Ingénierie Logicielle

Une introduction

Rudi GIOT - giot@isib.be

Introduction

Définitions

Nous allons essayer de donner une définition la plus complète possible et qui contiendra le vocabulaire de base que nous utiliserons tout au long de ce cours.

Le génie logiciel c'est
dans des
des programmes et de la
en vue d'utiliser des systèmes informatiques pour résoudre un problème.

Une autre définition plus officielle : selon l'arrêté ministériel du 30 décembre 1983 relatif à l'enrichissement du vocabulaire de l'informatique [Journal officiel du 19 février 1984], le génie logiciel est « l'ensemble des activités de conception et de mise en œuvre des produits et des procédures tendant à rationaliser la production du logiciel et son suivi ».

Naissance

On peut considérer que le génie logiciel est né suite à une conférence promue par l'OTAN en octobre à Garmisch Partenkirchen en Allemagne[1] et qui évoquait la nécessité d'instaurer des « bonnes pratiques » en matière de réalisation des logiciels. Voir le rapport en PDF annexé à ce cours.

Nécessité

Dans notre histoire contemporaine, plusieurs exemples malheureux ont montré la nécessité d'instaurer des "bonnes pratiques" en termes de :

- tests : la sonde Mariner 1 perdue à cause d'une "faute de frappe" dans un programme Fortran
- complexité : le crash du système de communication d'AT&T en 1990[2]
- délais : OS-360 d'IBM retardé plusieurs fois et finalement livré "bugué"[3]

Vous pourrez trouver d'autres exemples malheureux en suivant ce lien :
http://www.devtopics.com/20-famous-software-disasters/

---

[1] http://homepages.cs.ncl.ac.uk/brian.randell/NATO/NATOReports

[2] http://www.phworld.org/history/attcrash.htm

[3] http://www.psexam.com/Notes-for-Computer-Science/operating-systems-history-of-operating-system-article/1960s-%E2%80%93-Disappointing-Efforts-of-IBM-to-Develop-OS/360.html

Le Software Engineering Institute (http://www.sei.cmu.edu/) a défini la notion de "maturité du logiciel" dans son "Capability Maturity Model" (voir PDF en annexe) en cinq niveaux :

1. « **Initial** » (niveau de maturité 1)
Dans ce niveau il n'y a pas d'analyse, pas de suivi, pas d'évaluation. Les travailleurs ne sont pas clairement assignés à des tâches et de ce fait ce niveau est parfois aussi appelé "chaotique".
2. « **Managed** » (niveau de maturité 2)
Une discipline est établie pour chaque projet et se matérialise essentiellement par des plans de projet (plan de développement, d'assurance qualité, de gestion de configuration…). Le chef de projet a une forte responsabilité dans le niveau 2 : il doit définir, documenter, appliquer et maintenir à jour ses plans. D'un projet à l'autre, il capitalise et améliore ses pratiques de gestion de projet et d'ingénierie.
3. « **Defined** » (niveau de maturité 3)
Ce niveau est caractérisé par une standardisation adéquate des pratiques, une capitalisation centralisée (en particulier sur les mesures réalisées dans les projets) et une maîtrise du référentiel interne (ou Système Qualité). Il existe des lignes directrices, un plan stratégique et une planification de l'amélioration de processus pour le futur, en ligne avec les objectifs d'affaire de l'organisation. Les employés sont formés et conscients de leurs responsabilités ainsi que de leurs devoirs.
4. « **Quantitatively managed** » (niveau de maturité 4)
Les projets sont pilotés sur la base d'objectifs quantitatifs de qualité produit et processus. La capacité des activités (ou sous-processus) critiques est déterminée par l'organisation, ainsi que les modèles de performance et de prévision associés. L'expression de la qualité demandée par le client est prise en compte pour quantifier les objectifs du projet et établir des plans selon la capacité des processus de l'organisation.
5. « **Optimizing** » (niveau de maturité 5)
En plus d'être de niveau de maturité 4, les processus sont optimisés.

Histoire de l'informatique

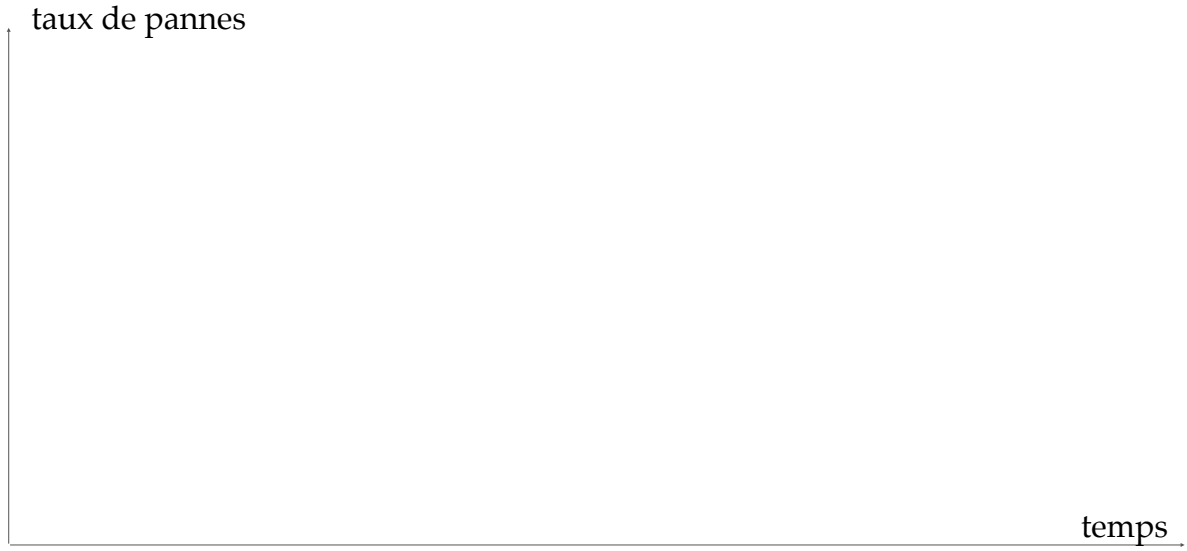
Vous trouverez plusieurs "ligne du temps" sur le site du "Musée de l'histoire de l'informatique" en Californie : http://www.computerhistory.org/timeline/

Les catégories de logiciels

Il est difficile de « classer » les logiciels en catégories. Cependant nous pouvons évoquer quelques familles de logiciels qui regroupent un ensemble de spécificités communes :

Spécificités du logiciel

Si on dessine un graphique pour représenter le taux de pannes d'un appareillage ménager ou celui d'un logiciel en fonction du temps, on obtient deux courbes légèrement différentes.



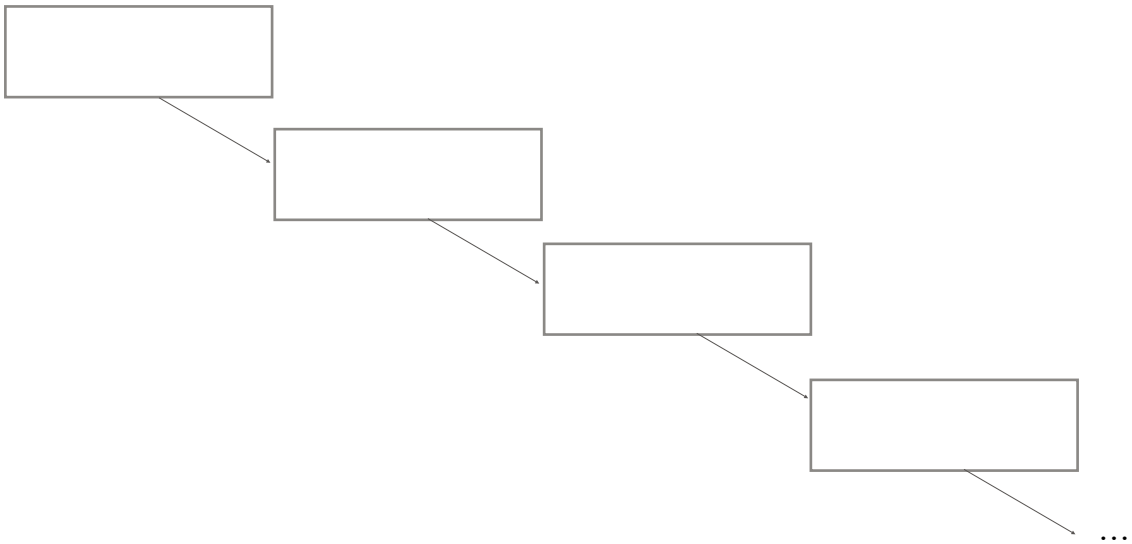
Dans le monde du logiciel la notion d' est plus critique que dans les autres domaines.

Les différentes activités

Dans le cycle de vie d'un projet logiciel nous allons trouver un ensemble d'activités. Ces activités seront plus ou moins nombreuses suivant le projet et seront de natures différentes. Nous allons essayer d'énumérer les plus fréquentes sans nécessairement les mettre dans un ordre particulier puisque c'est dans le point suivant (les différents modèles) que nous les organiserons.

