

Architecture des Bases de Données



Samuel "Big Boss" MONROE

22 décembre 2015

Table des matières

1	Avant-Propos	3
2	Démarche d'Analyse et de Modélisation d'un Système d'Information	4
1	Découpe des étapes	4
3	Database	5
1	Schéma Entité-Association	5
1.1	Entités	5
1.2	Associations	5
1.3	Identifiants	8
1.4	Cas concret	9
2	Schéma Relationnel	9
2.1	Transformation d'attributs composés	10
2.2	Transformation d'associations	11
3	DDL	12
4	Application	14
1	Use Case Model	14
1.1	Concepts basique du Use-Case Diagram : Acteur	14
1.2	USe-Case Diagram : Use-Case	14
1.3	Relation entre l'acteur et le use-case	14
1.4	Use-Case description	15
1.5	Autres types de relations	15
2	Diagramme d'activité	16
2.1	Activités	16
2.2	Etats	16
2.3	Transitions	16
2.4	Points de décisions	16
2.5	Exemple de diagramme d'activité complet	16
3	Diagramme de classe	17
3.1	Généralisation	17
3.2	Classification Dynamique	18
3.3	Relation d'association	18
3.4	Agrégation	18
3.5	Composition	18
3.6	Business Rules	18
4	Diagramme d'interaction	19
4.1	Diagrammes de Séquences	19
4.2	Contrôle de flux	19
4.3	Comment produire un diagramme de séquence	20
4.4	Relations avec les autres diagrammes	20
5	Schéma Récapitulatif	21

Chapitre 1

Avant-Propos

Voici la synthèse pure ultime qui va fournir une vision pure et complète du cours de Mr. Faulkner afin que tu ne te fasses pas déchirer la raie à l'examen, sauf si tu n'as rien fait pour les projets mon asti.

Chapitre 2

Démarche d'Analyse et de Modélisation d'un Système d'Information

L'ensemble du cours d'Architecture des Bases de données avait pour but de faire progresser l'étudiant dans cette démarche d'analyse et de modélisation.

Cette synthèse va donc suivre cette démarche, en revenant sur la découpe et le cheminement du processus, nous verrons ensuite individuellement chacune des étapes qui a été accomplie, et nous terminerons par une analyse supplémentaire de ces étapes afin de lier celles-ci entre elles et parfaire la vision globale de la démarche.

1 Découpe des étapes

Pour l'instant, nous allons décomposer ce travail en trois grosses étapes, qui seront également les trois chapitres qui vont suivre :

1. **Database** : Cette étape reprend principalement la réalisation du **schéma entité-association**, du **schéma relationnel** et du **Data-Definition Language**.
2. **Application** : Dans cette étape, le but à atteindre est l'obtention d'une base prête l'emploi pour implémentation du produit, mais aussi d'autres que nous détaillerons plus tard. Les étapes de la démarche sont ici la réalisation de **Use-Case Models**, de **Scénarii** qui vont appuyer ces use-cases, ensuite viennent les **diagrammes d'activités** qui vont terminer la base nécessaire à la création d'un **diagramme de classe**, enfin le **diagramme de séquence** viendra corriger et compléter ce diagramme de classe.
3. **Design - Visualisation** : A partir du diagramme de classe, ce sont les **objets** qui doivent être créés. Il y a enfin la réalisation de **Mockups** afin de terminer cette démarche et préparer l'implémentation réelle.

Chapitre 3

Database

Cette étape du travail est effectuée à partir d'une situation réelle décrite, et pour laquelle le but final est de fournir un **Système d'Information** qui répondra aux besoins identifiés par la situation.

Le but de la partie Database est de fournir une structure de données exploitable pour la suite de la démarche d'Analyse et de Modélisation.

On peut découper cette démarche Database de la sorte :

1. A partir de l'**énoncé**, on effectue une **analyse conceptuelle** afin de créer un **schéma conceptuel**.
2. A partir de ce même **schéma conceptuel**, on produit un **schéma de base de données**, qui fournira ensuite le SQL.

1 Schéma Entité-Association

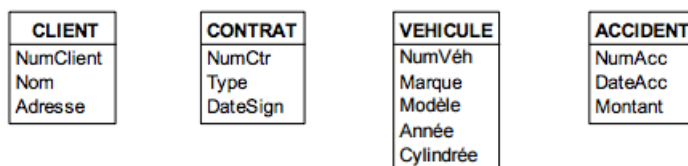
En tout premier lieu, il convient d'Analyser le **Case-Study** afin d'en repérer toutes les entités intervenantes et les associations entre celles-ci.

Ce modèle fournit un monde formé d'entités, ayant des propriétés (**attributs**) et en association les unes avec les autres. Ce **monde** est notre cas décrit, appelé aussi **le domaine d'application**.

1.1 Entités

Les entités représentent les objets du domaine d'application, qui possèdent eux-mêmes des propriétés ou attributs qui le définissent.

