimisation des coûts induits par le mode de gestion.

Il se traduit en termes de modèle logique des données (MLD) et de modèle logique des traitements (MLT).

- Niveau physique / opérationnel : *« Décrit le COMMENT faire ? »*

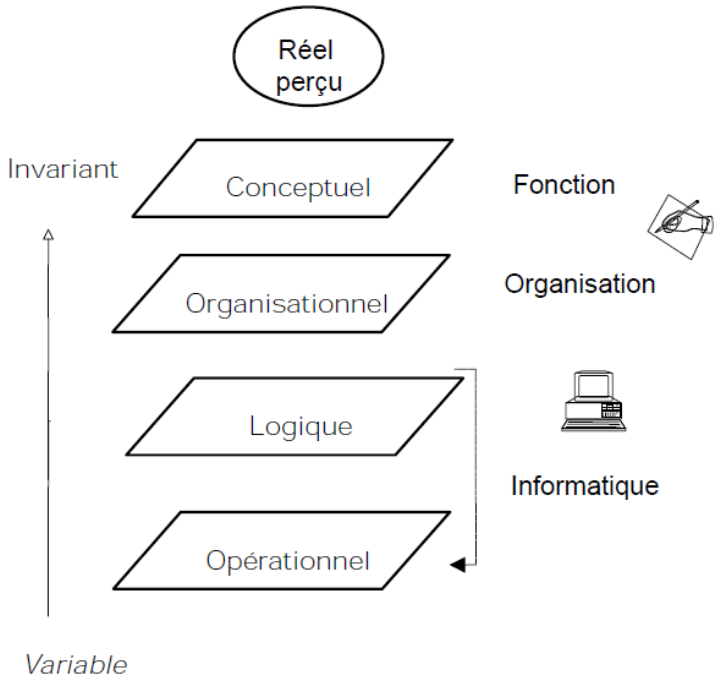
Son rôle est de définir les solutions techniques (mode de stockage pour les données, matériels, découpage des programmes pour les traitements) et la prise en compte de leurs spécificités.

Il répond aux besoins des utilisateurs sur les aspects logiciels et matériels. Il définit complètement les fichiers, les programmes, l'implantation physique des données et des traitements, les ressources à utiliser et les modalités de fonctionnement.

C'est la description des moyens mis en œuvre pour gérer les données et effectuer les traitements. Ils se traduisent en termes de Modèle physique des données et de Modèle physique ou opérationnel des traitements.

Remarque : Chacun de ces niveaux n'est pas indépendant. Il va de soi, que les choix techniques (niveau physique) peuvent être imposés dès le début du projet. Les modèles conceptuels et organisationnels seront alors étudiés en tenant compte de ces contraintes. Chacun des modèles seront affinés au cours de la vie du projet.

Le schéma suivant résume les modèles que nous allons aborder tout au long des lignes qui suivent :



Les niveaux conceptuel et organisationnel représentent toute l'organisation, et les niveaux logique et physique la solution informatique.

III.2. PRESENTATION DES MODELES

III.2.1. LES MODELES AU NIVEAU CONCEPTUEL

III.2.1.1. LE MODELE CONCEPTUEL DES DONNEES (MCD)

Un modèle conceptuel des données est une représentation des besoins en matière de données pour un système d'information. Il met en évidence les entités, leurs attributs, les associations et contraintes entre ces entités pour un domaine donné. Cette représentation, de nature sémantique, ne comporte aucune indication concernant la structure de mémorisation des données associées aux entités. Le modèle conceptuel est généralement représenté à l'aide du Formalisme entité-association (EA, appelé aussi entité-relation ou ER).

Plusieurs formalismes ont été proposés pour réaliser un modèle conceptuel de données, par exemple le réseau sémantique de J. F. Sowa [SOW 84], mais seul le formalisme entité-association fait l'objet d'un réel consensus en cette matière et est une composante essentielle de la plupart des outils de conception de base de données.

a. Modèle entité-association

Dans le monde réel des organisations, le modèle entité-association permet de repérer et de décrire des entités ou des associations d'entités à l'aide d'un nombre limité des mots appelés : valeurs.

L'idée fondamentale du modèle EA est de retenir comme concepts de base pour la représentation, les mêmes concepts génériques que ceux qui guident le processus d'abstraction conduisant de l'observation d'une réalité à sa description. On suppose que la perception d'une situation observée se fait naturellement sur la base d'une identification des objets présents (qu'ils soient réels, une personne, ou abstraits, une foule), de liens entre ces objets (une personne conduit une voiture) et de propriétés observables (la taille d'une personne, la couleur d'une voiture, ...).

Le modèle EA propose une description sur la base de ces mêmes trois concepts renommés de façon à distinguer facilement le discours sur la réalité (fait en termes d'objets, liens et propriétés) du discours sur la représentation de la réalité (fait en utilisant les termes du modèle). La correspondance entre les trois concepts génériques et la terminologie du modèle EA est la suivante :

Objet $\rightarrow$ entité

Lien $\rightarrow$ association

Propriété $\rightarrow$ attribut

Entité : est une représentation d'un objet du monde réel (concret ou abstrait), perçu par le concepteur comme ayant une existence propre, et à propos duquel on veut enregistrer des informations.

Une entité existe indépendamment du fait qu'elle puisse être liée à d'autres entités de la base de données.

Exemple : KAFUNDA, NZUZI, Un 4x4, Une BMW, Bas-congo, Equateur

Type d'entités (TE) : est une représentation d'un ensemble d'entités perçues comme similaires et ayant les mêmes caractéristiques.

Exemple : Personne, Voiture, Province

Association : est une représentation d'un lien entre plusieurs entités, lien où chaque entité liée joue un rôle déterminé.

Exemple : KAFUNDA a un 4x4, NZUZI a une BMW, …

Si l'association lie deux (ou plus) entités du même type, elle est dite "cyclique" ou "réflexive" et, dans ce cas, la spécification du rôle de chaque entité est indispensable pour supprimer les ambiguïtés possibles.

Type d'associations (TA) : c'est la représentation d'un ensemble d'associations ayant la même sémantique et décrites par les mêmes caractéristiques.

Exemple : Avoir entre KAFUNDA et 4x4, et entre NZUZI et BMW.

Attribut : c'est la représentation d'une propriété associée à un type d'entité ou à un type d'association, ou participant à la composition d'un autre attribut. L'ensemble des attributs d'un type d'entités (type d'associations) représente l'ensemble des informations inhérentes que l'on souhaite conserver sur les entités (associations) du type d'entités (type d'associations).

Exemple : L'âge d'une personne, la puissance d'une voiture, la superficie d'une province...

Identifiant : Un identifiant d'un type d'entités ou d'un type d'associations est un attribut ou un ensemble d'attributs tels qu'il n'existe pas deux occurrences du type d'entités ou d'associations qui ont la même valeur pour cet attribut ou cet ensemble d'attributs.

- Un attribut monovalué est un attribut qui ne peut prendre qu'une seule valeur par occurrence du type d'entités ou d'associations ;
- Un attribut multivalué est un attribut qui peut prendre plusieurs valeurs par occurrence du type d'entités ou d'associations.
- Un attribut obligatoire est un attribut qui doit prendre une valeur au moins par occurrence du type d'entités ou d'associations ;

- Et un attribut facultatif est un attribut qui peut ne pas prendre une valeur par occurrence.

Exemple : Le numéro d'immatriculation peut servir d'identifiant d'une voiture...

Remarques :

- On souligne les identifiants d'une entité.
- L'identifiant d'une association est un sous-ensemble des identifiants des entités liées.
- Un type d'entités possède au moins un identifiant (son identifiant), tandis qu'un type d'associations peut ne pas avoir d'attribut.

Cardinalités : ce sont deux nombres formant un couple qui expriment le minimum et maximum qu'une occurrence d'un type d'entités associée peut intervenir dans le type d'associations. (C'est une information supplémentaire exprimant la règle de participation des entités dans les associations au niveau des occurrences).

Exemple : un client peut commander entre 1 et n produits

Cardinalité minimale possible :

- 0 si une occurrence de l'entité peut exister tout en n'intervenant dans aucune occurrence de l'association
- 1 si une occurrence de l'entité ne peut exister que si elle intervient dans au moins une occurrence de l'association
- N : cas rare à éviter

Cardinalité maximale possible :

- 1 si une occurrence de l'entité ne peut pas être impliquée dans plus d'une occurrence de l'association
- N si une occurrence de l'entité ne peut être impliquée dans plus d'une occurrence de l'association

b. Description d'un schéma EA

Un type d'entités est décrit par les spécifications suivantes :

- Le nom du type d'entités,
- Une définition libre précisant la population exacte du type d'entités,
- La description des attributs du type d'entités,
- La composition des identifiants du type d'entités s'ils existent.

Remarque : Deux types d'entités différents ne peuvent avoir le même nom. La définition libre est une partie très importante de la description d'un type d'entités.

Un type d'associations est décrit par les spécifications suivantes :

- Le nom du type d'associations,
- Une définition libre précisant la population exacte du type d'associations,
- Les noms des types d'entités participant au type d'associations, avec les noms du rôle les associant au type d'associations. En pratique le nom de rôle n'est obligatoire que pour les associations cycliques.

Un attribut est décrit par les spécifications suivantes :

- Nom d'attribut,
- Définition libre.

Si l'attribut n'est pas composé d'autres attributs, il faut spécifier le domaine de valeurs associées c'est-à-dire spécifier quel est l'ensemble de valeurs autorisées pour l'attribut.

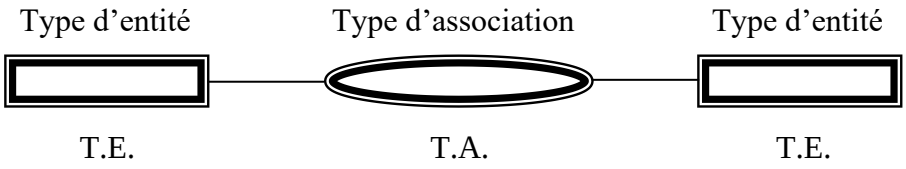
Exemple : le domaine associé à l'âge d'une personne est un nombre entier naturel dont la valeur peut être 9 ans pour KAFUNDA.

Si l'attribut est composé d'autres attributs, il faut décrire des attributs qui le composent.

c. Diagramme EA

Une base de données décrite par le modèle entité-association est l'ensemble de populations de types d'entités et de types d'associations apparaissant dans la réalité.

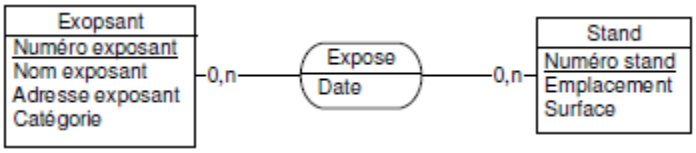
Le modèle entité-association permet une représentation graphique assez lisible de la base de données. Il s'appelle diagramme d'entité-association.



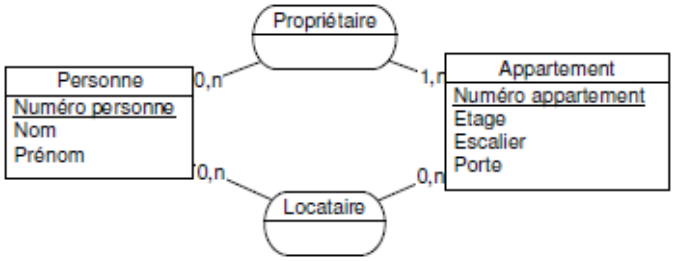
Remarque : *On accepte généralement cet abus de langage consistant à parler d'« Entité » à la place de« Type d'entités », d'« Occurrence d'entité » à la place d'« Entité », d'« Association » à la place de « Type d'associations » et d'« Occurrence d'associations » à la place d'« Association »*

Exemples :

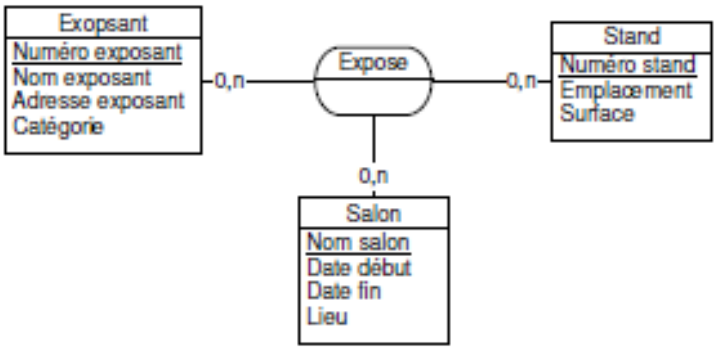
- Une association peut avoir des propriétés.