Les différents modèles

Modèle en cascade



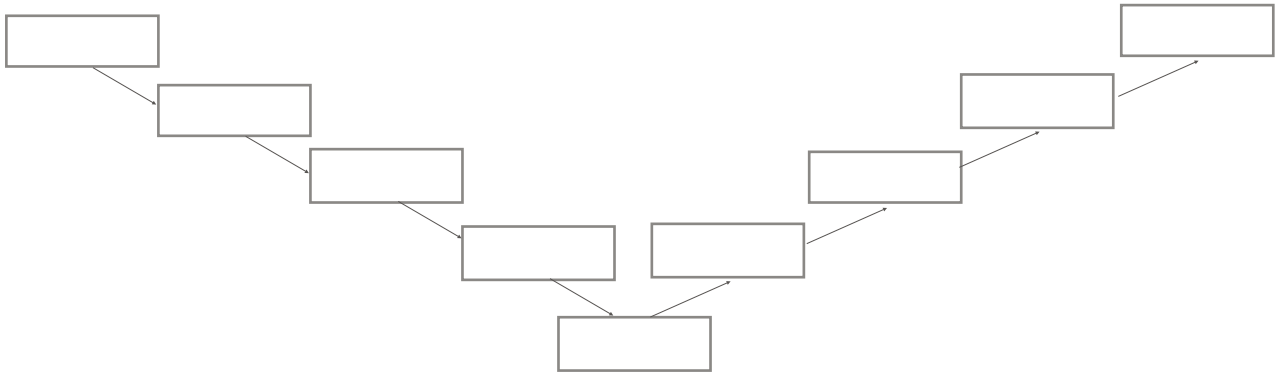
Ce modèle présente un avantage indéniable :
Mais présente malheureusement plusieurs désavantages :

On peut tenter d'améliorer le modèle en

Ce qui amène un peu de souplesse mais peut conduire à

Modèle en V

Le modèle en « V » ou cycle en « V » est une forme du modèle en cascade dans lequel on insiste explicitement sur la notion de « tests ».



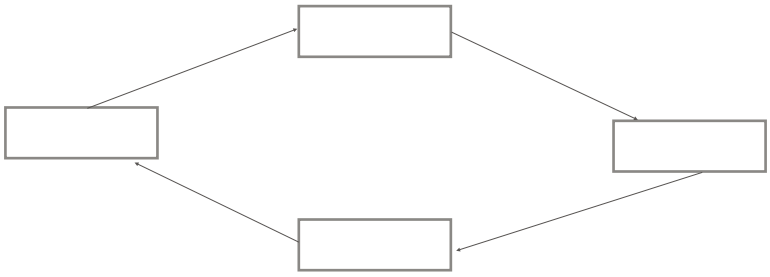
…

Ce modèle présente comme avantages :

Mais présente toujours quelques désavantages :

Modèle itératif à prototypes

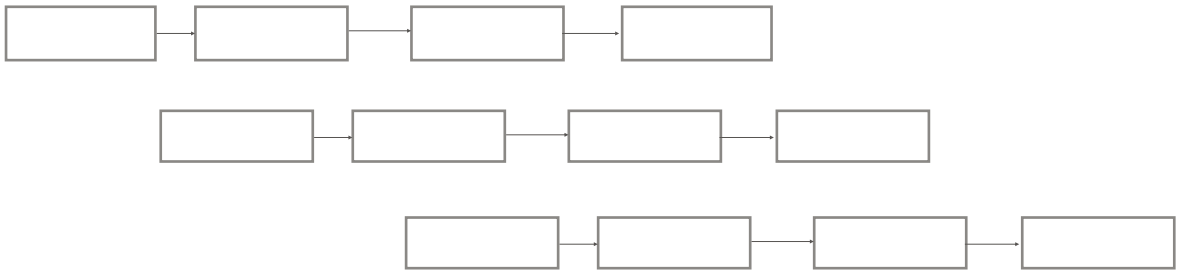
Le modèle itératif



Attention, il ne faut pas confondre protide et maquette.

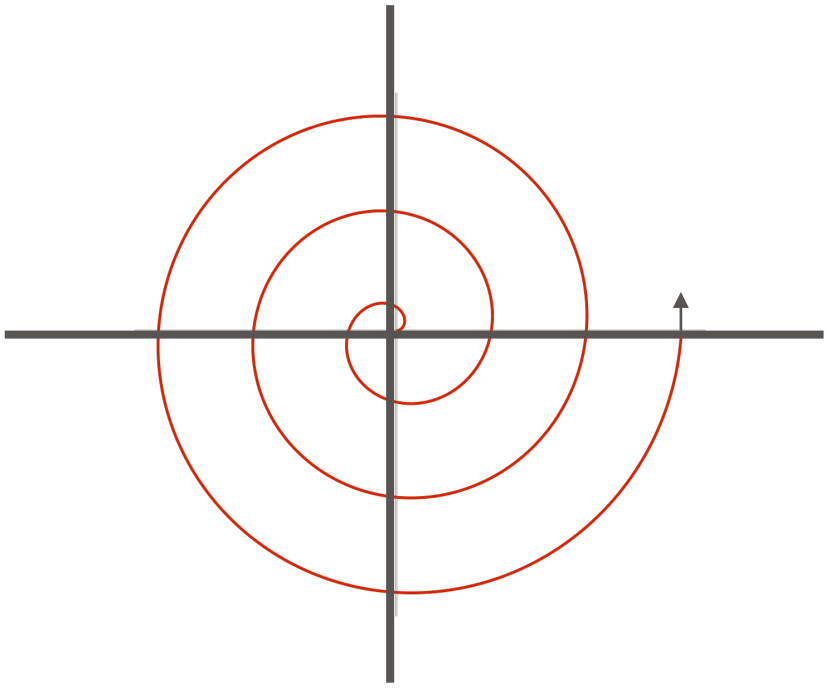
Modèle incrémental

Le modèle incrémental …



Modèle à spirale

Le modèle à spirale est un modèle très polyvalent. Il permet de découper la spirale en un nombre de morceaux laissé au choix du chef de projet. Chaque tâche assignée à chacun des morceaux peut être également définie par le chef de projet. Chaque itération doit donner lieu à une production : une documentation, un prototype, …



Remarques

1. Le RAD

Il faut rester très prudent avec l'appellation RAD (Rapid Application Development).

Par exemple sur Wikipedia vous pourriez lire : " La méthode de développement rapide d'applications, dite méthode RAD est la première méthode de développement de logiciels où le cycle de développement est en rupture fondamentale par rapport à celui des méthodes antérieures dites « en cascade ». Ce nouveau cycle qualifié d'itératif, d'incrémental et d'adaptatif, se retrouvera dans toutes les méthodes dites « agiles » publiées par la suite."

Pour d'autres, " Le RAD (Rapid Application Development) est une technologie permettant de créer en quelques clics une application complète (ou un site complet). Le RAD se base sur des modèles de programmation pré-établis (appelés Patterns), permettant de générer l'application voulue. Il est également possible de créer entièrement ses propres "patterns" pour générer automatiquement une application personnalisée : vous décidez du code généré ainsi que du positionnement des champs. " (Publicité pour le logiciel WinDev)

Pour d'autres encore : "RAD model is Rapid Application Development model. It is a type of incremental model. In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements."[4]

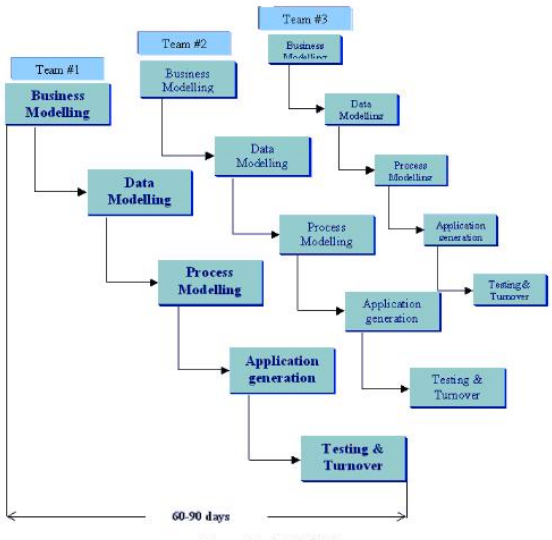


Figure 1.5 – RAD Model

Vous devrez donc toujours demander à un interlocuteur de quoi il parle quand il évoque le "Rapid Application Development".