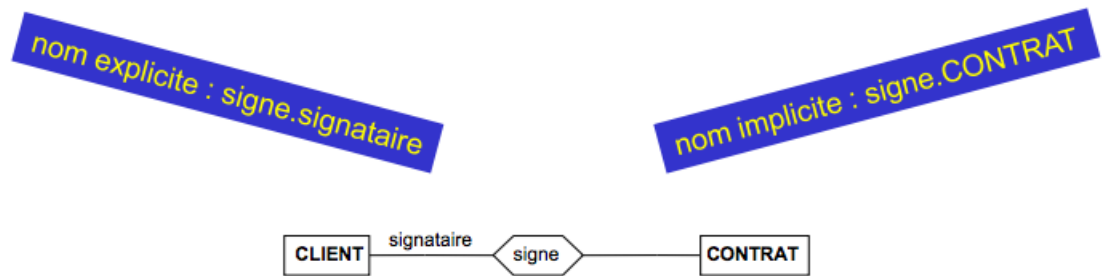
Ces entités sont représentées de la manière suivante, un cadre portant le nom de l'entité et comportant ses attributs :



1.2 Associations

Ces entités sont liées entre elles par des associations.

Ces associations portent un nom représentant le type, de préférence un verbe bien spécifique (pas être ou avoir), et les entités peuvent posséder un nom de **rôle** implicite ou explicite sur cette association :



Ces associations ont également une **cardinalité** qui complète leur type, elle définit sur quelle base numérique va s'effectuer l'association, c'est-à-dire combien d'entités A sont en association avec l'entité B.

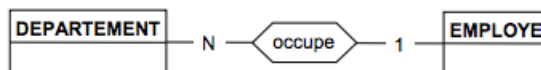
Selon les standards, cette cardinalité se lit en commençant par l'élément le plus éloigné.

On retrouve plusieurs types d'associations :

Un-à-plusieurs

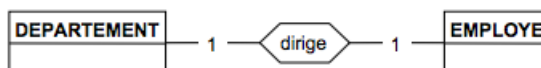
Aussi appelée **1 :N**, elle se lit ici : "Un département emploie N employés"

Ou bien de point de vue d'un employé : "N Employés sont occupés par 1 (seul) département"



Un-à-Un

Ou **1 :1**, on lira ici : "Un département dirige un employé"



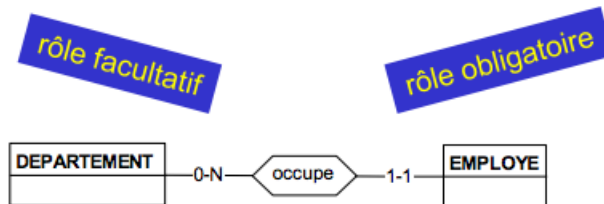
N à N

Ou **N :N**, on lira ici : "Plusieurs usines fabriquent plusieurs produits."



Rôles obligatoires ou facultatifs sur la cardinalité

Une cardinalité plus détaillée peut informer du caractère facultatif d'un rôle tel qu'on peut le voir ici :



Contrainte de cardinalité

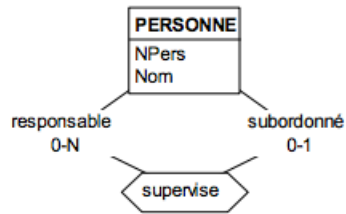
- 1-1
- 0-1
- 0-N

Combinaisons admises

- [0-1] [0-N]
- [1-1] [0-N]
- [0-1] [0-1]
- [1-1] [0-1]

Associations cycliques

Dans cette association, les deux rôles sont joués par le même type d'entité. A ne pas appeler **récur-sif** :



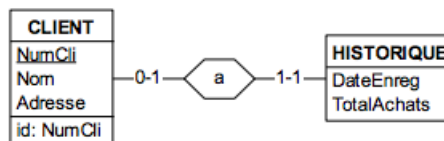
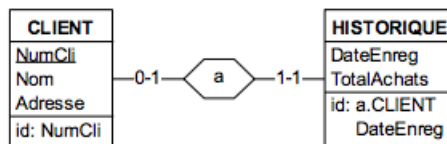
Les deux rôles sont joués par le même type d'entités

Type d'associations cyclique, ~~unaire~~, ~~réflexif~~, ~~récuratif~~

1.3 Identifiants

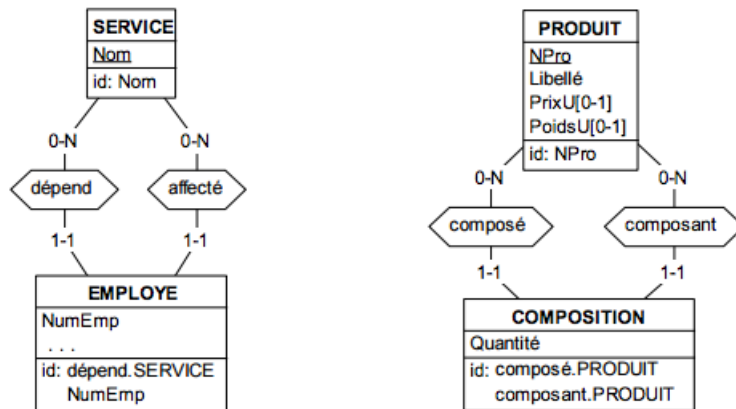
On repère les entités d'une population par un ou plusieurs attributs qui seront son ou ses identifiant, et notés de la façon suivante :