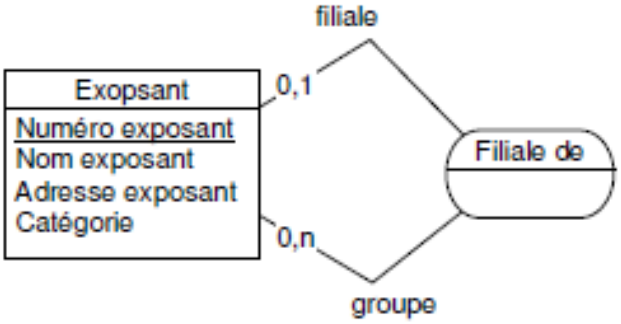
- Il peut y avoir plusieurs associations liant les mêmes entités si la sémantique est différente. (Association plurielle)



- Une association peut être multiple (ternaire ou n-aire)



- Une association peut être réflexive ou cyclique (plusieurs occurrences de la même entité seront associées)



Remarques :

- Il est parfois difficile de faire un choix entre entité et association, souvent le contexte aide à décider.

  Ex : Un mariage est-il une association entre deux personnes ou une entité pour lequel on veut conserver un numéro, une date, un lieu, etc. et que l'on souhaite manipuler en tant que tel ?

- Lorsqu'on ne parvient pas à trouver d'identifiant pour une entité, il faut se demander s'il ne s'agit pas en fait d'une association. Si ce n'est pas le cas, un identifiant arbitraire numérique entier peut faire l'affaire.
- Lorsqu'une association porte les cardinalités 1,1 de part et d'autre, il faut se demander si cette association et les entités liées ne décrivent pas en fait une seule entité ?

En fonction des cardinalités une association est définie de type :

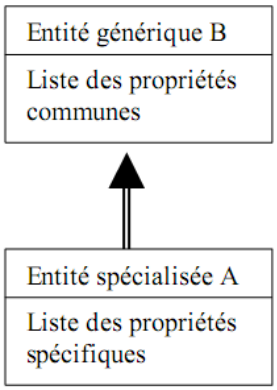
- 1 :1 si toutes les cardinalités maximales valent 1
- 1 : n s'il existe au moins une cardinalité maximale à n et une autre à 1
- n :m si toutes les cardinalités maximales valent n

d. Quelques concepts étendus (MERISE 3)

Le modèle entité-association retenu par la méthode Merise date des années 70. Or les concepts de ce modèle peuvent s'avérer insuffisants pour modéliser certaines situations ou contraintes et l'on est obligé dans ce cas d'ajouter des commentaires pour en faire mention. Les extensions au modèle individuel remédient aux faiblesses du formalisme de base.

d.1) Le concept d'héritage

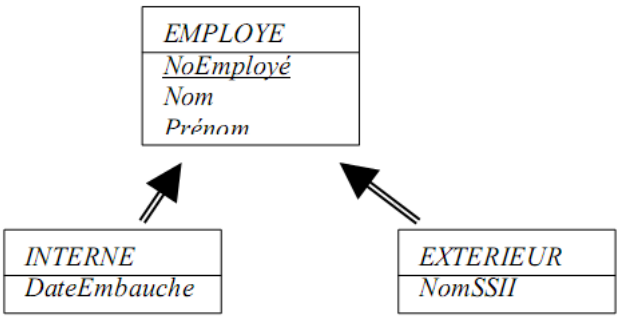
Quand le concepteur s'aperçoit que plusieurs entités, proches mais distinctes, partagent un ensemble de caractéristiques, il doit mettre en œuvre un processus de création d'entités génériques (ou entités sur-types) et d'entités spécialisées (ou entités sous-types) appelé « héritage ». Ce concept qui permet de représenter le lien « est-un » ou « IS-A » entre deux entités A et B (une occurrence de A est une occurrence de B) est représenté graphiquement par une flèche double allant de A vers B.



On dit qu'il y a héritage simple quand un sous-type n'a qu'un seul sur-type. Dans ce cas, toutes les occurrences du sous-type sont en même temps des occurrences de son sur-type. Cela n'implique pas que toutes les occurrences du sur-type soient des occurrences de l'un des sous-types.

Le sous-type hérite de toutes les propriétés de son sur-type y compris de son identifiant.

Exemple :



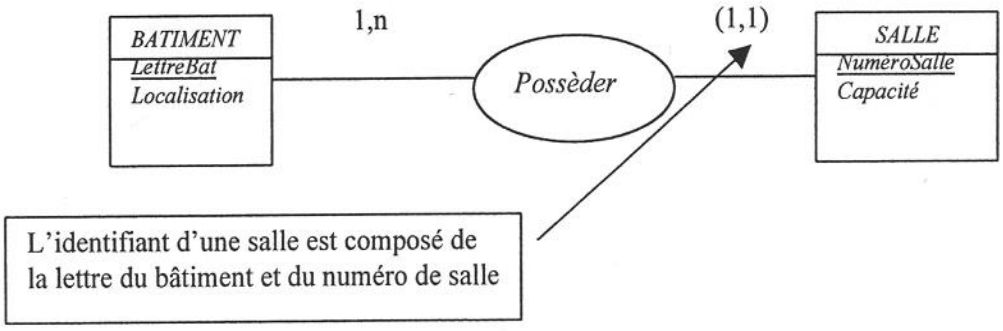
Le sous typage est une orientation vers le monde "objet".

d.2) L'identification des occurrences d'entités

Les extensions à l'identification des entités permettent d'accepter plusieurs façons d'identifier une entité et suppriment ainsi les identifiants artificiels, introduits uniquement pour respecter la définition d'une entité. Merise 2 admet les deux types d'identifiants suivants :

- **l'identifiant absolu** constitué d'une ou plusieurs propriétés de l'entité. Dans la représentation graphique celles-ci sont soulignées. On pourra ainsi identifier une personne au moyen des trois propriétés : nom, prénom et date de naissance.

- **l'identifiant relatif** constitué de propriétés de l'entité et/ou au moins de l'identifiant d'une autre entité reliée par une association 1-n ou 1-1. L'exemple suivant illustre une telle notion et introduit sa représentation graphique.



Remarque : l'association entre les deux entités doit être stable, c'est-à-dire qu'une fois un lien établi entre deux occurrences, celui-ci ne doit plus être modifié dans le temps.

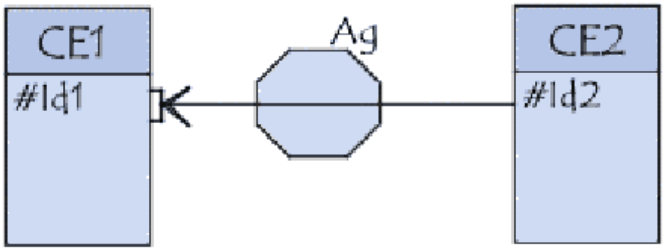
La notion d'identifiant relatif permet aussi d'exprimer un lien entre une association et une ou plusieurs entités. Certains auteurs appellent une telle association pseudo-entité ou **agrégation**. Lorsqu'un identifiant est constitué uniquement d'attributs intrinsèques à une entité, c'est-à-dire ne faisant référence à aucune autre entité, on le nomme identifiant absolu. Les entités comportant des identifiants absolus peuvent être définies indépendamment des autres occurrences d'entités, on dit que ces entités sont indépendantes. Certaines entités ne peuvent toutefois être identifiées que par l'intermédiaire d'autres entités, c'est la raison pour laquelle on parle d'identification

relative. On parlera par exemple de la 4ème porte au 2ème étage du bâtiment B au lieu de dire la porte n°3451... Ainsi, l'agrégation (appelée aussi identification relative) permet de spécifier qu'une entité est nécessaire pour en identifier une autre.

- La classe d'entité permettant d'identifier est appelé classe d'entité agrégeante

- La classe d'entité identifiée est appelée classe d'entité agrégée

La représentation de ce type de relation est la suivante :



III.3. DE DIAGRAMME ENTITE-ASSOCIATION VERS LE DIAGRAMME DE CLASSES

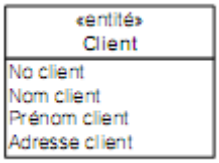
UML est suffisamment souple pour permettre qu'une notation serve à exprimer plusieurs types de modèles. Ainsi, en UML, la notation utilisée pour formuler un modèle conceptuel de données est appelée diagramme de classes. Un diagramme de classes constitué essentiellement de cases rectangulaires : les classes ainsi que d'arcs reliant les cases et affichant des multiplicités, peut exprimer :

- **Point de vue conceptuel** : le diagramme représente alors un modèle entité-association et est interprété comme décrivant les entités du monde réel ou du domaine à l'étude. Le diagramme, s'il s'inscrit dans le contexte de la conception d'une base de données, est généralement appelé un modèle conceptuel de données. En principe, chaque case du diagramme devrait porter la mention Entity. Cependant, en pratique, si le diagramme porte le nom modèle conceptuel de données, il n'est pas requis d'inscrire la mention Entity dans chaque case. Si le modèle vise par ailleurs la réalisation d'un logiciel, il sera appelé un modèle du domaine où chaque case représente alors une classe conceptuelle du domaine.
- **Spécifications logicielles** : le diagramme représente un modèle objet, c'est-à-dire qu'il illustre des composants logiciels avec des spécifications et des interfaces qui pourront être programmées dans tout langage orienté objet comme par exemple C# ou Java. Chaque case représente une classe d'objets telle que définie dans un langage orienté objet.
- **Implémentation logicielle** : le diagramme décrit un modèle d'implantation du logiciel dans un langage orienté objet en spécifiant le comportement du logiciel grâce à des techniques spécifiques au langage.

N.B. : Une autre notation que le diagramme de classes, c'est le diagramme de cas d'utilisation d'UML. Elle est utilisée dans le contexte de l'étude d'une démarche systématique d'analyse, de conception et de réalisation d'une base de données.

Entité

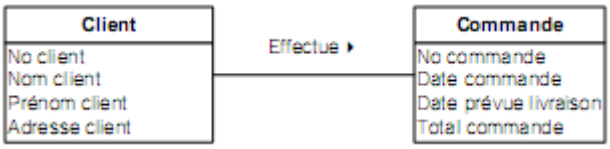
Dans le langage UML le rectangle peut représenter divers concepts, par exemple une classe d'objets dans un diagramme de classes. En matière de modélisation conceptuelle avec la notation UML, le rectangle représente des entités ou plus précisément une entité type.



Cette figure montre l'entité type Client qui décrit la structure commune pour l'ensemble des entités Client notamment les attributs que chaque entité de ce type possède : No client, Nom client, Prénom client et Adresse client.

Association

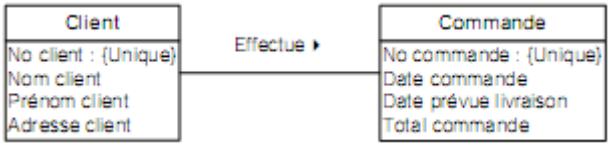
Tout comme une entité appartient à une entité type, une association appartient à une association type illustrée par un arc entre des entités types dans un modèle conceptuel. On notera que l'association est identifiée, en UML, à l'aide d'un verbe d'action qu'une flèche donne une direction à la lecture du modèle, ce qui permet de faire une lecture de l'association comme s'il s'agissait d'une phrase comportant un sujet, un verbe et un complément.



La figure suivante présente un modèle où les entités Client et Commande sont liées par une association portant le nom Effectue. L'association est définie par un arc allant de la gauche vers la droite. Celle-ci est dite binaire car elle met en cause deux entités. Il existe aussi des associations sur plus de deux entités, dites de degré supérieur.

Identifiant

Pour marquer un identifiant, dans la notation UML, l'attribut est marqué d'une contrainte, placée à la suite du nom de l'attribut, qui indique que la valeur de cet attribut est unique, c'est-à-dire qu'elle ne peut être prise par deux occurrences de la même entité. En UML, une contrainte est inscrite entre accolades comme le montre la figure suivante. Ici la contrainte {Unique} marque l'attribut agissant comme identifiant.



La présence d'un identifiant marqué de la contrainte {Unique} pour une entité représente une contrainte d'intégrité d'entité.

Lorsqu'un identifiant est formé d'un seul attribut, on le qualifie d'identifiant simple. S'il s'avère qu'un identifiant doive comporter plusieurs attributs, une contrainte {Unique} sera placée avant tous les attributs de l'entité indiquant en cela quels sont les attributs qui forment l'identifiant. On parle alors d'un identifiant composé.

Multiplicité d'une association binaire

Un autre type de contrainte doit être ajouté au modèle conceptuel dès lors que les identifiants des entités associées sont établis. Il s'agit des contraintes de multiplicité qui précisent à la fois que la participation d'une entité à l'association est obligatoire ou non, et le cas échéant, le nombre d'occurrences d'une association auxquelles elle peut participer.