[4] http://istqbexamcertification.com/what-is-rad-model-advantages-disadvantages-and-when-to-use-it/

2. Les “méthodes” formelles

Le but des “méthodes” formelle est de construire un modèle mathématique des programmes à réaliser.

```
MACHINE
        swap
VARIABLES
        xx, yy
INVARIANT
        xx : NAT & yy : NAT
INITIALISATION
        xx :: NAT ||
        yy :: NAT
OPERATIONS
        echange =
        BEGIN
                xx := yy
        ||      yy := xx
        END
END
```

Avantage :

- sûreté de fonctionnement des logiciels critiques: militaires, sécurité ferroviaire, …

Désavantages :

- certains problèmes se prêtent mal à une modélisation formelle
- ils ne peuvent généralement pas servir de moyen de communication avec le client
- gros investissements en formation, en temps d’analyse, …

3. Les languages et leur “génération”

La première génération de langage correspond au plus bas niveau de programmation : le code machine. Ce sont des instructions binaires exécutées par le processeur, ce code est évidemment dépendants du processeur pour lequel il est destiné.

La deuxième génération correspond aux langages assembleurs. Le codes est écrit sous forme d’instructions simples (assignation de registres, transferts de données vers la mémoire, opérations mathématiques simples, boucles, …). Ils sont dépendants du processeur pour lesquels ils sont destinés.

```
Mov Ax, 65535
Mov Cl, 01101b
Mov Dh, 0FAh
```

Les langages de 3e génération permettent d'écrire des programmes d'une manière plus naturelle dans une forme proche l’anglais sont indépendant du processeur. Ces langages nécessitent donc un compilateur ou un interpréteur. Les plus connus sont : Java, C/C++, Fortran, Pascal, Cobol, PHP, …

Le terme de « langage de 4e génération » formulé par James Martin en 1982 désigne des langages de programmation de haut niveau. Ils sont plus proches encore des langues naturels, ils sont généralement moins optimisés. On considère généralement le SQL comme étant de 4eme génération.

La cinquième génération de langages est difficile à définir tant les interprétations sont différentes. Ils s’agit souvent d’une appellation marketing pour vendre de nouveaux outils de reporting ou de création graphique d’applications.

Les outils

Les outils du génie logiciel sont nombreux. Nous allons en faire une liste non-exhaustive mais qui sera la plus complète possible. Nous expliquerons dans le chapitre suivant (Les méthodes) comment utiliser ces outils.

Les énoncés informels

Il s'agit de textes rédigés en langage naturel qui présente souvent des ambiguité, des incohérences et de la non complétude. Ca n'est pas une raison pour s'en passer, en effet, ils compléteront généralement les outils formalisés.

Les DFD

Dictionnaire des données

Un dictionnaire des données est une collection de métadonnées ou de données de référence nécessaire à la conception d'une base de données relationnelle. Il revêt une importance stratégique particulière, car il est le vocabulaire commun de l'organisation. Il décrit des données aussi importantes que les clients, les nomenclatures de produits et de services, les annuaires, etc. C'est donc le référentiel principal de l'entreprise, sur lequel s'appuient les décisions de celle-ci. Il est souvent représenté par un tableau à quatre colonnes contenant le nom, le code et le type de donnée ainsi que des commentaires.

Un dictionnaire des données doit respecter les contraintes suivantes.

Tous les noms doivent être monovalués et non décomposables.
Il ne doit pas y avoir d'homonymes, ni de synonymes.
Les données y sont regroupées par entité.
Les identifiants sont complètement précisés,
Les commentaires doivent être pertinents.

Les présentations formatées

Les présentations formatées de données respectent une grammaire quasi formelle pour décrire le contenu des objets (souvent des valeurs prises par les données) définis dans une analyse.

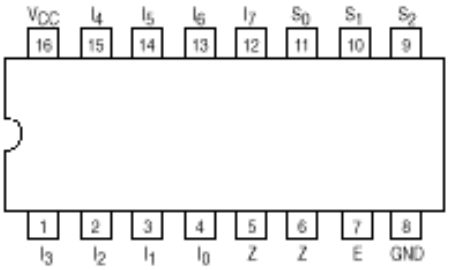
Table de décision

Une table de décision est un outil logique permettant de modéliser un nombre restraint (sinon la table devient difficile à remplir) de possibilité en entrée avec la sortie attendue. Cela sous entend qu'on doit connaitre à l'avance tous les cas possible avec leur solution.

	Behaviour / Status	Spends less than 100 dollars	Spends more than 100 dollars	Spends more than 150 dollars	Spends more than 200 dollars
1	Customer with bronze status	Pays 5 dollars for item	Pays 4 dollars for item	Pays 3 dollars for item	Pays 2 dollars for item
2	Customer with silver status	Pays 4 dollars for item	Pays 3 dollars for item	Pays 2 dollars for item	Pays 1 dollar for item
3	Customer with gold status	Pays 3 dollars for item	Pays 2 dollars for item	Pays 1 dollar for item	Item is free

On utilise beaucoup cette méthode en électronique numérique dans des tables appelées tables de vérités.

CONNECTION DIAGRAM DIP (TOP VIEW)

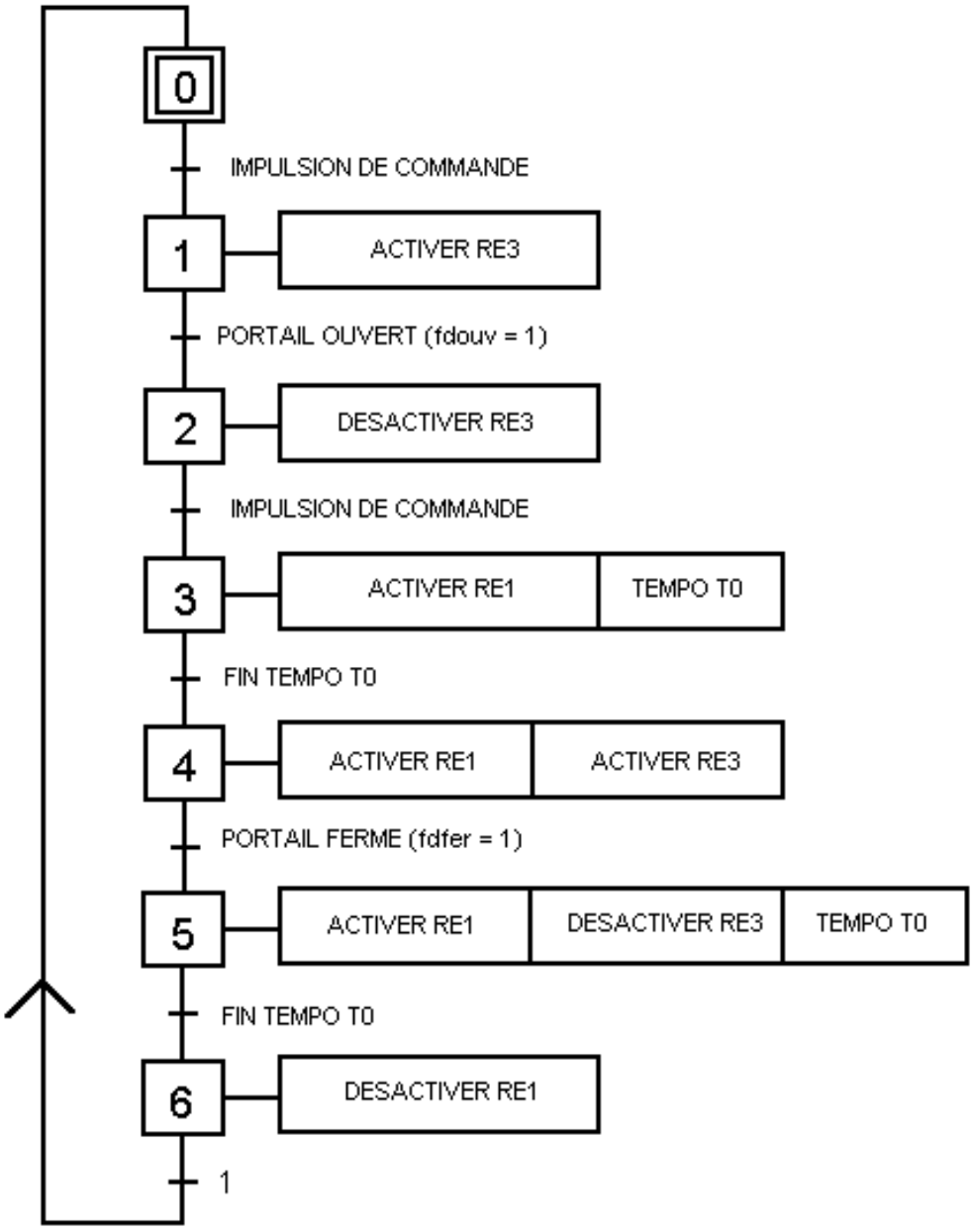


$\overline{E}$	S_2	S_1	S_0	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	$\overline{Z}$	Z
H	X	X	X	X	X	X	X	X	X	X	X	H	L
L	L	L	L	L	X	X	X	X	X	X	X	H	L
L	L	L	L	H	X	X	X	X	X	X	X	L	H
L	L	L	H	X	L	X	X	X	X	X	X	H	L
L	L	L	H	X	H	X	X	X	X	X	X	L	H
L	L	H	L	X	X	L	X	X	X	X	X	H	L
L	L	H	L	X	X	H	X	X	X	X	X	L	H
L	L	H	H	X	X	X	L	X	X	X	X	H	L
L	L	H	H	X	X	X	H	X	X	X	X	L	H
L	H	L	L	X	X	X	X	L	X	X	X	H	L
L	H	L	L	X	X	X	X	H	X	X	X	L	H
L	H	L	H	X	X	X	X	X	L	X	X	H	L
L	H	L	H	X	X	X	X	X	H	X	X	L	H
L	H	H	L	X	X	X	X	X	X	L	X	H	L
L	H	H	L	X	X	X	X	X	X	H	X	L	H
L	H	H	H	X	X	X	X	X	X	X	L	H	L
L	H	H	H	X	X	X	X	X	X	X	H	L	H

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

Les diagrammes états/transitions

Dans ces diagrammes on retrouve aussi les réseaux de Pétri, les graphite, … Il matérialise l'incidence des événements dans un processus sur les différents états du système en indiquant les actions à effectuer. Il sont très utilisés en automatique à destination des automates programmables.