Pour les relations 1 à 1, l'identifiant d'une entité peut-être implicite de la manière suivante :



a.CLIENT est un identifiant implicite de HISTORIQUE

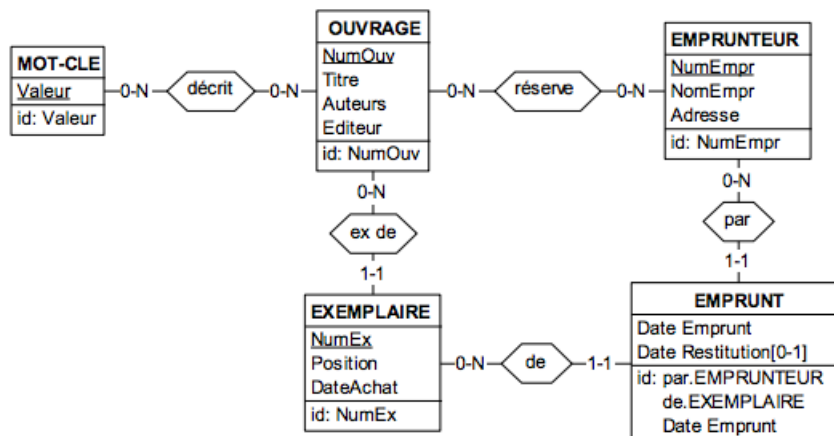
Un identifiant peut aussi être hybride de part une relation multiple :



Un composant *type d'entités* d'un identifiant hybride est désigné par son rôle

1.4 Cas concret

Voici un exemple concret d'un petit schéma E-A d'une gestion de bibliothèque :



2 Schéma Relationnel

Cette étape du travail va essentiellement consister à reprendre le schéma **Entité-Association** et à transformer nos associations pour produire un schéma **relationnel**.

Comme nous allons le voir juste après, nos associations vont se transformer en identifiants supplémentaires qui vont exprimer ces associations d'une manière exploitable pour produire le DDL(SQL).

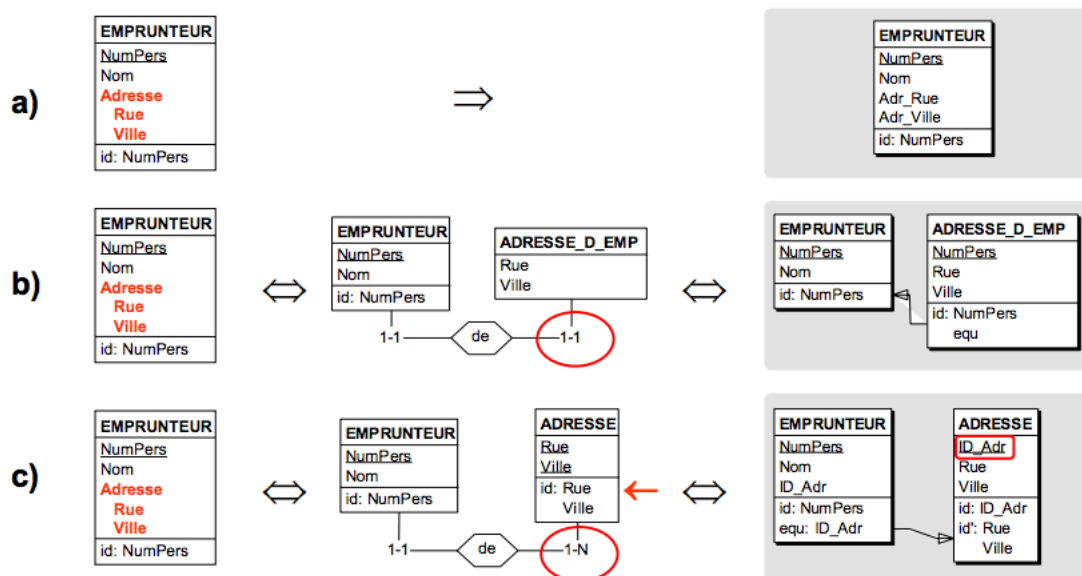
2.1 Transformation d'attributs composés

La première étape consiste en la transformation des attributs composés, telle l'adresse contenue dans une autre entité.