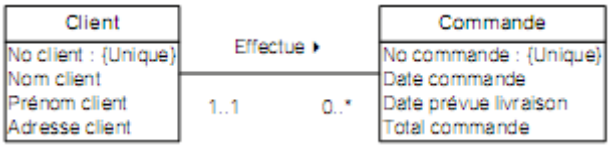
Contrainte inscrite à chaque extrémité d'une association binaire comportant un couple de valeurs (minimum - maximum) qui établit, pour chaque entité de l'association, les nombres minimum et maximum d'occurrences de l'autre entité qui peuvent lui être associées (Multiplicity).

Les contraintes de multiplicité dépendent fortement des règles 0d'affaires (business rules) d'une organisation. Il est fondamental que le concepteur et l'ensemble des utilisateurs du système aient une compréhension commune de ces règles pour que les contraintes de multiplicité soient valables et s'avèrent en conséquence pertinentes.

Multiplicité UML	Signification
`0..1`	Au plus un
`1..1 (ou 1)`	Un seul
`0..* (ou *)`	Un nombre indéterminé
`1..*`	Au moins un

Ce tableau donne les quatre combinaisons de base du couple de valeurs. Le premier chiffre (0 ou 1), indique qu'il s'agit d'une association optionnelle (0) ou d'une association obligatoire (1). Le deuxième chiffre indique le nombre maximum d'occurrences des associations auxquelles une entité participe : une (1) ou strictement plus d'une, c'est-à-dire plusieurs (*). L'astérisque représente en effet la valeur plusieurs.

Exemple d'un modèle conceptuel avec multiplicités



Dans le modèle ci-haut, le concepteur a été tenu de répondre à deux questions qui ont conduit à deux réponses différentes :

1. « Un client est-il obligatoirement lié à une commande et à combien au maximum ? »
2. « Une commande est-elle obligatoirement liée à un client et à combien au maximum ? »

Attributs d'une association

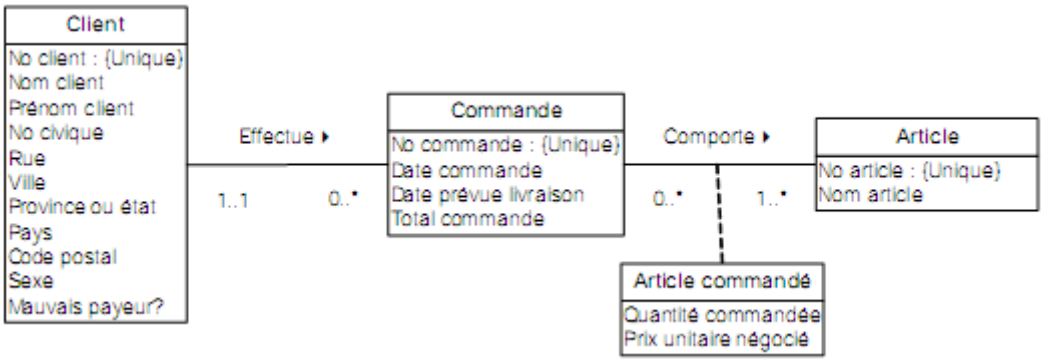
Un attribut qui dépend fonctionnellement de deux entités ou plus doit être considéré comme un attribut de l'association.

Les attributs d'une association sont représentés en UML à l'aide d'une entité rattachée à l'association par une ligne pointillée. On pourrait qualifier cette entité d'entité d'association.

Il peut être nécessaire de définir des entités d'association pour regrouper des attributs qui ne peuvent appartenir ni à l'une ni à l'autre des entités associées et éviter ainsi de transgresser la règle de construction. Bien qu'il s'agisse sur le plan sémantique d'une entité, son identifiant n'y est généralement pas inscrit car il résulte implicitement de la combinaison des identifiants des entités associées. Une entité d'association est en fait ce que les concepteurs appellent une entité faible.

Entité faible est un type d'entité dont l'existence dépend de deux ou plusieurs entités. Son identifiant se définit en fonction des identifiants des entités dont elle dépend. Elle ne possède pas d'attribut qui puisse servir d'identifiant et qui lui appartienne en propre.

Exemple de Modèle conceptuel final avec une entité d'association : Article commandé



Article mène en effet à une seule occurrence d'Article commandé. Celle-ci est dès lors considérée comme une entité faible et le concepteur a donc fait un bon choix en la considérant comme une entité d'association.

Associations de degré supérieur

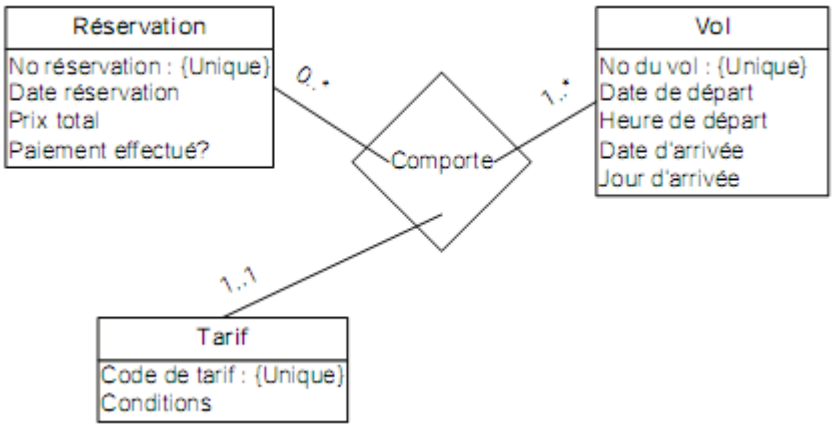
Les associations liant seulement deux entités, sont dites binaires. Celles liant plus de deux entités sont dites de degré supérieur. Ces dernières sont représentées à l'aide d'un grand losange où des arcs reliant les entités impliquées dans l'association sont marqués chacun de multiplicités.

Une occurrence d'une association de degré supérieur, tout comme une occurrence d'association binaire, est susceptible d'être liée à une occurrence de chacune des entités associées. Chaque arc doit disposer impérativement d'une multiplicité minimale et d'une multiplicité maximale.

Déterminer les multiplicités dans une association de degré supérieur est un exercice intellectuel plus exigeant que celui déployé pour une association binaire.

Dans une association binaire, la multiplicité la plus près de l'entité indique combien d'occurrences de cette entité peuvent être associées à une occurrence de l'autre entité. Dans le cas d'une association de degré supérieur n, la multiplicité près d'une entité indique combien d'occurrences de cette entité peuvent être associées à une combinaison mettant en cause une occurrence de chacune des n–1 autres entités.

L'exemple suivant tiré du modèle sur la société aérienne où il est stipulé qu'une réservation comporte des vols et des tarifs. Pour la compréhension du domaine, précisons qu'une réservation peut comporter plusieurs vols différents, par exemple Québec-Montréal, Montréal-Paris, Paris-Toronto, Toronto- Québec Sur chaque vol de la réservation, un tarif est appliqué et celui-ci peut varier d'un vol à l'autre. Un tarif est identifié par un code qui précise les conditions rattachées au prix du billet. Par exemple le tarif YHJLI pourrait avoir comme conditions : classe économique, non remboursable, payé 30 jours avant le départ.



Multiplicité du côté Réservation : Combien de réservations peuvent être associées à une occurrence du couple Tarif-Vol ? Considérons la combinaison Code de tarif = YHJLI avec No du vol = AC2033, combien de réservations peuvent être faites avec cette combinaison ? Aucune ou plusieurs, donc multiplicité 0..*. Pourquoi ? Il est possible qu'aucune réservation n'ait été accordée avec ce code de tarif pour ce vol ou bien plusieurs peuvent l'avoir été

Multiplicité du côté Tarif : Combien de tarifs peuvent être associés à une occurrence du couple Réservation-Vol ? Considérons la combinaison No réservation = 4567725653 avec No du vol = AC2033, combien de tarifs peuvent être appliqués à cette combinaison ? Une et une seule, donc multiplicité 1..1. Pourquoi ? On ne retrouve qu'un seul tarif pour un vol dans une réservation.

Multiplicité du côté Vol : Combien de vols peuvent être associés à une occurrence du couple Réservation-Tarif ? Considérons la combinaison No réservation = 4567725653 avec Code de tarif = YHJLI, combien de vols peuvent être impliqués dans cette combinaison ? Un ou plusieurs, donc multiplicité

1..*. Pourquoi ? S'il existe un code de tarif associé à une réservation, il doit y avoir au moins un vol dans la réservation avec ce tarif ou plusieurs.

Décomposition des associations de degré supérieur

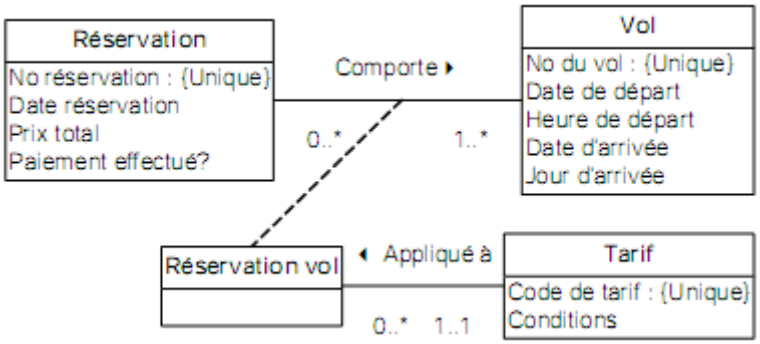
Il existe deux situations principales où une association de degré supérieur peut être divisée en deux associations de degré moindre. La première concerne une dépendance fonctionnelle entre les identifiants de deux entités de l'association, la seconde la présence d'une multiplicité maximale de 1 sur un des arcs de l'association. Ces deux conditions se rencontrent souvent simultanément dans une association de degré supérieur.

Dans la figure précédente, bien qu'une multiplicité maximale valant 1 se trouve côté Tarif, cette entité ne dépend fonctionnellement ni de Réservation, ni de Vol. En effet Vol ne détermine pas le Tarif car sur le même vol plusieurs tarifs sont applicables.

La Réservation ne détermine pas le tarif car une réservation comporte des vols avec des tarifs différents. Cependant Tarif dépend fonctionnellement à la fois de Réservation et Vol. La combinaison Réservation-Vol mène indubitablement à un seul tarif considérant la multiplicité 1..1 du côté de Tarif.

L'association de degré trois peut être théoriquement ramenée à deux associations binaires : une première liant les deux entités dont dépend la troisième. Cette nouvelle association aura une entité d'association artificielle sans attribut explicite qui sera associée à la troisième entité, soit Tarif. Par ailleurs, si une entité d'association avait été rattachée à l'association de degré 3 dans le modèle d'origine, la création d'une entité

artificielle n'aurait point été requise. Il aurait suffi d'associer l'entité Tarif à cette entité d'association et de réduire à deux le degré de l'association initiale.



CHAPITRE IV. LE NIVEAU LOGIQUE : DU RELATIONNEL A L'OBJET

IV.1. ITRODUCTION

L'objectif du niveau logique est la définition des moyens informatiques à disposition des postes de travail (utilisateurs) afin d'effectuer les opérations organisées.

Chaque outil informatique "transactionnel" se décrit sous la forme d'enchaînement d'états (MLT) affichant des informations et prêt à en saisir d'autres.

La spécification externe comprend la description des états et des informations affichées et saisies approuvée par l'utilisateur final. La spécification interne comprend la description des actions de création des informations du MLD (enregistrements, informations et chemins d'accès).

IV.2. LE MODELE LOGIQUE DES DONNEES (MLD)

Le modèle logique de données (MLD) décrit les structures de données indépendamment de la gestion physique des bases de données.

Un premier MLD se déduit d'un MOD (Modèle Organisationnel de Données) ou d'un MCD (Modèle Conceptuel de Données). Il est ensuite optimisé ou modifié suivant le choix de l'utilisateur pour accélérer certains traitements effectués par les outils informatiques.

Le MCD ou MOD étant établi, on peut le traduire en différents systèmes logiques, comme les systèmes par fichiers ou les bases de données (Avec les variantes basées sur les modèles des données). Nous retenons ici l'approche fondée sur le modèle relationnel.