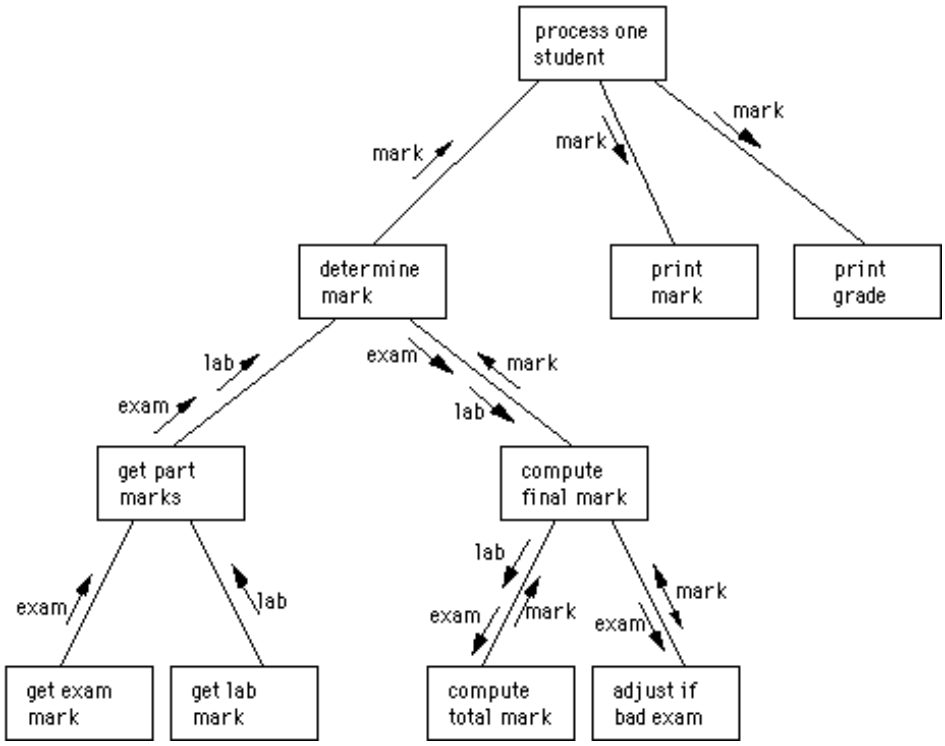


Exemple de Graphcet [5]

[5] http://www.ile-reunion.org/louispayen/cours/grafcet_ouverture_fermeture_complexe.gif

Les diagrammes de structure

Ces diagrammes sont bien adaptés pour montrer la hiérarchie des fonctions dans une arborescence de programme. Ils sont bien adaptés pour des applications programmées avec un langage procédural.



Exemple de diagramme structuré[6]

[6] http://www.ada95.ch/doc/tut1/Subprograms/structure_chart.gif

L

D

Entités / Associations

Le modèle ***Entités/Associations*** (Entity-Relationship - ER, en anglais) est un formalisme graphique défini en 1969[7] pour la modélisation des données destiné en général à la structuration des tables des bases de données relationnelles.

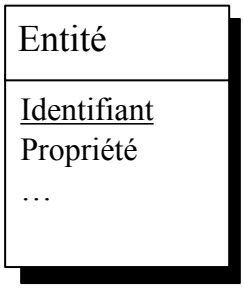
Une ***entité*** est une chose concrète ou abstraite de la réalité perçue à propos de laquelle on veut conserver des informations. Ces informations seront consignées dans des ***propriétés***.

Un ***propriété*** est une caractéristique d'une entité qui peut prendre une valeur.

Une ***occurrence d'entité*** est un individu concret issu d'une entité, les propriétés prennent alors des valeurs concrètes.

Chaque entité doit posséder un et un seul ***identifiant*** qui est formé d'une ou plusieurs propriétés de l'entité à identifier. Son rôle est de permettre de distinguer une occurrence d'une entité d'une autre occurrence de la même entité.

Au niveau du formalisme on représente l'entité nommée dans un rectangle. On y ajoute les propriétés et on souligne les propriétés qui servent d'identifiant.



[7] C.W. Bachman, "Data Structure Diagrams," DataBase: A Quarterly Newsletter of SIGBDP, vol. 1,1969.

Exercice
Modéliser le problème suivant : Une entreprise de plomberie effectue des travaux dans des immeubles dans toute la Belgique pour des montants fixés préalablement par des devis. Si le devis ne convient pas au client, il se peut que le travail ne soit jamais exécuté.

Une ***association*** (parfois aussi appelée « relation ») est une correspondance entre deux ou plusieurs entités à propos des quelles on veut conserver des informations.

Une association peut posséder une ou plusieurs propriétés.

Exercice
Modéliser le problème suivant : Un concessionnaire automobile vend des voitures neuves à des particuliers uniquement (pas à des sociétés). Il souhaite informatiser son stock de voitures et aussi avoir un fichier client pour faire de la prospection.

Les ***cardinalités*** (minimum et maximum) donnent une indication sur la participation d'une occurrence d'une entité à l'association aux quelles elles sont attachées.

Une cardinalité **minimum de 0** signifie qu'une occurrence de l'entité visée n'est pas obligée de participer à l'association.

Une cardinalité **minimum de 1** signifie qu'une occurrence de l'entité visée participe obligatoirement à l'association.

Une cardinalité **maximum de 1** signifie qu'une occurrence de l'entité visée participer au plus une fois à l'association.

Une cardinalité **maximum de type « n »** signifie qu'une occurrence de l'entité visée peut participer un nombre indéfini de fois à l'association.

Exercice
A partir du problème précédent, préciser les cardinalités sur le schéma et justifier chaque cardinalité par une phrase.

Attention aux propriétés des associations qui ont une pate d'association binaire à 1,1

Les associations ternaires

On peut avoir une association reliée à plus de deux entités. On parle alors d'une association n-aire.

Exercice

Le syndic d'un ensemble d'immeuble voudrait informatiser sa gestion des travaux. Il emploie des entreprises qui réalisent des travaux dans les différents immeubles d'appartements dont il s'occupe.

Important

Dans les associations ternaires (ou n-aires) il faut être vigilant au fait qu'à : une occurrence de l'association est associée une et une seule occurrence de chacune des entités visées par l'association et que cette association ne peut exister qu'une et une seule fois.

Les ***auto-asssociations***.

Une association peut associer une entité à elle -même. Il devient alors nécessaire en plus du nom donné à l'association de qualifier les pates de l'association.

Exercice :
Comment modéliser le fait que dans le personnel d'une entreprise certains employés sont responsables d'autres employés. Sachant qu'un employé n'a qu'au maximum un seul chef.

Les ***doubles, triples, … associations***.

On peut trouver deux ou plusieurs associations différentes qui relient deux mêmes entités. Les cardinalités peuvent varier en fonction des associations.

Exercices :
Une personne peut posséder ou habiter dans un logement.

Une personne loue des logements à d'autres personnes.

Les ***types et sous-types d'entités***.

Il est possible de spécialiser ou généraliser des entités.

Exercice

Une école engage du personnel enseignant et ouvrier. Les enseignants possèdent un statut (temporaire/nommé) et une fonction (assistant/chargé de cours/professeur). Les ouvriers .. Parfois certains étudiants sont engagés pour réaliser divers travaux et font donc partie du personnel.

Les ***restrictions et sous-types d'associations***.

Il est également possible de spécialiser des associations.

Exercice

Une entreprise engage des employés qui sont parfois des secrétaires. Les employés travaillent sur des projets alors que les secrétaires ne font que gérer des projets. Un projet est toujours géré par un et une seul secrétaire.

Les contraintes interrelations - DF et CIF.

Une ***dépendance fonctionnelle*** (DF) ou ***contrainte d'intégrité fonctionnelle*** (CIF) implique qu'à un élément de A correspond **au plus** un élément de B.