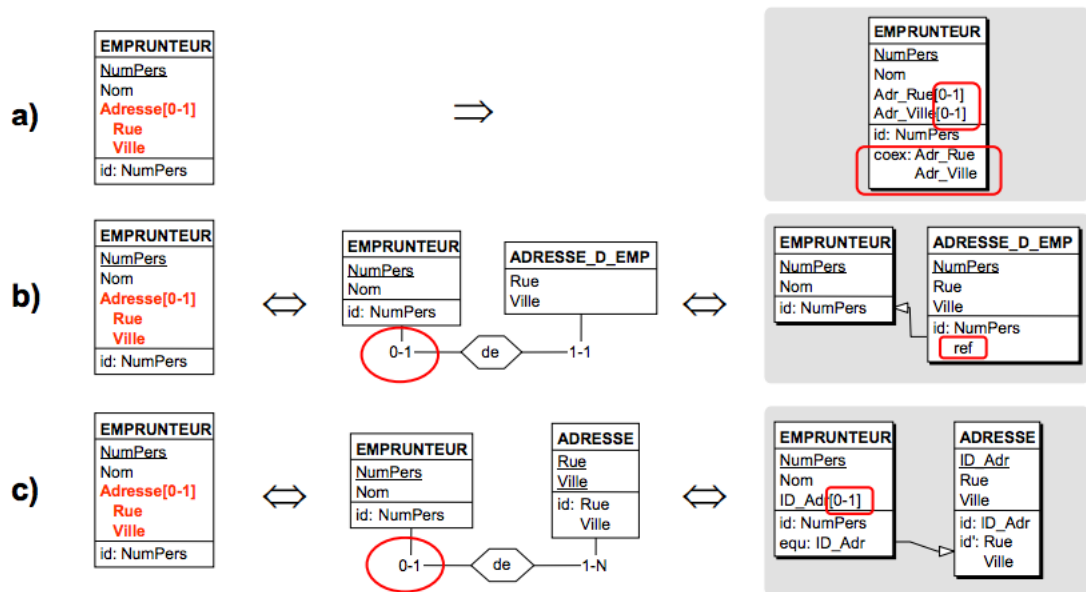
Ceci peut-être accompli par :

- Désagrégation, on remplace l'attribut par ses composants dans la même entité
- Transformation en types d'entités instanciés, dans laquelle la nouvelle entité adresse est directement liée à l'autre entité en possédant le même identifiant. Ceci est le cas pour les relations 1 :1
- Transformation en types d'entités par valeurs, la nouvelle entité est "indépendante", possède son propre identifiant, c'est dans l'autre entité que l'on va placer un champ destiné à identifier à quelle(s) adresse(s) elle est reliée.

Voici un exemple pour des attributs composés obligatoires :



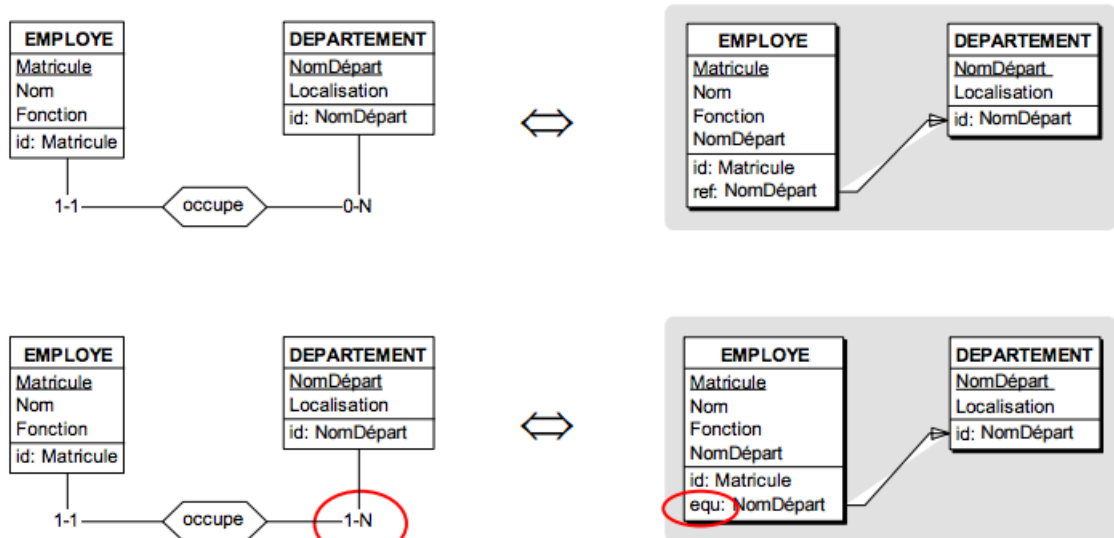
Et pour des facultatifs :