Un modèle relationnel de données est un modèle logique de données spécifiant un schéma pour une base de données relationnelle soit : les tables, les champs de chaque table et leurs propriétés, la clé primaire des tables, les clés étrangères assurant les liaisons entre les tables et les contraintes d'intégrité portant sur ces liaisons (Relational Data model). C'est donc un ensemble de schémas relationnels de la forme :

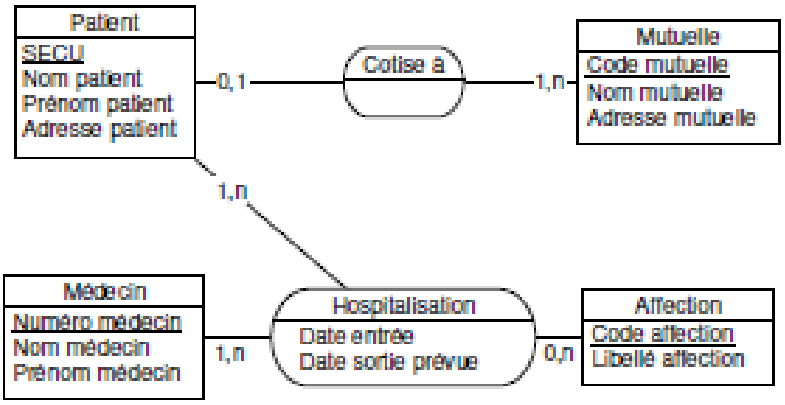
Relation (clé1, ... clén, attribut1, ... attributm)

Par convention, on souligne les clés primaires et on fait précéder les clés étrangères d'un dièse # dans la description des éléments de la relation.

En somme un modèle relationnel de données n'est qu'un cas particulier de modèle logique de données. Un modèle réseau de données ou un modèle hiérarchique de données font aussi partie des modèles de données de niveau logique.

a. Passage du MCD (ou MOD) au MLD relationnel

Considérons le modèle conceptuel des données suivant :



Règle 1 : Chaque entité avec au moins un attribut non identifiant donne lieu à un schéma relationnel (table), les identifiants deviennent les clés.

Patient (SECU, NomPatient, PrenomPatient, AdressePatient)

Médecin (NuméroMédecin, NomMédecin, PrénomMédecin)

Mutuelle (CodeMutelle, NomMutuelle, AdresseMutuelle)

Affection (CodeAffection, LibelléAffection)

Règle 2 : Les associations binaires de type 1: n donnent lieu à l'ajout de l'identifiant côté 1 vers le côté n, en tant qu'attribut non-clé (ou étrangère). Cette clé étrangère ne peut recevoir la valeur vide si la cardinalité est 1, 1.

Patient (SECU, NomPatient, PrenomPatient, AdressePatient, #CodeMutuelle)

Dans cas où l'association avait des attributs, ceux-ci glissent vers le schéma relationnel (table) côté 1.

Règle 3 : Les associations binaires de type n:m ou les associations non binaires donnent lieu à la création de nouveaux schémas relationnels supplémentaires (tables de jointure).

– Les identifiants des entités liées deviennent des clés.

– Les propriétés de l'association deviennent des attributs simples.

Hospitalisation (NuméroMedecin, SECU, CodeAffection, DateEntrée, DateSortie)

Règle 4 : Les associations binaires de type 1:1 sont traduites comme des associations binaires de type 1: n, sauf que la clé se voit imposer une contrainte d'unicité (sans doublon) en plus d'une éventuelle contrainte de non vacuité.

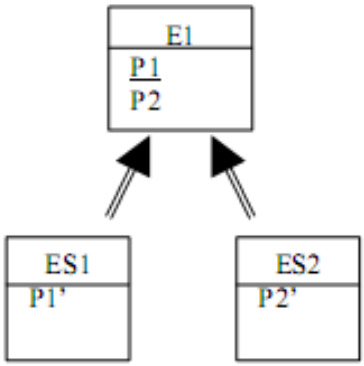
Si d'un côté, on a la cardinalité 0, 1. C'est dans la table côté opposé que devra aller la clé étrangère. Si les deux côtés sont de cardinalité 0, 1 alors la clé étrangère peut être placée indifféremment dans l'une de deux tables.

Une alternative consiste à voir une association binaire de type 1 : 1 comme une association bina ire de type n : m particulière, il suffit pour cela d'ajouter une

contrainte d'unicité sur chacune des clés étrangères de la table de jointure supplémentaire.

b. Les concepts étendus

En ce qui concerne les concepts étendus, mis à part la notion d'identifiant relatif, leur implantation en relationnel n'est pas directement réalisable, car il est impossible de représenter les contraintes ensemblistes. Il faudra donc mettre en place, au niveau des traitements, des dispositifs pour garantir toutes les contraintes qu'ils expriment. La suite de ce paragraphe présente trois possibilités de traduction du concept d'héritage rappelé par le schéma ci-dessous :



b.1) Table sur-type et disparition des sous-types

Première possibilité : intégration des sous-types dans la relation sur-type (les sous-types disparaissent). Avec un tel principe les propriétés spécifiques à chacun des sous-types ne seront pas valorisées pour certaines occurrences de la relation sur-type.

E1 (P1, P2, P1', P2')

b.2) Table sous-types et disparition du sur-type

Seconde possibilité : intégration des propriétés figurant dans le sur-type dans tous les sous-types (le sur-type disparaît). Cette solution entraîne une redondance importante des données du sur-type s'il n'y a pas exclusion entre les sous-types.

ES1 (P1, P2, P1')

ES2 (P1, P2, P2')

b.3) Table sur-type et sous-types

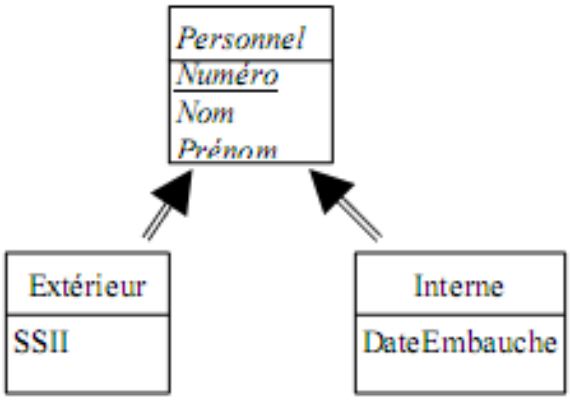
Troisième et dernière possibilité : conservation de l'entité sur-type et des entités sous-types. Dans chacune des relations sous-types, l'identifiant de l'entité sur-type est intégré. Il est à la fois clé primaire de la relation et clé étrangère par rapport à l'entité sur-type.

E1 (P1, P2)

ES1 (#P1, P1')

ES2 (#P1, P2')

Il est important de noter que quel que soit la solution adoptée, toute la puissance portée par *le concept d'héritage est perdue dans le modèle relationnel*. L'exemple ci-dessous illustre ces trois possibilités pour le modèle conceptuel décrivant la composition du service informatique.



Exemple d'occurrences :

Internes

1 - Annick Lassus (14/06/1999)

2 – Armelle Mundubeltz (20/09/2000)

Extérieur

3 – Bernadette Chaulet (CAP GEMINI)

Première possibilité : PERSONNEL (Numéro, Nom, Prénom, SSII, DateEmbauche)

PERSONNEL

Numéro	*Nom*	*Prénom*	*SSII*	*DateEmbauche*
1	*Lassus*	*Annick*		*14/06/1999*
2	*Mundubeltz*	*Armelle*		*20/09/2000*
3	*Chaulet*	*Bernadette*	*CAP GEMINI*	

Seconde possibilité :

EXTERIEUR (Numéro, Nom, Prénom, SSII)

INTERNE (Numéro, Nom, Prénom, DateEmbauche)

EXTERIEUR

Numéro	*Nom*	*Prénom*	*SSII*
3	*Chaulet*	*Bernadette*	*CAP ..*

INTERNE

Numéro	*Nom*	*Prénom*	*DateEmbauche*
1	*Lassus*	*Annick*	*14/06/1999*
2	*Mundubeltz*	*Armelle*	*20/09/2000*

Troisième possibilité : PERSONNEL (Numéro, Nom, Prénom)

EXTERIEUR (#Numéro, SSII)

INTERNE (#Numéro, DateEmbauche)

EXTERIEUR

Numéro	*SSII*
3	*CAP ..*

INTERNE

Numéro	*DateEmbauche*
1	*14/06/1999*
2	*20/09/2000*

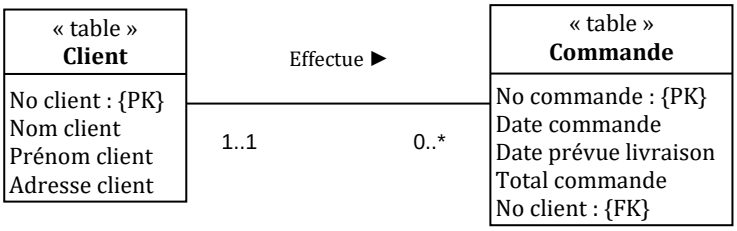
PERSONNEL

Numéro	*Nom*	*Prénom*
1	*Lassus*	*Annick*
2	*Mundubeltz*	*Armelle*
3	*Chaulet*	*Bernadette*

IV.3. NOTATION UML ET MODELE RELATIONNEL DE DONNEES

Tout comme pour le modèle conceptuel, le modèle relationnel est formulé par le biais du diagramme de classes de UML. Ici encore on mise sur le concept de prototype du langage UML pour donner au diagramme de classes et à la case rectangulaire une sémantique particulière.

Tout comme pour le modèle conceptuel, la case rectangulaire d'un modèle relationnel porte une mention de prototype. Cette fois il s'agit de la mention « table » comme le montre la figure suivante où est présentée la version graphique du schéma relationnel de la figure précédente. Dans un modèle relationnel, les associations sont toutes des associations binaires portant des contraintes de multiplicité entre des tables, d'où la présence de la mention « table » en guise de prototype. Toutefois, cette mention peut être masquée.



Dans un modèle relationnel les associations entre les tables n'ont pas à être nommées. Elles indiquent simplement qu'il existe un lien entre les deux tables et que la clé primaire d'une table marquée {PK} (pour Primary Key) est reproduite dans l'autre table sous forme de clé étrangère. Une clé étrangère est marquée {FK} (pour Foreign Key).

La présence d'un attribut avec la mention {PK} dans une table reflète une contrainte d'intégrité appelée contrainte d'intégrité d'entité.

Un corollaire de cette contrainte d'intégrité est que la mention {PK} devrait toujours être suivie de la contrainte {Non nul} dans un modèle relationnel. Encore une fois, puisque {PK} sous-entend {Non nul}, on inscrit rarement les deux mentions dans un modèle relationnel de manière à alléger le diagramme.

Toutes les tables devraient posséder une clé primaire et les associations entre les tables assurées par ces clés seront de deux types :

1. les multiplicités maximales sont 1 d'une part et 1 de l'autre, on parle ici d'une association un à un ;

2. les multiplicités maximales sont 1 d'une part et * de l'autre, on parle d'une association un à plusieurs.

Contrairement au modèle conceptuel, le modèle relationnel ne supporte que les associations binaires un à un et un à plusieurs donc aucune association plusieurs à plusieurs n'est acceptable dans ce dernier type de modèle. Il s'agit d'une erreur grave sur le plan sémantique d'avoir des multiplicités maximales * de part et d'autre d'une association dans un modèle relationnel. La raison est simple : comme un attribut clé étrangère ne peut avoir qu'une seule valeur dans une ligne donnée de la table, elle ne peut assurer la liaison de cette ligne qu'à une seule autre ligne de la deuxième table. La multiplicité maximale est donc systématiquement à 1 du côté de la table possédant la clé primaire qui a été copiée dans l'autre table avec la mention {FK} (clé étrangère).

Un modèle relationnel complet devrait notamment faire état du domaine de chaque attribut. Cela est habituellement réalisé en inscrivant à la suite des attributs et des mentions {PK} et {FK} le cas échéant, le type de données de l'attribut et les contraintes d'intégrité de l'attribut. Par exemple, si la clé primaire porte sur un seul attribut, il s'agit donc d'une clé simple. Cet attribut devrait avoir les contraintes d'intégrité {Unique} et {Non nul}. Rappelons qu'une clé primaire est par définition obligatoire et non redondante.

Règles de dérivation des relations à partir d'un MCD

- *Le cas des entités*

Pour chaque entité du modèle conceptuel, une table est créée dans le modèle relationnel, sauf dans le cas d'une association d'héritage où les attributs des entités en cause peuvent être combinés pour former une table. Les attributs de l'entité deviennent les attributs de la table et, dans le cas d'une entité forte, son identifiant, simple ou composé, devient la clé primaire de la table. Que la clé primaire soit simple ou composée, chaque attribut formant la clé porte la mention {PK}.

- *Dérivation à partir d'une entité d'association*

En ce qui concerne une entité d'association, la table correspondante aura comme clé primaire, une clé obligatoirement composée des attributs de tous les identifiants des entités participant à l'association et chaque attribut de la clé composée portera la mention {PK, FK}.

La mention {Cascade} peut accompagner les clés étrangères présentes dans la table dérivée de l'entité d'association pour faire état d'une contrainte d'intégrité fort importante appelée contrainte d'intégrité référentielle (Referential integrity).

