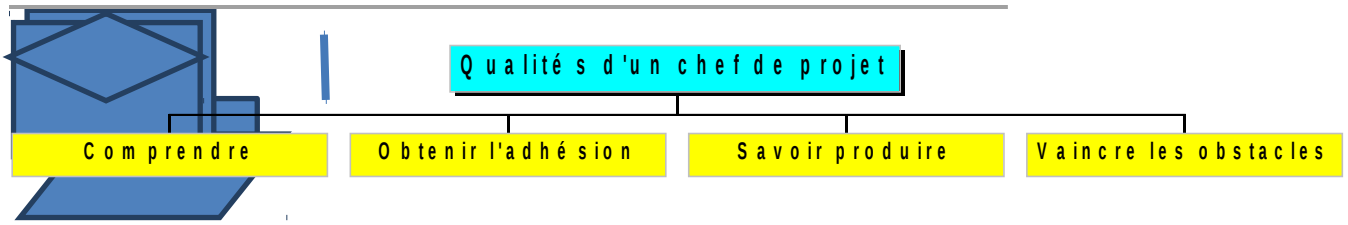


INFORMATIQUE DE GESTION

A l'usage des étudiants de la L2

Ecrit par
[Nom de l'auteur]
Licencié en sciences Mathématiques et Informatiques
Année académique 2015 - 2016

COURS D'INFORMATIQUE DE GESTION

PLAN DU COURS

I. Notion première d'informatique générale

II. Système d'information

III. Gestion des projets

IV. Sécurité informatique

V. La programmation

VI. Exercices et travaux pratiques

PREAMBULE

L'informatique de gestion est la discipline qui traite de l'application des technologies de l'information et de la communication dans les entreprises et les institutions publiques. Elle a notamment pour objectif de mettre au point de nouvelles solutions informatiques, de perfectionner les processus existants et de contribuer à leur déploiement. Elle joue un rôle essentiel dans la coordination entre les différents services des entreprises, comme le marketing, les ventes, la production ou la logistique.

Dans le cadre de ce cours, les étudiantes et étudiants reçoivent une formation initiale exigeante dans le domaine de l'informatique et travaillent notamment sur les systèmes d'information, la gestion de projets, la sécurité informatique, la programmation et les méthodes quantitatives. Le programme inclut également des cours de gestion d'entreprise et d'économie. Ce cours accorde une place centrale à la dimension pratique des enseignements, aussi bien dans le cadre des cours proposés que des travaux personnels. Ce cours permet en outre aux étudiantes et étudiants d'acquérir de précieuses compétences en gestion assistée aux NTIC.

Le programme de cours en informatique de gestion propose une formation universitaire élémentaire dans le domaine de l'informatique, et plus particulièrement de l'informatique de gestion, la spécialité qui traite de l'utilisation des technologies de l'information et de la communication dans les entreprises publiques et privées. Cette discipline a de nombreuses applications passionnantes, comme la réflexion sur le perfectionnement des processus existants et l'élaboration de systèmes innovants à l'aide de solutions informatiques, et participe pour une large part à leur conception et à leur déploiement. Chacun peut mesurer au quotidien l'importance de cette contribution en utilisant divers services sur l'Internet, comme le suivi d'envois postaux, les billetteries en ligne ou encore les plateformes de partage de fichiers ou de vente aux enchères.

La conception et la production de solutions informatiques s'effectuent généralement dans le cadre de projets, qui sont confiés à des équipes regroupant différents métiers représentés au sein de l'entreprise, comme le marketing, les ventes, la production ou la logistique. Dans ce contexte, les informaticiennes et informaticiens de gestion jouent un rôle central de coordination et de pilotage. Ils doivent donc disposer non seulement d'une solide formation en informatique, mais également posséder des connaissances approfondies en microéconomie et en macroéconomie et, bien sûr, de bonnes compétences linguistiques et relationnelles.

L'informatique de gestion regroupe les cadres chargés de développer des applications destinées à la gestion (comptabilité, paie, finance, commercial, logistique, etc.).