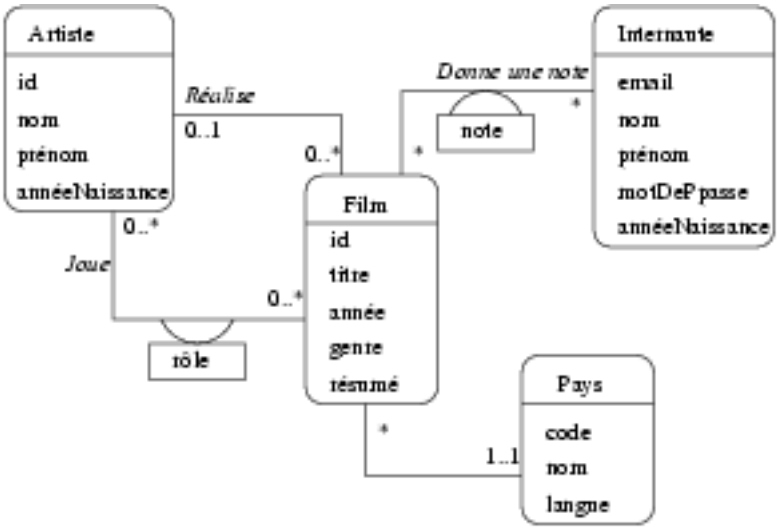
Exemple :
Une voiture est achetée par une et une seule personne. Dans une association binaire la CIF apparait donc de manière implicite quand la cardinalité maximum est de 1.

Exercice :
Reprenons l'exemple du syndic, des ses immeubles, des types de travaux et des entreprises en imposant que une entreprise doit toujours êtres spécialisée dans un type de travail. La société X ne fait que de la plomberie. La société Y ne fait que de l'entretien d'ascenseurs.

Remarque :
La décomposition d'une association n-aire avec CIF en plusieurs associations plus simples est possible si le nombre d'entités impliquées dans la CIF est strictement inférieure au nombre d'entités impliquées dans l'association.

Remarques

Attention au différentes variations de formalisme :



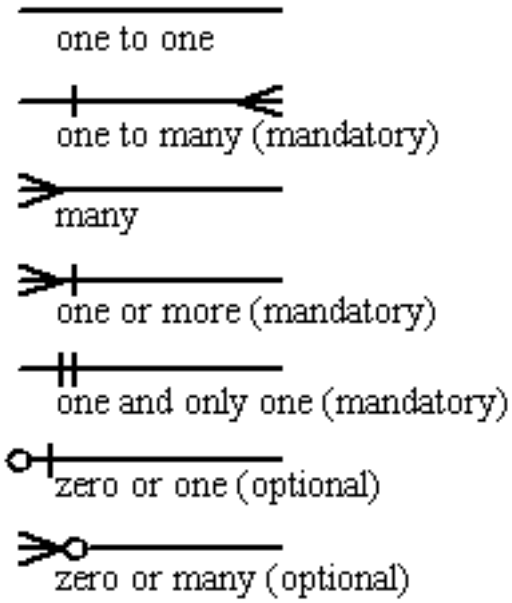
Exemple d'alternative au niveau du formalisme[8]

Exercice

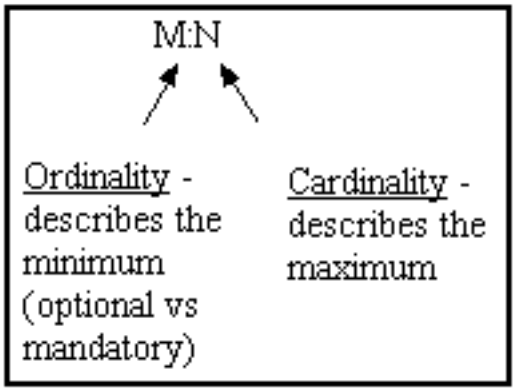
Sachant que la notation « 0..* » sur l'association « Réalise » serait équivalente à notre « 0,n » si elle avait été placée du côté Artiste, que la notation « * » est équivalente à notre « 0,n » et que la notation « 1 » est équivalente notre à « 1,1 », expliquer en quelques lignes les différentes associations et cardinalités.

[8] Source : http://perso.limsi.fr/jacquemi/COGNITIQUE02/corpus/chap-ea.html

Information Engineering style



Chen style

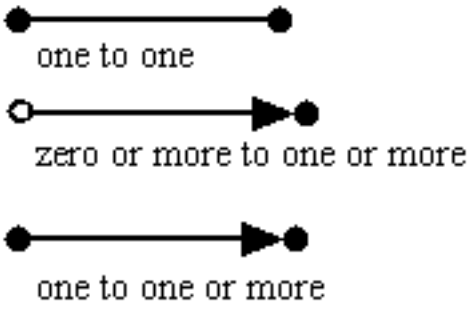


1:N (n=0,1,2,3...)
one to zero or more

M:N (m and n=0,1,2,3...)
zero or more to zero or more
(many to many)

1:1
one to one

Bachman style



Martin style

1 - one, and only one (mandatory)

* - many (zero or more - optional)

1...* - one or more (mandatory)

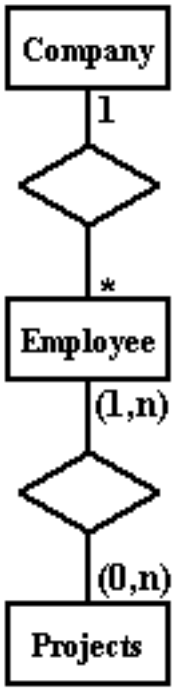
0...1 - zero or one (optional)

(0,1) - zero or one (optional)

(1,n) -one or more (mandatory)

(0,n) - zero or more (optional)

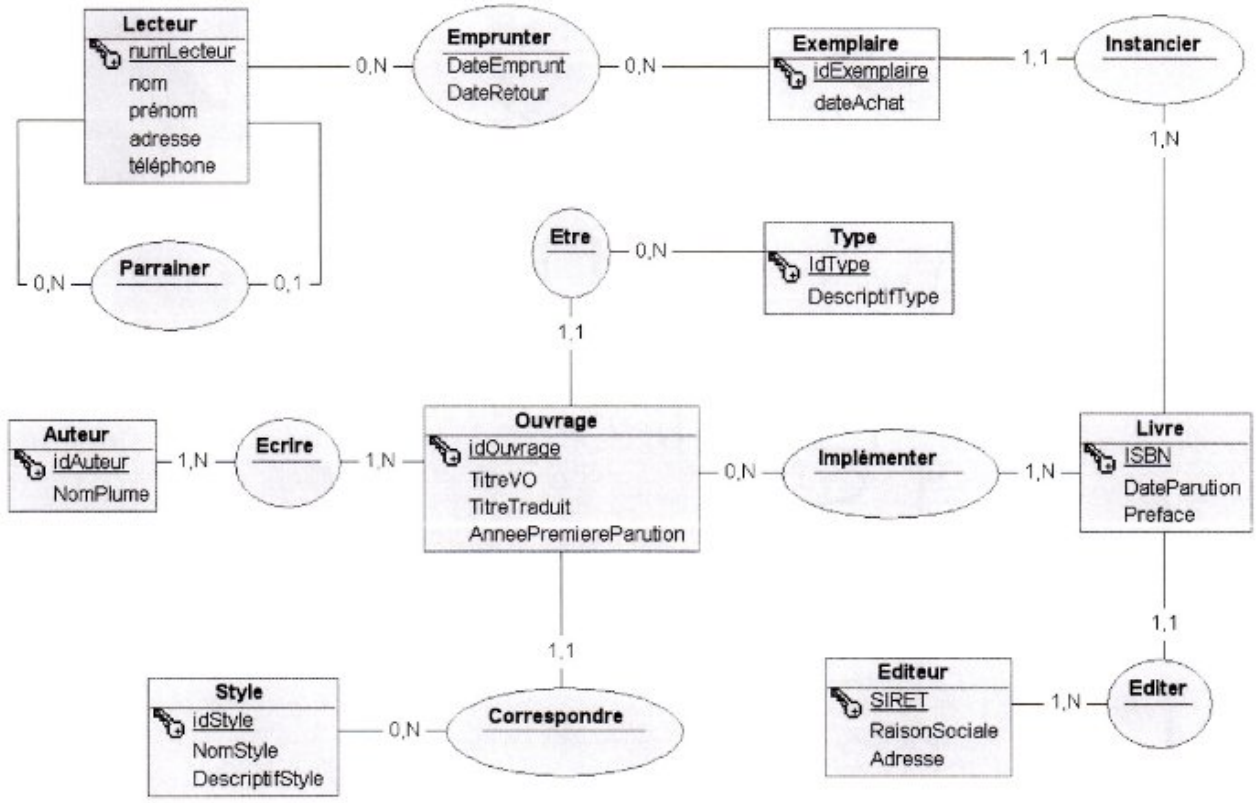
(1,1) - one and only one (mandatory)