La mention {Cascade} comporte des restrictions supplémentaires :

1 si une ligne de la table mère est supprimée, toutes les lignes associées à celle-ci via la clé étrangère dans la table fille sont aussi supprimées (contrainte de suppression en cascade) ;

2 si la valeur de la clé primaire d'une ligne est modifiée, les valeurs des clés étrangères des lignes associées dans la table fille sont modifiées pour continuer d'assurer la liaison (contrainte de mise à jour en cascade).

Ces contraintes sont incontournables dans le cas d'une entité faible, car son existence dépend de l'existence d'une entité forte. Logiquement toute occurrence d'entité forte qui est détruite entraîne avec elle la disparition des entités faibles qui en dépendent.

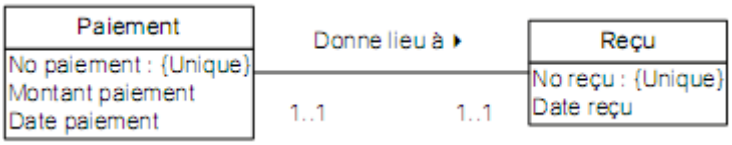
Les associations binaires

- *Association binaire un à un*

Dans un tel cas, la dérivation de la clé étrangère repose sur l'analyse des multiplicités minimales de l'association. Cela consiste à établir si la participation des occurrences à l'association est obligatoire d'un côté ou de l'autre ou des deux à la fois.

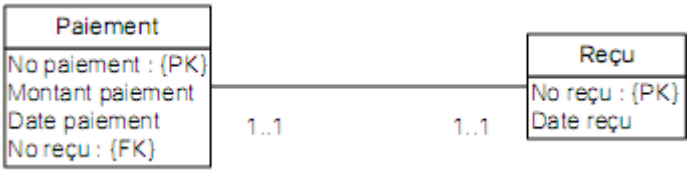
Si la participation est obligatoire des deux côtés de l'association (multiplicités minimales 1 de part et d'autre), l'une ou l'autre des tables dérivées des entités peut agir comme table mère. Cependant, si une seule des deux entités est associée à une troisième, on choisira comme table fille celle dérivée de l'entité qui comporte une deuxième association. Si toutes deux sont associées à une autre, on choisit comme table fille celle qui possède une clé primaire composée.

Considérons la figure suivante où une association binaire montre des multiplicités 1..1 de part et d'autre des entités concernées.



Pour ce modèle, nous faisons l'hypothèse que l'entité Paiement est associée par ailleurs à l'entité Client (non illustrée). La table fille sera donc Paiement, la table mère sera Reçu, et la clé primaire de la table mère, No reçu, est reproduite dans la table fille à titre de clé étrangère (cfr. figure suivante). Les multiplicités sont reprises telles quelles du modèle conceptuel.

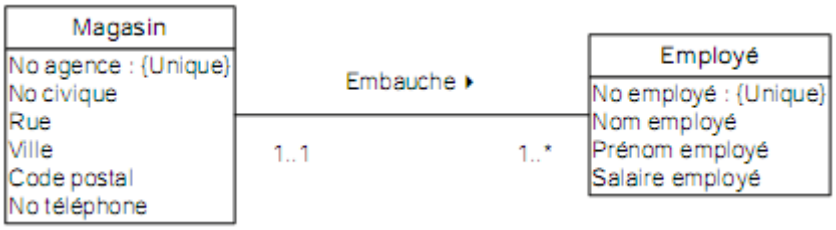
Sans cette hypothèse, la table fille aurait tout aussi bien pu être Reçu.



Si la participation est optionnelle sur au moins un côté de l'association (multiplicité minimale 0 d'un côté ou de l'autre), la table dérivée de l'entité qui a la participation optionnelle devient la table fille. Une copie de la clé primaire de la table mère est déposée dans la table fille à titre de clé étrangère.

- *Association binaire un à plusieurs*

Les associations un à plusieurs sont les plus nombreuses dans un modèle conceptuel. Sa règle de dérivation se résume ainsi : la table dérivée de l'entité du côté de la multiplicité maximale plusieurs (*) est considérée la table fille. En conséquence, une copie de la clé primaire de la deuxième table, la table mère, est déposée dans la table fille à titre de clé étrangère. Cette règle ne comporte aucune exception



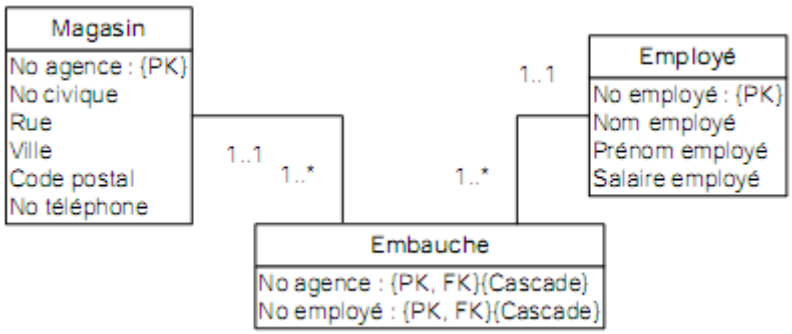
La figure ci-haute montre un modèle conceptuel avec une association binaire un à plusieurs où la participation est obligatoire de part et d'autre. La participation ne joue aucun rôle dans la dérivation du modèle relationnel pour ce type d'association. La table dérivée de Employé devient la table fille, car elle est côté plusieurs, et la table Magasin, la table mère. La clé primaire de Magasin, No agence, devra apparaître dans Employé sous forme de clé étrangère.

La figure suivante présente le modèle relationnel dérivé.

- *Association binaire plusieurs à plusieurs*

Une association plusieurs à plusieurs sans entité d'association donne lieu à une table fille comportant comme clé primaire la combinaison des clés primaires des tables mères. Chaque composant de la clé primaire est aussi clé étrangère. Aucun autre attribut n'est présent dans la table. Deux tables mères sont présentes et deux associations vont assurer la liaison avec leur table fille. Chaque association portera une multiplicité 1..1 du côté de la table mère et quant aux multiplicités du côté de la table fille, elles sont reprises du modèle conceptuel comme pour une entité d'association. S'il y a au modèle conceptuel la multiplicité 0..* sur la terminaison d'arrivée de l'association lue de A vers B, il y aura multiplicité 0..* du côté de la table fille pour l'association la liant à A. S'il y a au modèle conceptuel la multiplicité 1..* sur la terminaison d'arrivée de l'association lue de B vers A, il y aura multiplicité 1..* du côté de la table fille pour l'association la liant à B.

Si nous voulons que le modèle de la figure sur l'association binaire un à plusieurs reflète un contexte plus général et moins restrictif où un employé peut être embauché dans plusieurs magasins, nous aurions alors une association binaire entre Magasin et Employé du type plusieurs à plusieurs Le modèle relationnel résultant, figure suivante, comporterait alors une nouvelle table qui porterait le nom de l'association et dont les attributs se résument à la combinaison des clés primaires des tables mères. Chaque attribut est une clé étrangère pour laquelle une contrainte d'intégrité en ajout, suppression et mise à jour devra être appliquée.



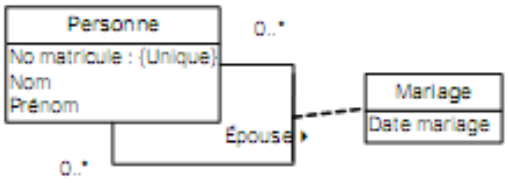
N. B. : Il y a un doute sur le fait que la clé primaire de la table fille dérivée d'une association binaire plusieurs à plusieurs exclue tout doublon

Association réflexive

La dérivation d'une association réflexive reprend les règles évoquées plus haut en les spécialisant pour tenir compte du fait qu'une entité est associée à elle-même.

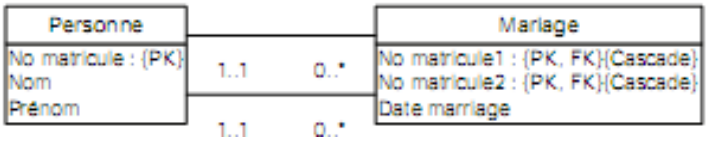
- *Cas 1 : L'association réflexive comporte une entité d'association*

En vertu des règles données plus tôt, l'entité d'association donne lieu à une table fille associée à la table mère dérivée de la seule entité en cause. La table dérivée de l'entité d'association comporte les attributs de cette dernière avec comme clé primaire deux exemplaires de la clé primaire de la table mère, les exemplaires portent un nom différent pour éviter toute redondance. Chaque exemplaire est aussi une clé étrangère avec la mention {Cascade} pour assurer l'intégrité référentielle.



Ce modèle permet de faire état de tous les mariages qu'une personne peut avoir contractés. On fait l'hypothèse qu'une personne ne peut épouser la même personne qu'une seule fois. Chaque mariage peut donc être identifié de manière unique par la combinaison des matricules des deux personnes qui s'épousent à la date donnée.

La table fille Mariage est dérivée de l'entité d'association. Elle comporte une clé primaire composée des deux numéros matricules des personnes contractant un mariage. Chaque matricule porte un nom d'attribut différent. À chaque matricule correspond une association reprenant du côté de la table fille les multiplicités du modèle conceptuel. Le matricule est aussi une clé étrangère pour laquelle l'intégrité référentielle doit être assurée en ajout, en suppression et en mise à jour.

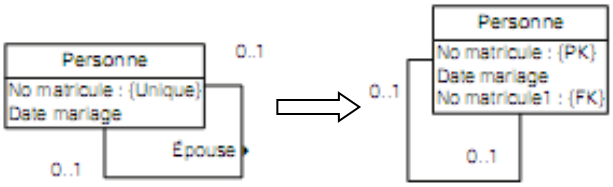


- *Cas 2 : L'association réflexive est du type un à un ou un à plusieurs*

Il s'agit d'une situation où l'association réflexive ne porte pas d'entité d'association et où les multiplicités maximales sont de 1 d'un côté ou de l'autre de l'association, ou bien des deux côtés à la fois. Dans ce cas une seule table est dérivée et elle comporte une clé étrangère. Celle-ci est un double de sa clé primaire qui doit porter un nom différent. Les multiplicités de l'association au modèle conceptuel sont reprises intégralement dans le modèle relationnel dérivé.

La figure suivante est aussi un modèle portant sur la notion de Mariage où cette fois on ne fait état que du dernier mariage contracté par une personne. En effet, on voit

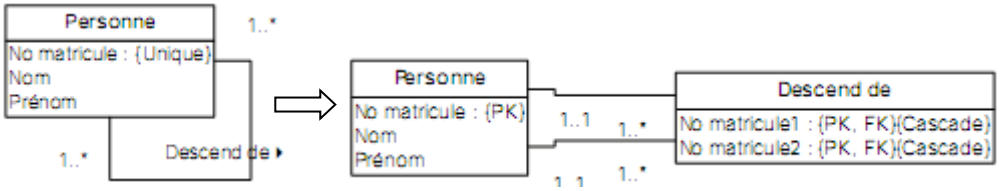
clairement qu'une personne contracte au plus un mariage et la date de ce mariage peut être considérée comme un attribut de l'entité personne. Cette association réflexive est du type un à un.



- *Cas 3 : L'association réflexive est du type plusieurs à plusieurs*

Une association réflexive plusieurs à plusieurs est traitée comme une association réflexive comportant une entité d'association. Une table doit être créée pour assurer une liaison plusieurs à plusieurs sur la même table. Cette table n'a pas d'attribut propre. Sa clé primaire est une combinaison de deux exemplaires de la clé primaire de la table dérivée de l'entité. Chaque exemplaire aura un nom différent et sera traité comme une clé étrangère avec contrainte d'intégrité référentielle en ajout, suppression et mise à jour (mention {Cascade}).

La figure suivante modélise la descendance d'une personne. Une personne descend de plusieurs personnes et par ailleurs possède plusieurs descendants.



Les associations de degré supérieur

Toute association de degré supérieur donne lieu à une table qui s'ajoute aux tables dérivées des entités associées.