2.2 Transformation d'associations

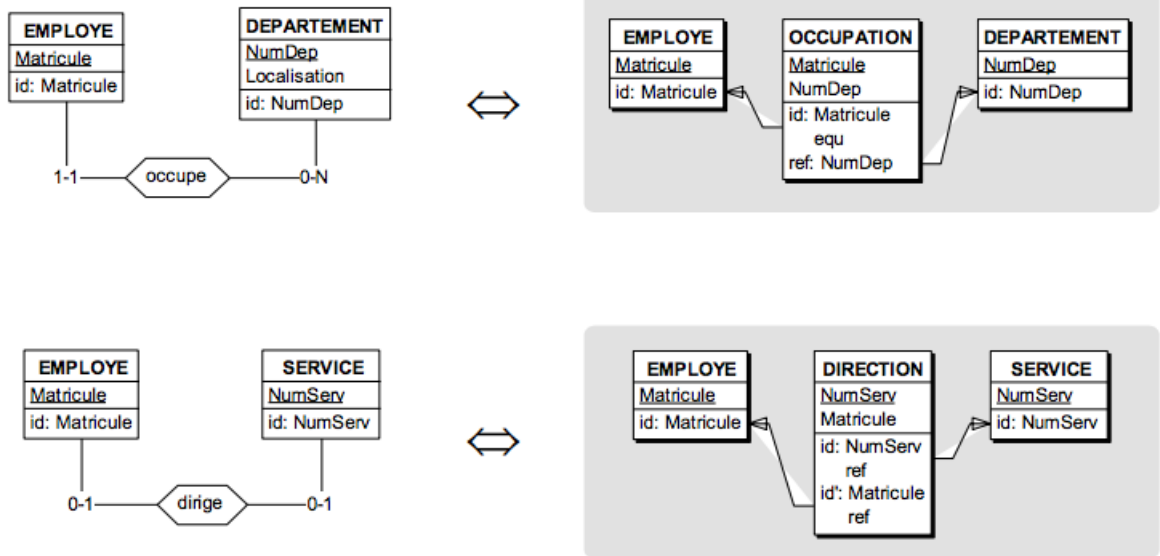
Il y a lieu à présent de transformer les associations en identifiants supplémentaires.

Basiquement cette transformation est la suivante :

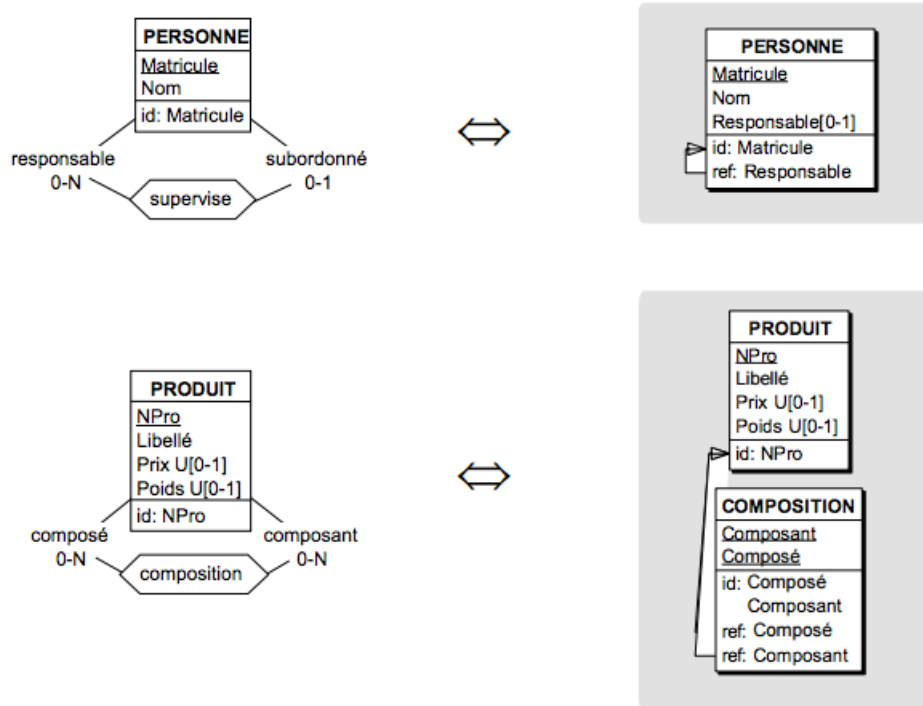


Le caractère facultatif d'une relation 0 :1 est exprimé par un **ref** dans l'id, tandis que l'obligation s'exprime par **eq**.

Certaines associations peuvent mener à la création de nouvelles tables, typiquement pour les relations 0 :N ou 1 :N, mais pas seulement :



Pour les relations cycliques, l'association peut également se transformer en nouvelle table pour une association 0 :N, ou en références pour une relation 0 :1 :



3 DDL

Maintenant que l'on possède le schéma relationnel, il s'agira uniquement de traduire ces boxes en tables via le langage SQL, typiquement de la manière suivante :

```
create table Client (  
  nom varchar(50) not null,  
  prenom varchar(50) not null,  
  idClient numeric(4) not null,  
  idCoord numeric(4) not null,  
  constraint ID_Client_ID primary key (idClient)),  
  constraint EQU_Clien_Coord_FK foreign key (idCoord) references Coordonnees;
```

Chapitre 4

Application

Ce chapitre va s'attarder sur la partie liée à l'application en elle-même, maintenant que nous possédons la partie base de données.

Nous allons d'abord voir le **Use Case Model**, ensuite les **Scénarios** duquel nous tirerons les **Activity Diagrams**. De là nous tirerons un **Diagramme de Classe** qui sera étoffé par la réalisation finale d'un **Diagramme d'Interaction**.