Source : http://www.smartdraw.com/resources/tutorials/cardinality-notations/

EXERCICES

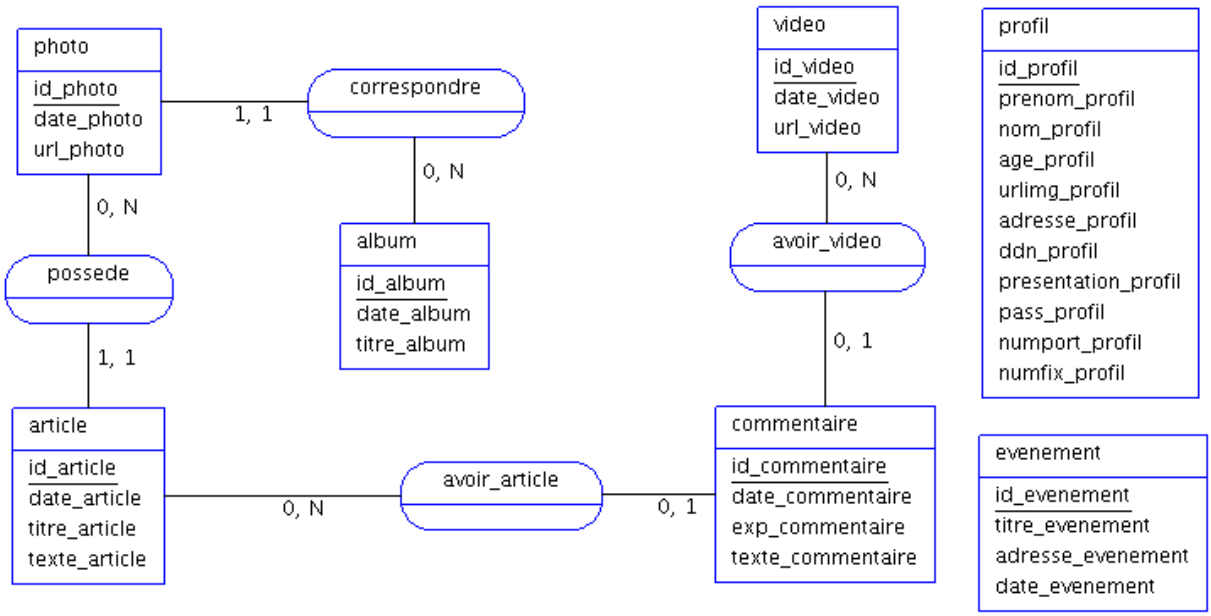
Analyser, critiquer et corriger le schéma suivant :



http://img535.imageshack.us/img535/2984/img001vr.jpg

EXERCICE

Analyser, critiquer et corriger le schéma suivant :



http://img697.imageshack.us/img697/6809/schemaentiterelation.png

Modèle Organisationnel des données : MOD

On ne conserve par rapport au MCD que les données qui seront informatisées.

Pourquoi ne pas informatiser certaines données :

Certaines nouvelles données vont parfois apparaître :

Quantification des données :

Le but de la quantification des données est destiner la taille de la future base de données. Pour cela, nous allons devoir :

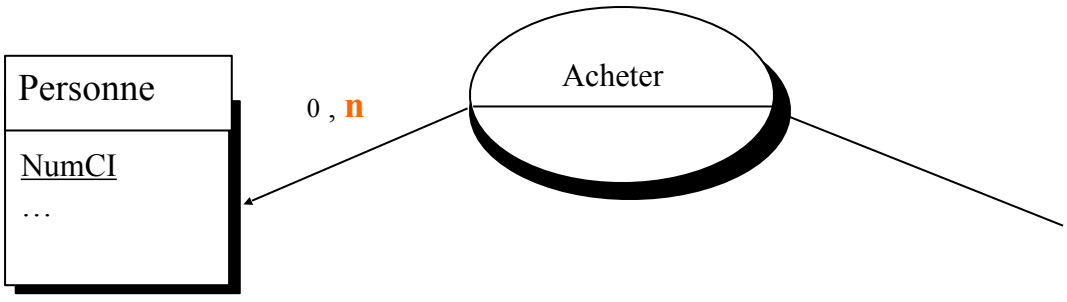
- Estimer la taille des propriétés

- Estimer le nombre d'occurrence des entités en fonction des archivages éventuels (mémoire immédiate <> mémoire à long terme)

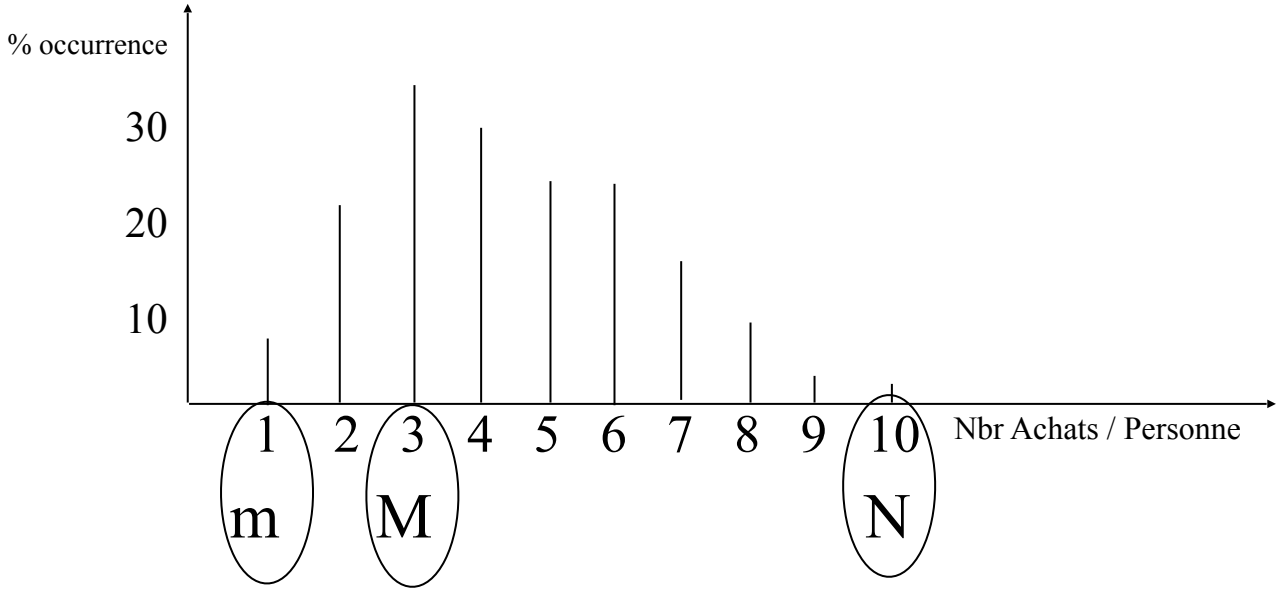
- Quantifier les associations (cardinalités) :

  Le ***taux de participation*** représente le pourcentage d'occurrence de l'entité qui participe à l'association. (0 < p <= 1)

  Exemple : Si on a 10 occurrences de « Personne » et 5 occurrences de « Posséder », le taux de participation se calcule :



La cardinalité moyenne est établie statistiquement en se basant sur la distribution statistique de participation des occurrences à l'association :



Répartition organisationnelle :

Transformation du MOD en MLD

Règles générales de transformation :

- Les **entités** sont transformées en **tables**
- Les **propriétés** deviennent des **attributs** (champs)
- Les **identifiants** deviennent **clés primaires** (PK)
- Les **occurrences des entités** sont des **enregistrements** (records) dans la table

Entité :

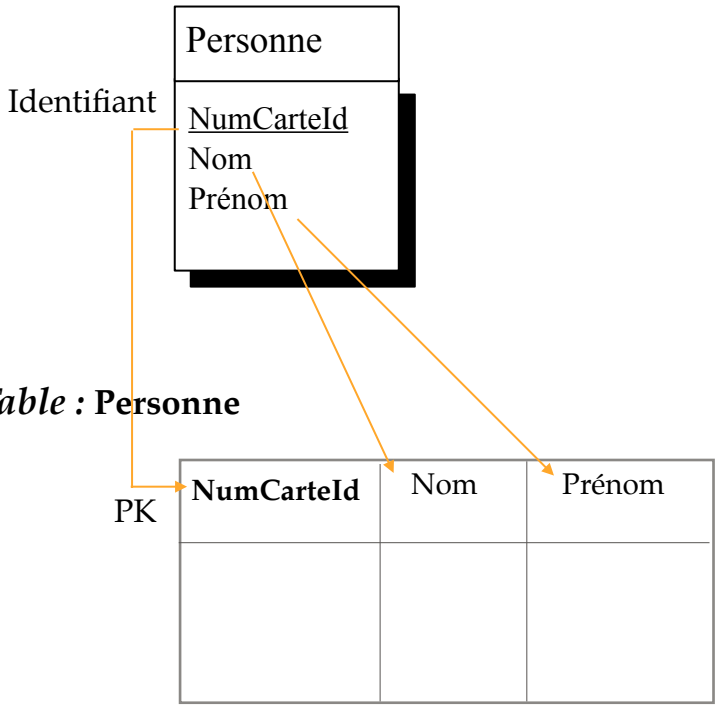
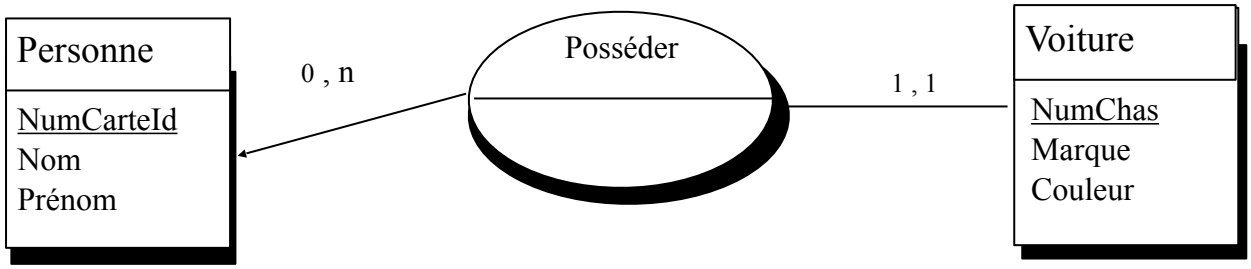