S'il s'agit d'une association de degré supérieur qui comporte une entité d'association, les règles vues précédemment pour la conversion d'une entité d'association présente sur une association binaire s'appliquent :

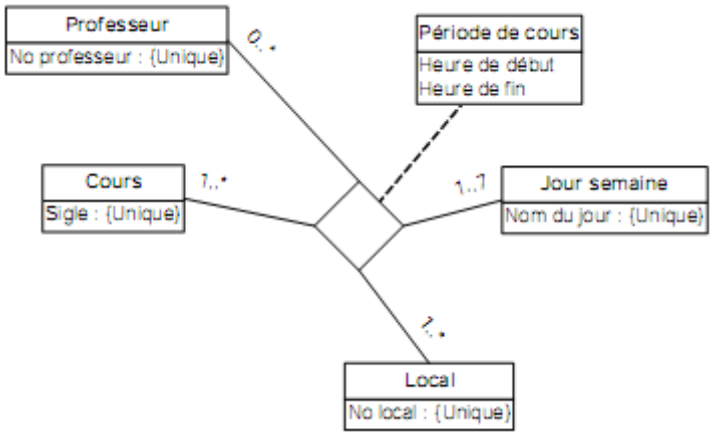
- chaque entité associée devient une table mère ;
- l'entité d'association devient une table fille ;
- la clé primaire de la table fille combine les clés primaires des tables dérivées des tables mères associées et chaque élément de la clé est une clé étrangère ;
- cependant, si une des multiplicités maximales sur les arcs vaut un (1), la clé primaire de la table fille est la combinaison des clés primaires des tables associées, sauf celle associée par l'arc retenu où une multiplicité maximale 1 est présente ;

• les clés primaires des tables mères qui n'ont pas été intégrées à la clé primaire sont présentes à titre de clés étrangères dans la table fille et les attributs de l'entité d'association s'y ajoutent. L'intégrité référentielle en ajout, suppression et mise à jour ne s'applique qu'aux clés étrangères qui font partie de la clé primaire.

• les multiplicités sont toutes à 1..1 du côté des tables mères.

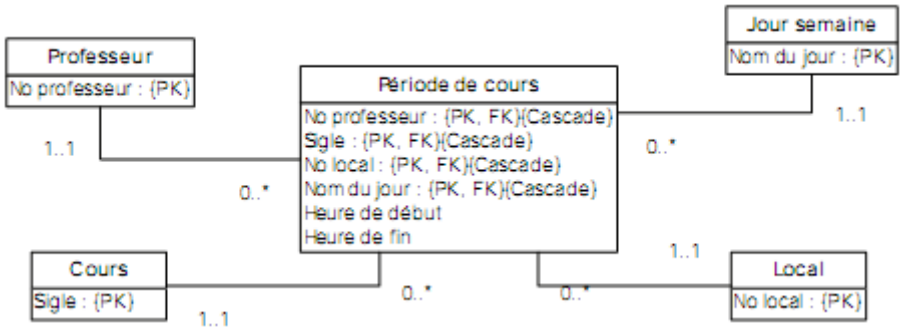
- *Cas 1 : L'association de degré supérieur comporte une entité d'association.*

Nous illustrons ici la dérivation d'une entité d'association présente sur une association de degré supérieur. La figure suivante montre une association de degré supérieur où toutes les multiplicités maximales sont plusieurs. La table fille Période de cours doit en conséquence posséder une clé primaire composée des clés primaires des quatre tables mères.



Le modèle relationnel qui en dérive, comporte cinq tables, dont une table dérivée de l'entité d'association qui possède une clé primaire composée des clés primaires des quatre autres. Cette table, Période de cours, est une table fille. Les clés primaires des quatre tables mères qui sont copiées dans la table fille sont toutes des attributs considérés comme clés étrangères sur lesquelles la contrainte d'intégrité référentielle s'applique. Les attributs propres à l'entité d'association, Heure de début et Heure de fin, apparaissent à leur suite.

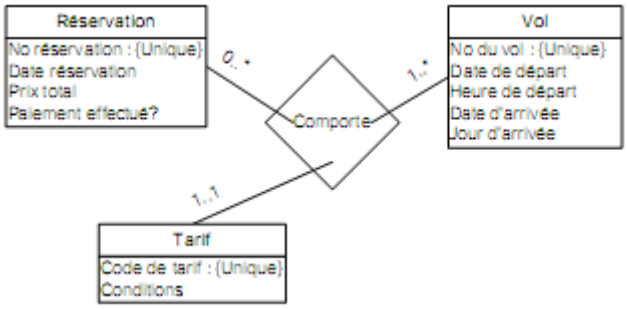
Les multiplicités placées du côté de la table fille sont toutes 0..*. Les multiplicités du côté des tables mères sont systématiquement 1..1. Puisque toutes les associations comportent des multiplicités maximales plusieurs du côté de la table fille, sa clé primaire ne peut être simplifiée.



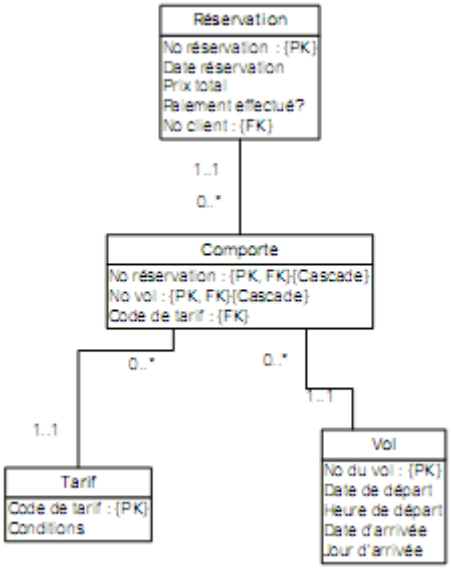
- *Cas 2 : L'association de degré supérieur n'a pas d'entité d'association*

Si l'association de degré supérieur ne comporte pas d'entité d'association, une table fille est tout de même dérivée de l'association, sans attribut propre. La table porte le nom de l'association. Sa clé primaire et les multiplicités des associations avec les tables mères sont déterminés selon les règles qui s'appliquent à une table fille dérivée d'une entité d'association présente sur une association de degré supérieur.

Cependant, une différence importante est à prendre en considération. Si les multiplicités maximales sont à plusieurs sur tous les arcs de l'association, la table fille pourrait avoir une clé primaire composée qui n'est pas valide. Il s'agit pour le concepteur d'analyser le contexte et de tenter d'établir si elle est acceptable. En cas de doute, il aura simplement à créer une clé primaire simple et artificielle avec la mention {PK auto}, tout en conservant dans la table fille les clés primaires des tables mères mais uniquement à titre de clés étrangères avec mention {FK}{Cascade}.



La figure précédente montre un modèle conceptuel avec association de degré supérieur sans entité d'association et où on retrouve au moins une multiplicité maximale 1. On peut donc effectuer une simplification de la clé primaire. La table fille nommée Comporte aura une clé primaire combinant les clés primaires des autres tables Réservation et Vol.



Dans ce modèle relationnel dérivé, on notera que la clé primaire de la table mère Tarif, Code de tarif, est tout de même copiée dans la table fille Comporte. Bien que clé étrangère, elle ne constitue cependant pas un élément de la clé primaire composée de cette dernière car la multiplicité maximale de son côté est 1. De plus la mention {Cascade} n'est pas présente, une conséquence de la simplification.

Puisque qu'une multiplicité maximale de 1 est présente sur un des arcs de l'association, il n'y a pas lieu de créer une clé primaire artificielle. La combinaison des attributs No réservation et No vol assure l'unicité d'accès à une ligne de la table.

IV.4. OPTIMISATION DU MODELE RELATIONNEL

Le modèle relationnel peut être optimisé de manière à produire un modèle physique techniquement performant. Deux mesures affectant le modèle relationnel peuvent être considérées de façon à améliorer la performance du modèle physique qui en sera dérivé.

1. Le nombre de tables peut être réduit en fusionnant certaines tables pour en arriver à éviter des opérations de jointure inutiles ;
2. Une clé primaire composée peut être remplacée par une clé primaire simple pour réduire la complexité des index et rendre plus performantes les opérations de jointure.

Deux types d'associations binaires présentes dans un modèle relationnel peuvent entraîner la fusion des deux tables concernées. Les associations un à un dont les multiplicités minimales sont 1 et 1. Les associations un à un dont les multiplicités minimales sont 0 et 1.

IV.5. OUTILS DU MARCHE : DE LA THEORIE A LA PRATIQUE AVEC ETUDES DE CAS.

5.1. BREVE PRESENTATION DE L'OUTIL DE MODELISATION CHOISI

Ce chapitre valide la démarche théorique du cours en la comparant aux principales solutions informatiques du marché qui mettent en œuvre la notation UML et l'interconnexion à une base de données (le plus souvent par un pilote ODBC ou JDBC). Les 14 outils étudiés sont: *Enterprise Architect, MagicDraw, MEGA Designer, ModelSphere, MyEclipse, Objecteering, Poseidon for UML, PowerAMC, Rational Rose Data Modeler, Together, Visio, Visual Paradigm, Visual UML et Win'Design. Vous trouverez en annexe les adresses Internet de ces produits*.

La partie consacrée aux bases de données n'est pas prépondérante pour la majorité des outils.

D'autres fonctionnalités sont offertes en ce qui concerne la modélisation de processus métier BPM (Business Process Models) qui peuvent être importés ou exportés conformément au langage BPEL4WS (Business Process Execution Language for Web Services). Bon nombre d'entre eux fournissent un référentiel pour le contrôle des données métiers utiles aux architectes des systèmes d'information qui en seront les principaux utilisateurs. Les plus récents s'inscrivent davantage dans l'architecture MDA (Model Driven Architecture) en incluant le standard QVT (Query View Transformation) pour la transformation de modèles.

Nous allons aborder deux outils (Start UML et Win'Design) avec de cas précis à l'appui.

5.2. BREVE PRESENTATION DE L'OUTIL DE MODELISATION CHOISI

Pour la modélisation, nous avons opté d'utiliser Win 'Design qui est une suite logicielle de modélisation d'entreprise.

WinDesign est un outil de modélisation comprenant 3 modules autonomes et communicants, permettant de concevoir, modéliser et spécifier chaque angle de vues des systèmes d'information.

Chacun de ces modules est dédié :

- **Module BUSINESS PROCESS/CARTOGRAPHIE SI** : Destiné aux utilisateurs métier pour la cartographie métier et aux services informatiques pour la cartographie fonctionnelle, applicative et infrastructure.

  Ce module permet de représenter tous les angles de vue des systèmes d'information et d'analyser tout impact local ou transverse, à tous les niveaux.

- **Module DATABASE** : destiné aux analystes et aux administrateurs de bases de données, le module Database de WinDesign permet de concevoir, générer, modifier, redocumenter les bases de données.

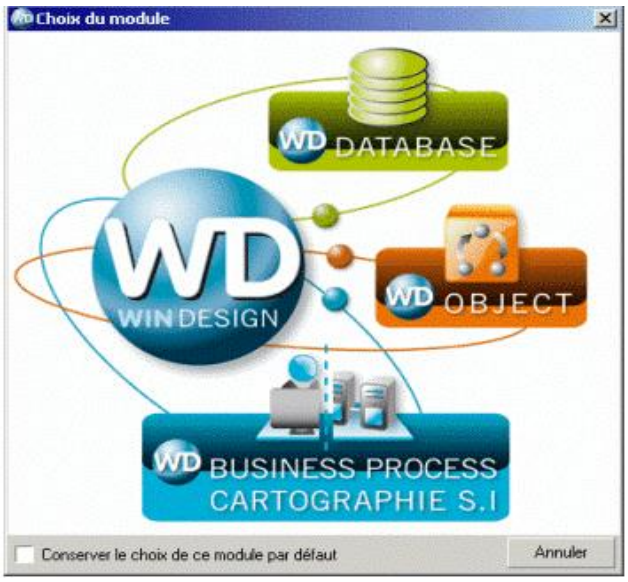
- **Module OBJECT** : destiné aux chefs de projets et développeurs, pour spécifier les points de vue statique et dynamique, d'une application ou d'un système, au travers des diagrammes et langages préconisés par UML

La licence d'utilisation de WinDesign que vous avez acquise correspond à tout ou partie de ces modules.

Lorsque vous ne disposez pas d'un module, celui-ci est accessible en mode Evaluation. Ceci est visible sur le bouton portant le nom du module, qui s'affiche alors avec la mention « Démo ». Vous pouvez utiliser un module en version d'évaluation, mais vous ne disposez pas alors des fonctions d'enregistrement.

Sélection d'un module

Cliquez sur le bouton correspondant au module gérant le type de modèle que vous souhaitez créer.



5.3. ETUDES DE CAS

1) Une société veut créer une petite base de données de gestion des commandes pour contrôler ses stocks et ainsi pouvoir alimenter ses clients de manière ininterrompue, et ce quelle que soit la demande.

- Chaque client est défini par un numéro interne à la société, un nom, un prénom et une adresse.

- La société possède une gamme de produits en catalogue, parmi lesquels les clients peuvent commander. Un produit possède une référence, un nom et un prix unitaire. Il entre dans une catégorie codée. Chaque catégorie a en plus d'un code, un nom et une description. Elle référence logiquement plusieurs produits.
- La société reçoit ainsi, à travers son vendeur, des commandes d'un client, chacune d'entre elles pouvant contenir un ou plusieurs produits. Cette commande est numérotée et datée. Le ou les produits qui la composent sont définis pour une quantité donnée.

a. De construire les modèles conceptuels de données.

b. De donner le modèle logique de données correspondants.