1 Use Case Model

Le Use-Case model est un modèle décrivant le comportement de quelques entités telles que le système ou sous-systèmes, son comportement est la façon dont il agit et réagit. C'est aussi un modèle qui décrit les interactions du système avec son environnement.

Ce Use-Case va permettre d'obtenir une communication facilitée entre l'expert et l'utilisateur, de permettre d'identifier les entités intervenantes et de pouvoir vérifier la validité du tout.

1.1 Concepts basique du Use-Case Diagram : Acteur

L'acteur représente un utilisateur agissant avec le système selon certains rôles, il peut être un humain, machine ou un autre système. Il peut échanger de l'information avec le système, être seulement donateur d'information, ou receveur passif d'information. Les acteurs sont externes.

1.2 User-Case Diagram : Use-Case

Le Use-Case en lui-même décrit une séquence d'actions qu'un système effectue afin d'obtenir un résultat observable par rapport à un acteur. Cette séquence d'action peut être décrite informellement dans du texte ou dans la spécification du comportement représentée par un lien vers un diagramme d'interaction.

Ces Use-Case sont représentés dans le diagramme par une bulle ovale portant le nom associé.

Les **Scénarios** sont utilisés pour référencer le déroulement des choses que peuvent prendre différentes instances possibles d'un même Use-Case.

1.3 Relation entre l'acteur et le use-case

Un Use-Case est initié par un acteur qui souhaite invoquer une fonctionnalité du système, l'acteur est connecté aux use-cases avec lesquels il interagit par une relation de communication.



1.4 Use-Case description

Généralement, une indication sur les scénarios couverts et les champs accédés sera ajoutée, le use-case description se présente en colonnes entre lesquelles les acteurs vont établir des actions.

4.3 Retirer le véhicule

Un client qui vient prendre son véhicule, se présente avec son bon de réservation auprès du réceptionniste du garage.

Scénarios pris en compte : a. le véhicule est indisponible, le client est invité à se représenter au service de réservation

b. le véhicule est disponible

Post-condition : le client repart avec le véhicule après avoir signé les contrats.

Client	Réceptionniste	Système
1. Le client se présente avec son bon de réservation pour retirer le véhicule.	2. Le réceptionniste vérifie de vue si le véhicule est disponible.	
3a. Le client retourne au service de réservation pour refaire une demande de véhicule (Use Case Réserver un véhicule).		
	3b. Le réceptionniste procède à l'enregistrement de la location et à l'impression en double exemplaire d'un contrat de location (Use Case Enregistrer une location).	4. Le système imprime un double exemplaire du contrat
5. Le client signe les contrats et prends un exemplaire.	6. Le réceptionniste garde un exemplaire	

1.5 Autres types de relations

On trouve :

1. Généralisation entre use-cases (inherits)
2. Généralisation entre acteurs (inherits)
3. Relation d'inclusion (uses)
4. Relation d'extension (extends)

2 Diagramme d'activité

Le diagramme d'activité est un moyen de décrire le fonctionnement d'un business, des uses-cases en eux-mêmes et entre eux.

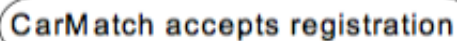
Ce diagramme consiste en activités, état et transitions entre ces activités et ces états.

- L'activité est une spécification du comportement exprimée en flux d'actions
- L'état est un point où un événement doit avoir lieu pour que l'activité continue
- La transition est un mouvement entre activités et ou états, elle est supposée représenter des actions qui surviennent rapidement et qui sont non-interruptibles.

On trouve des diagrammes d'activités qui décrivent un use-case, d'autres qui décrivent les opérations.

2.1 Activités

Les activité sont une unité de travail, ce travail peut être mentionné dans l'activité afin de bien signifier l'action entreprise.



CarMatch accepte registration

Les actions peuvent être déclanchées par entrée lorsque l'activité commence, simplement lors du temps de vie de l'activité, lors d'un événement, ou lors de la fin d'un événement.

2.2 Etats

Ils signifient une étape d'attente, on trouve également deux états spéciaux de fin (bulle noire entourée) et de début (bulle noire) de diagramme.

2.3 Transitions

Les transitions sont effectuées à la fin d'une activité, celle-ci peut être non déclanchée et survient naturellement, ou doit être déclanchée pour sortir d'un état d'attente par exemple.

On peut également trouver de multiples transition, aussi déclanchée ou non.