***Table :* Personne**

- ***L***es **associations** se transforment ***selon leurs cardinalités***

Cardinalités 1-N : (0,n) - (1,1)



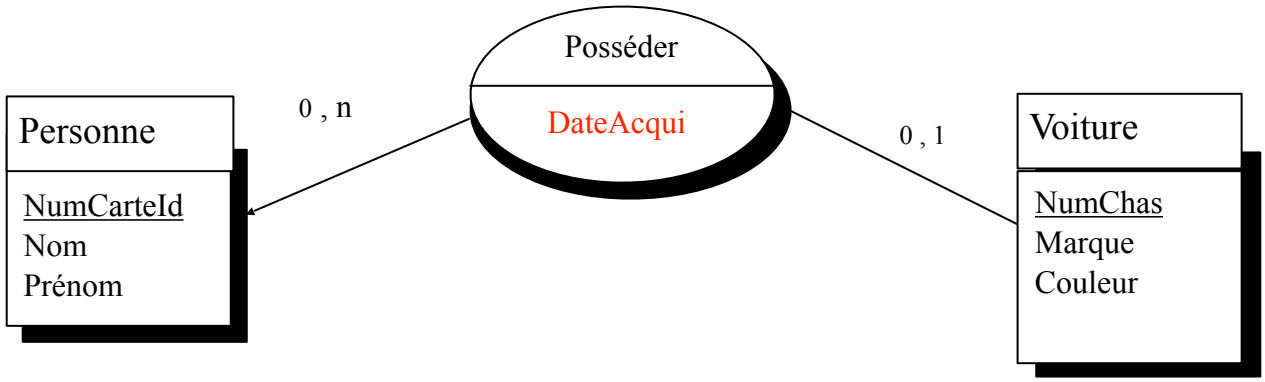
Personne

NumCarteId	Nom	Prénom

Voiture

NumChas	Marque	Couleur

Cardinalités 1-N : (0,n) - (0,1)



- Ce qu'il **ne faut pas faire** :

Personne

NumCarteId	Nom	Prénom

Voiture

NumChas	Marque	Couleur

- **Première possibilité :**

Personne

NumCarteId	Nom	Prénom

Voiture

NumChas	Marque	Couleur

- **Seconde possibilité :**

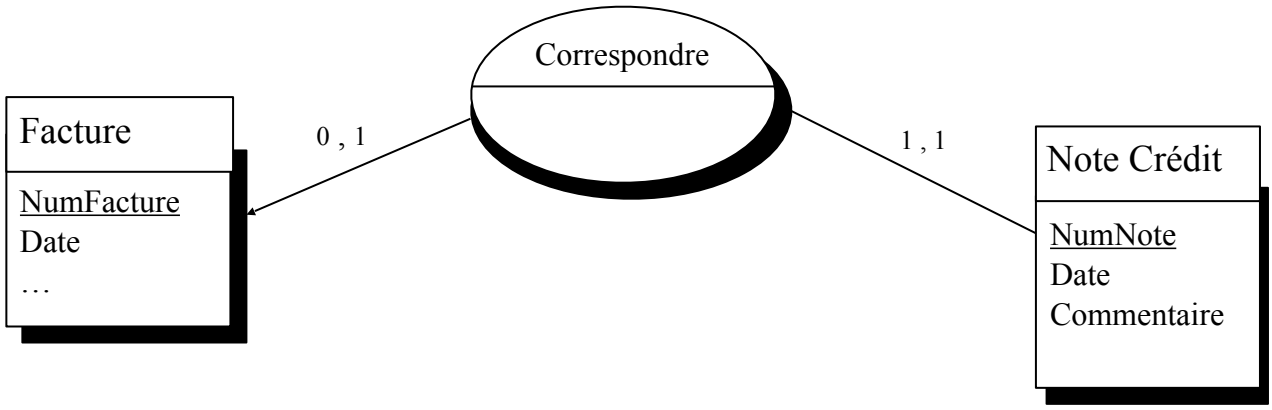
Personne

NumCarteId	Nom	Prénom

Voiture

NumChas	Marque	Couleur

Cardinalités 1-1 : (0,1) - (1,1)



- **Une seule possibilité :**

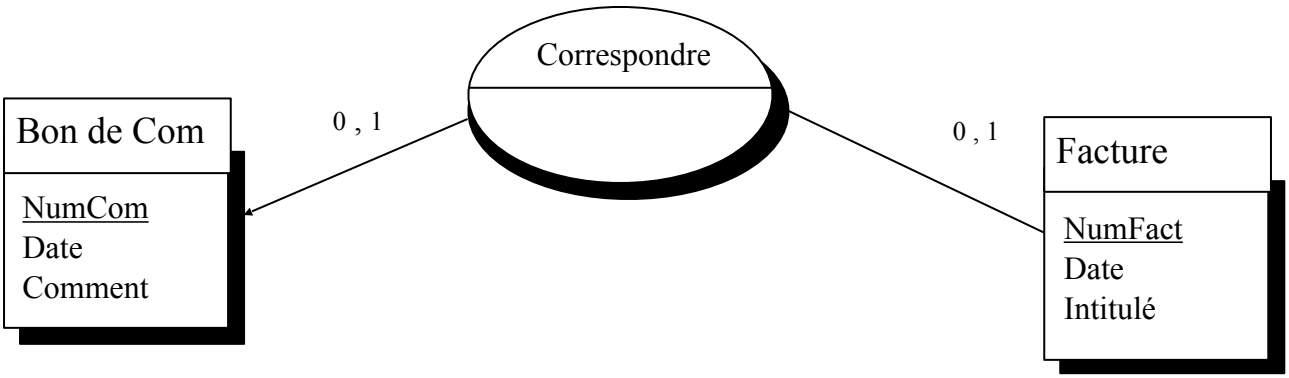
Facture

NumFacture	Date	…

NoteCrédit

NumNote	Date	Commentaire

Cardinalités 1-1 : (0,1) - (0,1)



- **La meilleure solution :**

Facture

NumFacture	Date	Intitulé

BonDeCommande

NumCom	Date	Comment

- **Une autre solution :**

Facture

NumFacture	Date	Intitulé

BonDeCommande

NumCom	Date	Comment

- **Ou encore :**

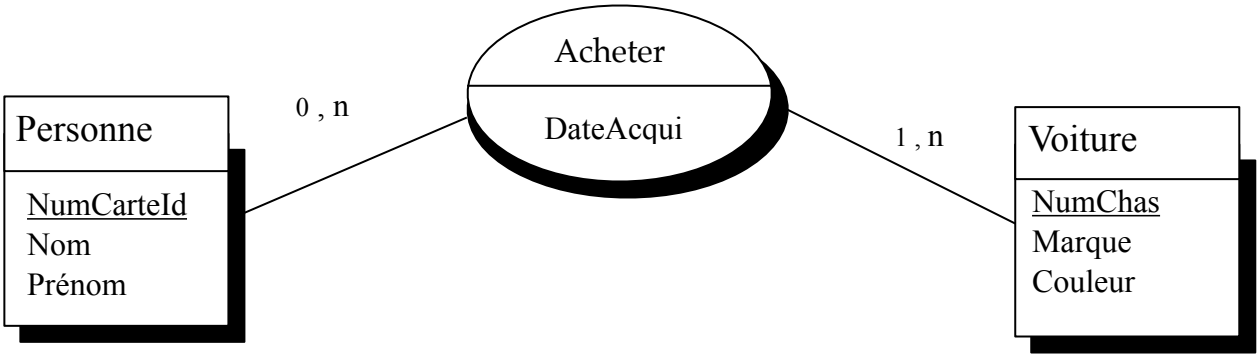
Facture

NumFacture	Date	Intitulé

BonDeCommande

NumCom	Date	Comment

Cardinalités N-N : (0,n) - (1,n)