2) La Bibliothèque d'un syndicat intercommunal consiste en 5 centres de prêt. Ces centres disposent d'ordinateurs personnels interconnectés qui doivent permettre de gérer les emprunts.

L'interview des bibliothécaires permet de déterminer les faits suivants :

- une personne qui s'inscrit à la bibliothèque verse une caution. Suivant le montant de cette caution elle aura le droit d'effectuer en même temps de 1 à 10 emprunts;
- les emprunts durent au maximum 15 jours;
- un livre est caractérisé par son numéro dans la bibliothèque (identifiant), son titre, son éditeur et son (ses) auteur(s);
- on veut pouvoir obtenir, pour chaque abonné les emprunts qu'il a effectué (nombre, numéro et titre du livre, date de l'emprunt) au cours des trois derniers mois;
- toutes les semaines, on édite la liste des emprunteurs en retard : nom et adresse de l'abonné, date de l'emprunt, numéro(s) et titre du (des) livre(s) concerné(s);
- On veut enfin pouvoir connaître pour chaque livre sa date d'achat, son état et s'il est disponible dans quel centre.

Proposer des diagrammes entité-association qui modélisent ce cas. Précisez les contraintes d'intégrité.

3) Pour les besoins de la gestion d'un aéroport on souhaite mémoriser dans une base de données les informations nécessaires à la description des faits suivants:

- chaque avion géré est identifié par un numéro d'immatriculation. Il est la propriété soit d'une société, soit d'un particulier: dans les deux cas on doit connaître le nom, l'adresse et le numéro de téléphone du propriétaire, ainsi que la date d'achat de l'avion;
- chaque avion est d'un certain type, celui-ci étant caractérisé par son nom, le nom du constructeur, la puissance du moteur, le nombre de places;
- la maintenance des avions est assurée par les mécaniciens de l'aéroport. Par sécurité, les interventions sont toujours effectuées par deux mécaniciens (l'un répare, l'autre vérifie). Un même mécanicien peut, selon les interventions, effectuer la réparation

ou la vérification. Pour toute intervention effectuée, on conserve l'objet de l'intervention, la date et la durée;

- pour chaque mécanicien on connaît son nom, son adresse, son numéro de téléphone et les types d'avion sur lesquels il est habilité à intervenir;
- un certain nombre de pilotes sont enregistrés auprès de l'aéroport. Pour chaque pilote on connaît son nom, son adresse, son numéro de téléphone, son numéro de brevet de pilote et les types d'avion qu'il est habilité à piloter avec le nombre total de vols qu'il a effectué sur chacun de ces types.

Des questions types auxquelles l'application doit pouvoir répondre sont les suivantes:

- liste des avions de la société "Voltige";
- liste des avions qui sont la propriété de particuliers;
- durée totale des interventions faites par le mécanicien Rochat au mois de janvier;
- liste des types d'avion de plus de 4 places;
- liste des pilotes habilités pour tel type d'avion;
- liste des interventions (objet, date) faites sur l'avion numéro 3242XZY78K3.

Proposer des diagrammes entité-association qui modélisent ce cas. Précisez les contraintes d'intégrité.

4) Le club sportif d'une école veut enregistrer les informations sur ses adhérents. Il lui importe de connaître, pour chaque adhérent, le nom et prénom, la date de naissance, la commune et canton de naissance, la section et année d'études (pour les étudiants), le département (pour les enseignants), le service (pour les administratifs). On veut aussi connaître les années d'adhésion précédentes, les sports pratiqués pour l'année en cours ainsi que le niveau de l'adhérent dans chacun de ces sports.

   Proposer deux diagrammes entité-association, l'un sans lien de généralisation / spécialisation, l'autre avec.

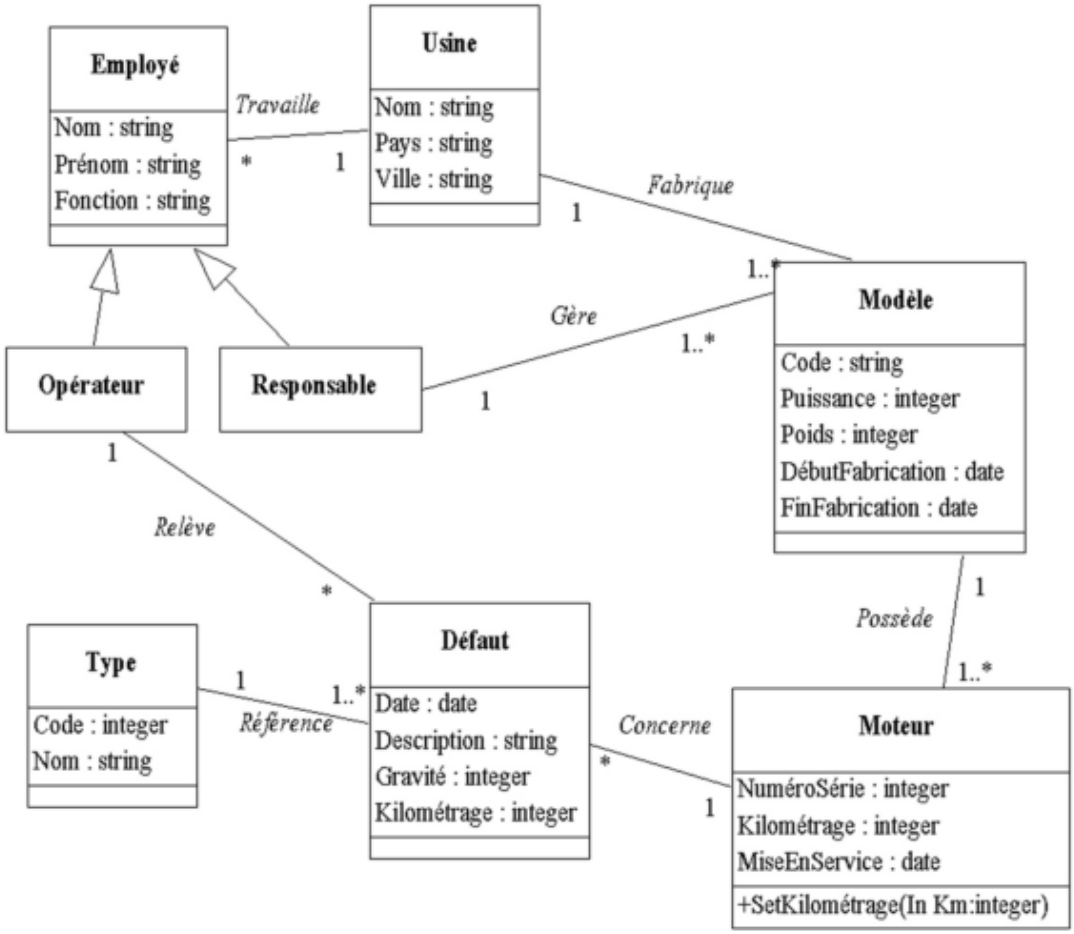
5) On veut représenter le personnel d'une entreprise et son affectation. L'entreprise est organisée en services auxquels est affecté le personnel. Chaque service est décrit par son nom, son chef (qui est nécessairement un cadre de l'entreprise) et la liste de ses locaux. Le personnel est réparti en trois catégories, les administratifs, les techniciens et les cadres. Tous possèdent un numéro d'employé, un nom, un prénom, une adresse, une identification bancaire (nom banque, nom agence, numéro de compte), un salaire et sont rattachés à un service. Chaque catégorie possède en outre des renseignements qui lui sont propres:

   - pour un administratif ou un technicien, le prix de l'heure supplémentaire;
   - pour un technicien, les machines dont il est responsable;
   - pour un administratif, le(s) cadre(s) pour le(s)quel(s) il travaille;

- pour un cadre, son bureau, son numéro de poste téléphonique et l'(les) administratif(s) (s'il en existe) qui lui est (sont) attaché(s).

Proposer des diagrammes entité-association qui modélisent ce cas.

6) Un groupe industriel construisant des moteurs cherche à organiser la gestion des défauts observés sur des moteurs confrontés à des tests en situation réelle. Pour cela un de ses ingénieurs modélise le processus de gestion des défauts, tel qu'il existe actuellement, par le diagramme de classes suivant.



Questions

a) Décrivez ce que représente ce diagramme ?

b) Etant donné ce modèle, est-il possible de savoir dans quelle usine a été fabriqué un moteur et qui est responsable de sa production ?

c) La responsabilité d'un modèle est-elle toujours assumée par un employé travaillant dans l'usine dans laquelle ce modèle est produit ?

d) Pourquoi avoir fait le choix d'une classe Type pour codifier les défauts, plutôt qu'un attribut de type énuméré directement dans la classe Défaut ?

e) Pourquoi l'attribut kilométrage apparait-il à la fois dans les classes Défaut et Moteur et pourquoi avoir fait apparaître la méthode SetKilométrage ?

f) Ce diagramme permet-il de répondre à la question : Quel est le nombre moyen de défauts rencontrés pour un moteur dont le modèle a été mis en service avant 2000 ? Quelles sont les classes et attributs utiles ?

g) Peut-on également répondre à la question : Quel est le kilométrage moyen pour lequel un moteur est concerné par au moins deux défauts de gravité supérieure à 5 ?

7) La structure de données suivante est extraite d'un document comportant des données sur le dépôt d'une candidature à un poste dans une organisation. Comme on peut le noter facilement à la consultation de la structure de données, un candidat peut pratiquer plusieurs langues, avoir de multiples centres d'intérêt, posséder plusieurs diplômes, avoir eu plusieurs employeurs chez qui il a pu exercer plusieurs fonctions.

Descriptif du document: *Dépôt de candidature*

- No affichage du poste
- Nom du poste
- No du candidat
- Nom candidat
- Prénom candidat
- No téléphone
- Date de naissance
- **Langue pratiquée (*)**
  - Langue
  - Niveau de connaissance (parlé, écrit, les deux)
- **Centre d'intérêt (*)**
  - Désignation (sport, musique,…)
- **Employeur (*)**
  - Raison sociale entreprise
  - Date d'embauche entreprise
  - Date de départ de l'entreprise
  - **Fonction exercée (*)**
    - Désignation fonction
    - Date d'entrée en fonction
    - Date de départ de la fonction
    - Salaire à la fin
- **Diplôme (*)**
  - Désignation diplôme
  - Date d'obtention
  - Nom institution

Le candidat peut avoir été réembauché plusieurs fois chez un même employeur. Le candidat peut avoir exercé la même fonction chez plusieurs employeurs et avoir exercé la même fonction plus d'une fois chez un même employeur. Il ne peut cependant pas avoir obtenu le même diplôme plus d'une fois.

Il s'agit d'élaborer un diagramme des classes (formalisme UML) qui reflète ces exigences. Il peut comporter une association de degré 3 qui peut être réduite à deux associations binaires.

8) Le service de gestion du personnel d'une entreprise désire s'équiper d'un outil lui permettant de gérer informatiquement ses employés, leurs salaires et leurs congés.

Les spécifications suivantes ont pu être mises en exergue par une analyse des besoins réalisée auprès des utilisateurs du service du personnel.

Généralités :

♦ Tout employé est identifié par un nom, un prénom et une date de naissance.

♦ Tout employé remplit une fonction et appartient à un service.

♦ Pour chaque employé on gère la date d'embauche et la quotité (c'est à dire le pourcentage de temps travaillé par rapport au temps plein, en cas de travail à temps partiel)

Gestion des salaires :

♦ Pour chaque employé on gère l'historique de ses salaires dans l'entreprise, chaque salaire étant affecté à une période de temps.

♦ Un salaire est composé d'un salaire brut, de charges patronales et de charges salariales.

♦ On cherchera à partir de ces données à obtenir également le salaire chargé (brut + charges patronales), et le salaire net (brut - charges salariales), et ce en particulier pour le salaire en cours (celui que touche actuellement le salarié).

Gestion des congés :

♦ Pour chaque employé, on mémorise chaque congé pris (posant qu'un congé concerne toujours une ou plusieurs journées entières)

♦ Chaque employé a le droit aux jours de congés suivants :

- 25 jours (pour une quotité de 1) et 25 x quotité sinon

- Chaque fonction ouvre les droits à un certain nombre de jours de RTT

- Chaque service ouvre les droits à un certain nombre de jours de RTT

- Chaque tranche de 5 ans passés dans l'entreprise donne droit à 1 jour supplémentaire

- Les employés de plus de 40 ans ont un jour supplémentaire, et ceux de plus de 50 ans deux.