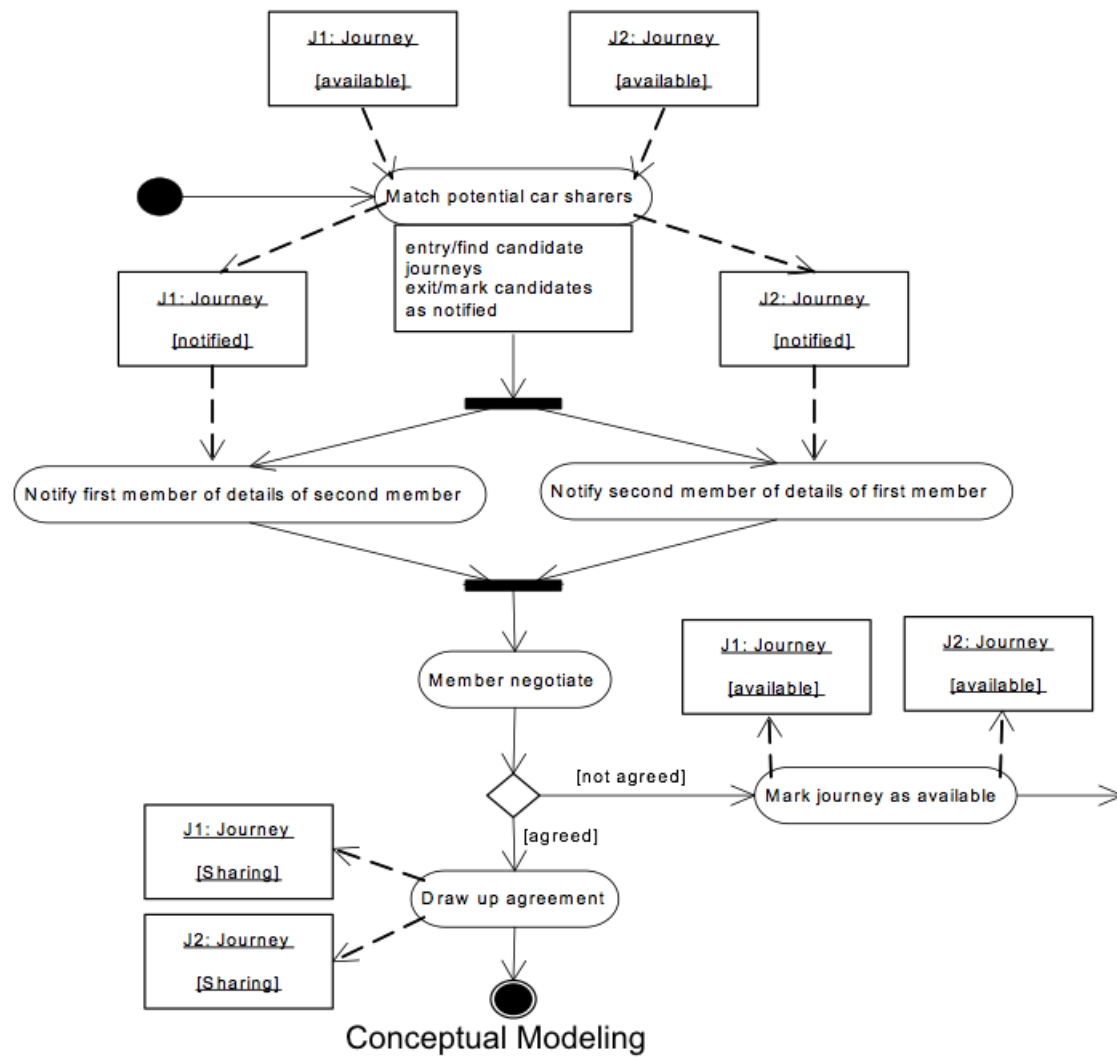
Une transition peut être **forkée**. Cela signifie que deux chemins parallèles vont être suivis en même temps, ces chemins peuvent se rejoindre sur un **join**.

2.4 Points de décisions

Point du diagramme où un choix est effectuée sur base de conditions.

2.5 Exemple de diagramme d'activité complet

Dans ce diagramme, on observe aussi l'apparition d'objets pour identifier lesquels sont effectués par un flux.



Conceptual Modeling

3 Diagramme de classe

Le diagramme de classe décrit une structure statique des classes qui agissent avec le système. Cette structure statique inclut les classes, leurs attributs et opérations ainsi que les relations entre elles.

Les classes qui sont montrées sur le diagramme dépendent de la phase de développement et du niveau de détail souhaité.

Les diagrammes de classe sont utilisés pour documenter les classes qui constituent le système ou le sous-système, pour décrire les associations, généralisations et agrégations entre ces classes.

Pour montrer les attributs et opérations de chaque classe, pour montrer la structure d'une classe au long du cycle de vie du développement.

Pour montrer les objets individuels dans une structure de classe ainsi que pour montrer les interfaces supportées par une classe donnée.

Je passerai ici les éléments déjà vu en Java concernant les classes et objets et m'attarderai sur des concepts un peu plus importants.

3.1 Généralisation

Relation d'héritage représentée par un triangle, la classe parent est la classe généralisée, les enfants sont les classes spécialisées et héritent du parent.

Les enfants peuvent redéfinir les méthodes afin de les utiliser selon la spécialisation.

3.2 Classification Dynamique

La classification réfère aux relations entre un objet et les classes dont il est une instance.