- **Une seule solution :**

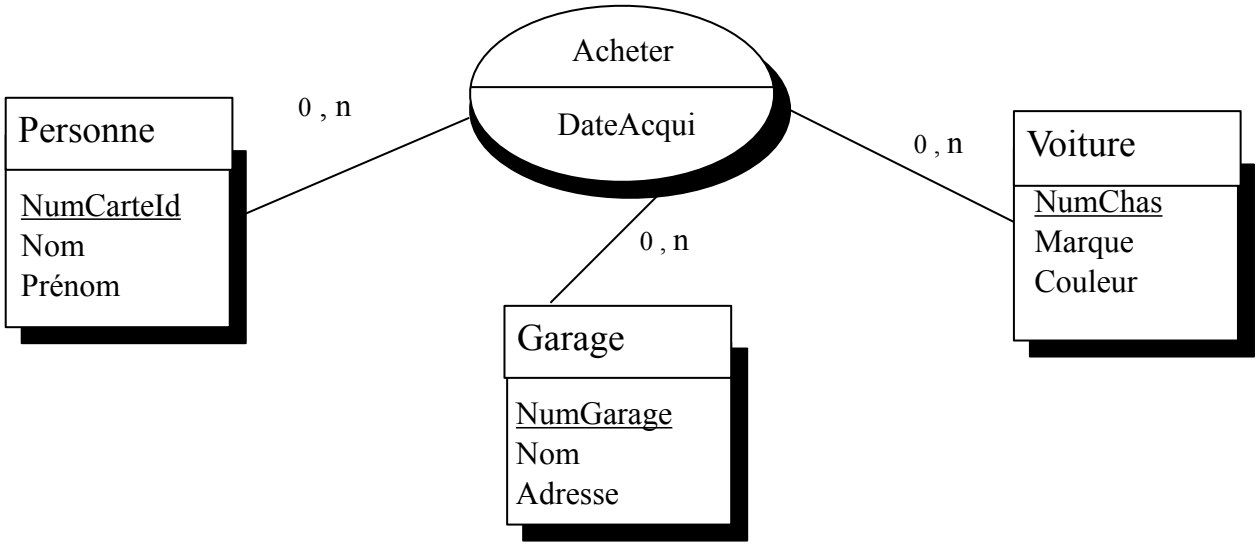
Personne

NumCarteId	Nom	Prénom

Voiture

NumChas	Marque	Comment

Associations N-aires :



- **Une seule solution :**

Personne

NumCarteId	Nom	Prénom

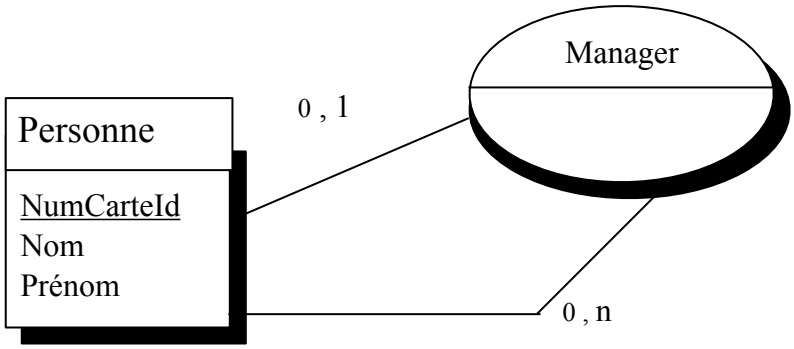
Voiture

NumChas	Marque	Comment

Garage

NumGarage	Nom	Adresse

Auto-associations :

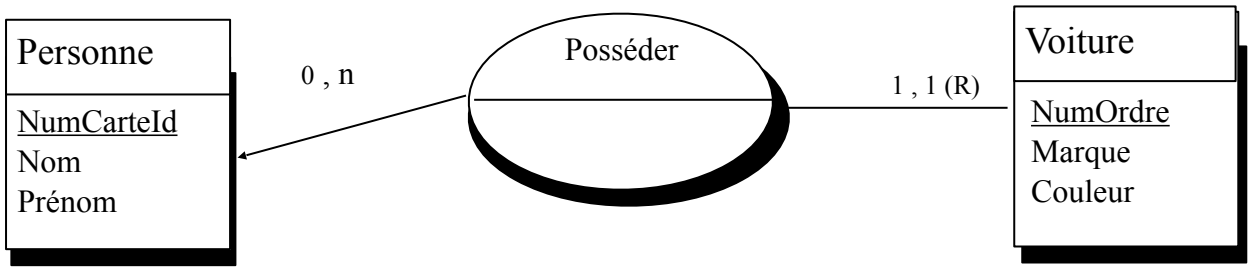


- **Une seule solution :**

Personne

NumCarteId	Nom	Prénom

Identifiant relatif :



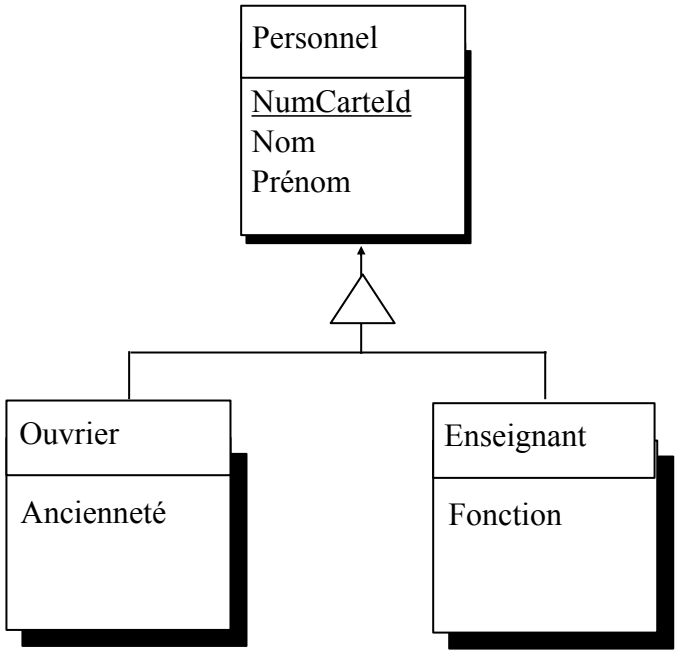
Personne

NumCarteId	Nom	Prénom

Voiture

NumOrdre	Marque	Couleur

Spécialisation / Généralisation :

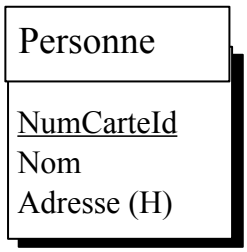


- **Plusieurs solutions :**

Personnel

NumCarteId	Nom	Prénom

Historisation :



- **Une seule solution :**

Personne

NumCarteId	Nom	Adresse

Les dépendances fonctionnelles

Soit R(X,Y,Z) une table avec X, Y et Z des ensembles d'attributs, Z pouvant être éventuellement vide.

On dit qu'il existe une dépendance fonctionnelle de X vers Y si la connaissance de la valeur de X détermine la valeur de Y.

La notation adoptée est la suivante : X -> Y

On dit que X détermine Y ou Y est en dépendance fonctionnelle avec X.

Problèmes posés par les D.F :

S#	City	Status

Propositions de décomposition :

S#	City

S#	Status

S#	City

City	Status

Status	**City**

S#	Status

Les formes normales

On dit qu'une table est dans une forme normale particulière si elle satisfait à un ensemble de conditions prédéfinies par la loi normale correspondante.

D'une manière générale (simplification du langage), on dit qu'une table est normalisée si elle respecte au moins la 3NF.

Il existe des procédures de normalisation qui permettent de transformer une relation dans une forme normale particulière en un ensemble de relation dans une forme normale plus satisfaisante (contraignante).

Exemple :

Voici une table répertoriant les séances des films dans un cinéma

Type	Prix	EIDR	Date + Heure	NumCiné + NumSalle	NbrPlace	Genre1	Genre2

Le « Type » prend comme valeurs : 3D, IMAX, 4K ou SD et induit le prix de la séance (un film en 3D coûte 9 euros, en IMAX 14 euros, …

EIDR est l'ISBN d'un film de cinéma.

Le genre représente : « Science-fiction », « Thriller », …

Cette table n'est pas « Normalisée ». Comment appliquer les lois normales ?

La première forme normale (1NF) : Une table est 1NF si tous les champs contiennent uniquement des valeurs scalaires. Autrement dit, les attributs sont simples (non répétitifs) et élémentaires (non décomposables).

La deuxième forme normale (2NF) : Une table est 2NF si et seulement si elle est 1NF et si chaque attribut ne faisant pas partie de la clé primaire est en dépendance irréductible avec la clé primaire.

La troisième forme normale (3NF) : Une table est 3NF si et seulement si elle est 2NF et si chaque attribut ne faisant pas partie de la clé primaire est en dépendance non transitive avec la clé primaire.

Exercice :

Modéliser et transformer en MLD le problème suivant …

Un complexe cinématographique (style Kinepolis) est implanté dans différentes villes de Belgique et présente des films. Le responsable du système d'information désire informatiser la billetterie (avec réservation de sièges) et la visualisation des séances et des films (avec leurs caractéristiques: réalisateur, acteurs, critiques, …). Les gens qui ont réservé pour un film peuvent écrire un commentaire après avoir vu le film.