♦ Pour chaque employé on cherchera à connaître le nombre total de jours de congés autorisés, le nombre de jours pris et le nombre de jours restants sur l'année en cours.

Question1

Réaliser le diagramme de classes permettant de modéliser ce problème et le modèle relationnel correspondant.

9) Vous avez en charge la réalisation d'un modèle de base de données pour la gestion d'un parc informatique.

L'analyse des besoins révèlent les informations suivantes : tout matériel informatique est identifié de façon unique par un numéro de série et est décrit par une désignation. Il existe trois types de matériel informatique : les PC, les serveurs et les imprimantes. Pour les PC les informations que l'on veut gérer sont la taille de la mémoire vive et la cadence du micro-processeur, pour les serveurs on veut gérer leur volume de disque dur et pour les imprimantes leur résolution maximale d'impression. On veut également gérer les connections réseau sachant que tout PC peut être relié à un ou plusieurs serveur et que chaque serveur sert bien entendu plusieurs PC ; et qu'un PC peut être relié à une imprimante, qui est également utilisée par plusieurs PC. Quand un PC est relié à un serveur, on veut gérer le quota de disque dont il dispose sur ce serveur.

Questions

a) Réaliser le modèle conceptuel UML de ce problème.

b) Réalisez le passage au modèle relationnel.

10) Le GAB – cas d'utilisation

Modélisation d'un GAB (Guichet Automatique de Banque). Les principales fonctions sont les suivantes :
-distribution d'argent à tout porteur d'une carte de la banque (autorisation d'un certain montant par le système d'information de la banque) ou d'une carte VISA (autorisation à distance par le système d'autorisation VISA),
-consultation du solde, dépôt en numéraire et de chèques pour les possesseurs d'une carte de la banque.
Toutes les transactions sont sécurisées (code personnel vérifié avec le code enregistré sur la puce de la carte ; la carte est avalée après trois échecs).
Il faut parfois recharger le GAB et retirer des choses …
Identifier les acteurs et les cas. Structurer les cas.

11) Le GAB – diagramme d'activités et diagramme de séquence

a) Modéliser le retrait d'argent avec une carte VISA avec un diagramme d'activités. La carte peut être invalide. Si elle est valide, le client doit taper son code. La carte est avalée après trois essaies infructueux. Le SA VISA autorise un certain montant ou refuse tout retrait. Une carte non récupérée est avalée. Les billets non récupérés par le client sont repris. Un ticket est toujours imprimé pendant que les billets sont proposés.

b) Modéliser le scénario nominal (succès) avec un diagramme de séquence.

12) Le cirque – diagramme de classes

Le propriétaire d'un cirque souhaite informatiser une partie de la gestion de ses spectacles. Proposer un modèle conceptuel UML (diagramme de classes) qui réponde aux spécifications, fournies ci-dessous.
Les membres du personnel du cirque sont caractérisés par un numéro (en général leur numéro INSEE), leur nom, leur prénom, leur date de naissance et leur salaire. On souhaite de surcroît stocker les pseudonymes des artistes et le numéro du permis de conduire des chauffeurs de poids lourds.
Les artistes sont susceptibles d'assurer plusieurs numéros, chaque numéro étant caractérisé par un code, son nom, le nombre d'artistes présents sur scène et sa durée. De plus, on souhaite savoir l'instrument utilisé pour les numéros musicaux, l'animal concerné par les numéros de dressage et le type des acrobaties (contorsionnisme, équilibrisme, trapèze volant…).
Par ailleurs, chaque numéro peut nécessiter un certain nombre d'accessoires caractérisés par un numéro de série, une désignation, une couleur et un volume.
On souhaite également savoir, individuellement, quels artistes utilisent quels accessoires.
Enfin, les accessoires sont rangés après chaque spectacle dans des camions caractérisés par leur numéro d'immatriculation, leur marque, leur modèle et leur capacité (en volume). Selon la taille du camion, une équipe plus ou moins nombreuse de chauffeurs lui est assigné (d'un à trois chauffeurs).

13) L'institut de formation – diagramme de classes

Il s'agit d'établir le schéma des données pour la gestion de formations d'un institut privé. Un cours est caractérisé par un numéro de cours (NOCOURS), un libellé (LIBELLE), une durée en heures (DUREE) et un type (TYPE). Un cours peut faire l'objet dans l'année de plusieurs sessions identiques. Une session est caractérisée par un numéro (NOSES), une date de début (DATE) et un prix (PRIX). Une session est le plus souvent assurée par plusieurs animateurs et est placée sous la responsabilité d'un animateur principal. Un animateur peut intervenir dans plusieurs sessions au cours de l'année. On désire mémoriser le nombre d'heures (NBH) effectué par un animateur pour chaque session.
Un animateur est caractérisé par un numéro (NOANI), un nom (NOMA) et une adresse (ADRA).
Chaque session est suivie par un certain nombre de participants. Un participant est une personne indépendante ou un employé d'une entreprise cliente. Un participant est caractérisé par un numéro (NO-PAR), un nom (NOMP) et une adresse (ADRP). Dans le cas d'un employé, on enregistre le nom (NO-MEN) et l'adresse de l'entreprise (ADREN). On désire pouvoir gérer d'une manière séparée (pour la facturation notamment) les personnes indépendantes d'une part, et les employés d'autre part.

14) Gestion de parc informatique – diagramme de classes

Une entreprise souhaite informatiser la gestion de son parc informatique (ordinateurs, imprimantes, etc.) pour en optimiser la maintenance.
Proposer un schéma de clases UML modélisant les spécifications ci-dessous (classes, associations entre classes, cardinalités des associations, attributs des classes).
Un ordinateur est caractérisé par son numéro d'inventaire, son adresse réseau (adresse IP), son modèle, la date de son acquisition, la date de la prochaine maintenance planifiée et le système d'exploitation installé.
Sur chaque ordinateur est installé un ensemble de logiciels caractérisés par un numéro de licence, un nom et une version.
Grâce à un système de mots de passe, chaque ordinateur peut être utilisé par plusieurs employés mais, pour des raisons de sécurité des données, un employé n'a le droit d'utiliser qu'un seul ordinateur. Un employé est caractérisé par son nom, son prénom et sa fonction dans l'entreprise.
Les ordinateurs sont reliés à un certain nombre de périphériques en réseau (imprimantes, scanners, etc.). Chaque périphérique est caractérisé par un numéro d'inventaire, son adresse IP, son type, son modèle, sa date d'acquisition et la date de la prochaine maintenance planifiée. Les périphériques pouvant servir à plusieurs ordinateurs simultanément, un indice de priorité est affecté à chaque ordinateur pour chaque périphérique auquel il est connecté.
Chaque ordinateur et chaque périphérique sont localisés dans un bureau donné. Les bureaux sont caractérisés par un numéro de bureau et le numéro du bâtiment dans lequel ils se trouvent. Un numéro de bureau est unique dans un bâtiment donné.

15) Cas d'utilisation, diagramme de classes, diagramme de séquence

---

Une entreprise souhaite modéliser avec UML le processus de formation de ses employés afin d'informatiser certaines tâches.
Le processus de formation est initialisé quand le responsable de formation reçoit une demande de formation d'un employé. Cet employé peut éventuellement consulter le catalogue des formations offertes par les organismes agréés par l'entreprise. Cette demande est instruite par le responsable qui transmet son accord ou son refus à l'employé. En cas d'accord, le responsable cherche la formation adéquate dans le catalogue des formations agrées qu'il tient à jour. Il informe l'employé du contenu de la formation et lui soumet la liste des prochaines sessions prévues. Lorsque l'employé a fait son choix il inscrit l'employé à la session retenue auprès de l'organisme de formation concerné. En cas d'empêchement l'employé doit avertir au plus vite le responsable formation pour que celui-ci demande l'annulation de l'inscription.
A la fin de la formation l'employé transmet une appréciation sur le stage suivi et un document attestant sa présence.
Le responsable formation contrôle la facture envoyée par l'organisme de formation.

a) Dessiner le diagramme des cas d'utilisation,

b) Dessiner le schéma des classes de cette application, incluant toutes les classes que l'on peut déduire de l'énoncé, ainsi que les associations entre classes avec leurs cardinalités.

c) Dessiner le diagramme de séquences associé à la demande initiale de l'employé décrite dans le deuxième paragraphe de l'énoncé ; assurer la cohérence avec votre réponse à la question précédente.

16) Cette étude de cas concerne un système simplifié de réservation de vols pour une agence de voyages. Les interviews des experts métier auxquelles on a procédé ont permis de résumer leur connaissance du domaine sous la forme des phrases suivantes :

1. Des compagnies aériennes proposent différents vols.
2. Un vol est ouvert à la réservation et refermé sur ordre de la compagnie.
3. Un client peut réserver un ou plusieurs vols, pour des passagers différents.
4. Une réservation concerne un seul vol et un seul passager.
5. Une réservation peut être annulée ou confirmée.
6. Un vol a un aéroport de départ et un aéroport d'arrivée.
7. Un vol a un jour et une heure de départ, et un jour et une heure d'arrivée.
8. Un vol peut comporter des escales dans des aéroports.
9. Une escale a une heure d'arrivée et une heure de départ.
10. Chaque aéroport dessert une ou plusieurs villes.

Déterminer le diagramme des classes modélisant le problème

17) Cet exercice concerne un système simplifié de caisse enregistreuse de supermarché. Le déroulement normal d'utilisation de la caisse est le suivant :

• Un client arrive à la caisse avec des articles à payer.
• Le caissier enregistre le numéro d'identification (CPU) de chaque article, ainsi que la quantité si elle est supérieure à un.
• La caisse affiche le prix de chaque article et son libellé.
• Lorsque tous les achats sont enregistrés, le caissier signale la fin de la vente.
• La caisse affiche le total des achats.
• Le client choisit son mode de paiement :
– numéraire : le caissier encaisse l'argent reçu, la caisse indique la monnaie à rendre au client ;
– chèque : le caissier vérifie la solvabilité du client en transmettant une requête à un centre d'autorisation *via* la caisse ;
– carte de crédit : un terminal bancaire fait partie de la caisse. Il transmet une demande d'autorisation à un centre d'autorisation en fonction du type de la carte.
• La caisse enregistre la vente et imprime un ticket.

• Le caissier donne le ticket de caisse au client.

Après la saisie des articles, le client peut présenter au caissier des coupons de réduction pour certains articles. Lorsque le paiement est terminé, la caisse transmet les informations sur le nombre d'articles vendus au système de gestion de stocks. Tous les matins, le responsable du magasin initialise les caisses pour la journée.

1) Déterminer le diagramme de cas d'utilisation.
2) Déterminer le diagramme de séquences.
3) Déterminer le diagramme d'états.

18) Affrètement d'avions

Un aéroport toulousain désire gérer les compagnies, leurs avions et les vols affrétés. Une compagnie est caractérisée par un code et un nom (comp, nomComp). Chaque avion est désigné par une immatriculation (immatriculation), un type (typeAvion), une capacité (capacite). Un avion est la propriété d'une compagnie. Un avion peut être affrété par une compagnie à différentes dates (dateVol), même plusieurs fois par jour par différentes compagnies. Pour chaque affrètement il faudra stocker le nombre de passagers transportés (nbPax) et le coût du vol pour la compagnie (cout). On ne pose pas de contrainte, donc, chaque compagnie peut affréter n'importe quel avion à n'importe quel moment. On suppose toutefois qu'une compagnie ne peut pas affréter le même avion plusieurs fois dans la même journée.

L'aéroport décide de stocker les caractéristiques de chaque type d'avion : le code de la désignation commerciale, le nombre maximum de passagers (npMax) et la désignation commerciale (nomAvion).

Exemple : l'A320 peut transporter au maximum 180 passagers et se dénomme « Airbus A320 ».

Déterminer le diagramme des classes modélisant le problème

URL utiles

1. EnterPrise Architect http://www.sparxsystems.com.au/products/ea.html
2. MagicDraw UML http://www.magicdraw.com/
3. MEGA Designer http://www.mega.com/en/product/mega_designer/
4. ModelSphere http://www.silverrun.com/modelsphere.html
5. MyEclipse http://myeclipseide.com
6. Objecteering http://www.objecteering.com/
7. Poseidon http://gentleware.com/index.php?id=30
8. PowerAMC http://www.sybase.com/products/developmentintegration/poweramc
9. Rational Rose http://www-306.ibm.com/software/awdtools/developer/datamodeler/
10. Together http://www.borland.com
11. Visio http://www.microsoft.com/france/office/visio
12. Visual Paradigm http://www.visual-paradigm.com/product/vpuml/productinfovpumlse.jsp
13. Visual UML http://www.visualuml.com/Products.htm
14. Win'Design http://www.win-design.com/fr/

Table des Matières