Certaines classification sont dites **mandatoires** ce qui signifie qu'une classe doit être une instance de telle classe, d'autres sont dites dynamique et peuvent cesser d'être une instance de telle classe pour devenir une autre.

3.3 Relation d'association

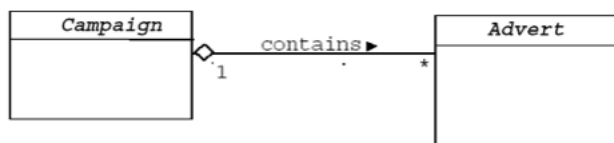
Une association est une relation structurée qui représente une relation binaire entre des objets, une association a un nom et doit être spécifiée de zéro, un ou deux rôles.

Une cardinalité peut être ajoutée à l'association (pas même sans que E-A !), et une direction peut-être apposée au nom.

3.4 Agrégation

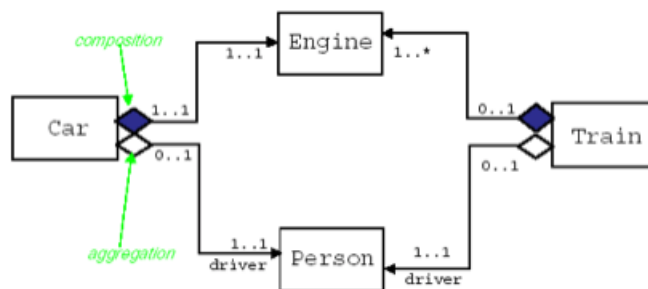
L'agrégation est une relation d'appartenance, ou un objet est un tout, et l'autre est une partie du premier.

Cette relation est marquée d'un losange :



3.5 Composition

Cas spécial de l'agrégation, une relation de composition implique une relation vie/mort entre les deux classes. Si une instance meurt, les instances liées à la composition doivent mourir elles aussi.



3.6 Business Rules

Une Business Rule est utilisée pour décrire les propriétés d'une application, par exemple le fait qu'un employé ne peut gagner plus que son manager.

Elle peut être :

- Description d'un concept ayant un rapport avec l'application
- Une contrainte d'intégrité sur les données de l'application
- Une règle dérivée, où l'information peut être dérivée d'une autre information dans le diagramme de classe

4 Diagramme d'interaction

En object-oriented les tâches du système sont effectuées par des objets interagissant ensemble. Les interactions entre ces objets sont modélisées par des diagrammes d'interaction. Une interaction entre deux objets A et B implique que A envoie un message qui requière une action que B peut effectuer.

Il y a deux types de diagrammes d'interaction :

- Les diagrammes de séquences
- Les diagrammes de collaboration

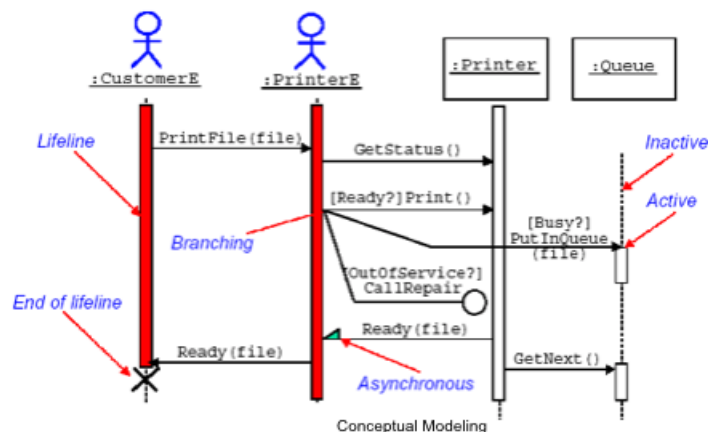
Le diagramme de classe modélise la structure statique du système tandis que le diagramme d'interaction modélise la partie dynamique.

4.1 Diagrammes de Séquences

Les diagrammes de séquences décrivent en détails comment les acteurs utilisent les use-cases. Les interactions consistent en un ou plusieurs messages échangés entre des objets externes et faisant partie du système. Les interactions peuvent être synchrones ou asynchrones.

Voici un exemple complet de diagramme de séquence :

- **Synchronous:** a message is sent by one object to another and the first object waits until the resulting action has completed
- **Asynchronous:** a message is sent by one object to another but the first object does not wait until the resulting action has completed to carry out the next action



7

4.2 Contrôle de flux

Des flèches pointillées optionnelles peuvent être ajoutées pour représenter les retours après appels de méthodes.

4.3 Comment produire un diagramme de séquence

Pour un cas particulier, commencer par identifier les objets et acteurs impliqués.

Elaborer pas à pas le diagramme de séquence.

Retourner à la définition de la classe pour repérer les attributs ou méthodes qui manqueraient.

4.4 Relations avec les autres diagrammes

Le diagramme de séquence peut être utilisé pour vérifier les use-cases ou les classes.

- Chaque opération du diagramme de séquence doit exister dans le diagramme de classe
- Le nom des messages doit être un événement ou une méthode de la classe qui reçoit ce message

Chapitre 5

Schéma Récapitulatif