Ils ont pour mission principale de paramétrer les spécificités d'une application (logiciel, progiciel, site web) selon les souhaits du commanditaire. Ils sont chargés :

- **d'étudier la faisabilité technique des besoins fonctionnels**. Ils établissent un cahier des charges techniques sur la base des besoins recueillis auprès des utilisateurs (analyse fonctionnelle).
- **de concevoir et prendre en charge tout ou partie du développement d'un produit (progiciel, logiciel).** Ces cadres définissent l'architecture globale du produit, réalisent et intègrent les différentes applications.
- **d'assurer la mise en production.** Ils sont chargés du déploiement du système d'information ou des applications à l'ensemble de l'entreprise (ou de l'entité).
- **d'effectuer la tierce maintenance des applications**. Ils prennent en charge la maintenance (évolutive, corrective et applicative) et préparent les évolutions du SI.

Parmi les métiers représentés dans la fonction informatique de gestion, on peut citer le chef de projet informatique technique, l'ingénieur d'étude développement, l'ingénieur développement logiciel et le consultant technique.

Chap1. Notion première d'informatique générale

L'informatique est la science du traitement automatique de l'information par l'ordinateur.
Elle a pris source depuis l'apparition des premiers calculateurs avant 1940, elle a montré son efficacité dans la gestion de stock, personnel, dans l'intégration du son et image, aussi dans l'interconnexion des réseaux intra et internet (Echanges des données), et dans la messagerie.

1.2 Présentation de l'ordinateur

Un ordinateur est un dispositif capable de traiter des informations. Il est formé d'un certain nombre d'éléments (écran, clavier, souris,...) reliés les uns aux autres, souvent par des fils. Pour comprendre comment fonctionne un ordinateur, il est important d'apprendre à distinguer ces éléments et à voir comment les informations circulent entre - eux.

1.1 éléments de base :

- un *écran*
- un *clavier*, pour taper du texte
- une *souris*, pour déplacer le *curseur* à l'écran
- des *enceintes* pour le son, ce n'est pas obligatoire mais tout de même mieux
- et surtout : une *unité centrale* qui est le coeur et le cerveau de l'ordinateur

1.2.1 Les périphériques d'un ordinateur

2.1 Quelques périphériques :

- *imprimante*, *scanner*
- *webcam*, *camescope numérique* , *appareil photo numérique*
- connexion Internet par câble ou sans fil (*Wifi*)
- *disque dur externe*, *clé USB*, *carte mémoire*...
- manette de jeu
- ...

Remarque :

Il existe des modèles consolidés appelés « Ordinateur portables » ;

Un *ordinateur portable* se doit d'être compact et facilement transportable. C'est un concentré d'*unité centrale*, d'*écran*, de *souris*, de *clavier*, de *webcam* intégrée et de son.

Tout comme l'ordinateur fixe, il est possible de brancher différents périphériques : *imprimante*, *souris*, *appareil photo numérique* ...

L'*ordinateur portable* possède une *batterie* qui lui permet d'être autonome jusqu'à plusieurs heures sans être alimenté en courant électrique.

Dès nos jours, branché un ordinateur est un jeu d'enfant, c'est réellement le cas ! Car chaque branchement a une forme et une couleur bien définie, c'est qui fait qu'il est presque impossible de se tromper. De plus à l'heure actuelle, la plus part des périphériques d'un ordinateur, souris, clavier, appareil photo, mémoire amovible, webcam se branchent tous via un branchement universel appelé port USB[1]

1 USB : Universal serial bus

1.2.2 L'arrière-plan d'un ordinateur

Voici l'arrière d'une *unité centrale*, où chaque branchement est indiqué par une couleur :

l'alimentation électrique, qui est reliée directement à une prise secteur. Un bouton 0 - 1 permet de couper l'arriver de courant.

les anciennes prises pour *clavier* et *souris*, rondes et vertes ou violettes.

Les anciens ports *COM* et parallèles, qui ne sont plus utilisés de nos jours.

Les ports *USB*, au nombre de 4 sur la photo, permettant de brancher divers périphériques. Ce sont actuellement les ports les plus utilisés !

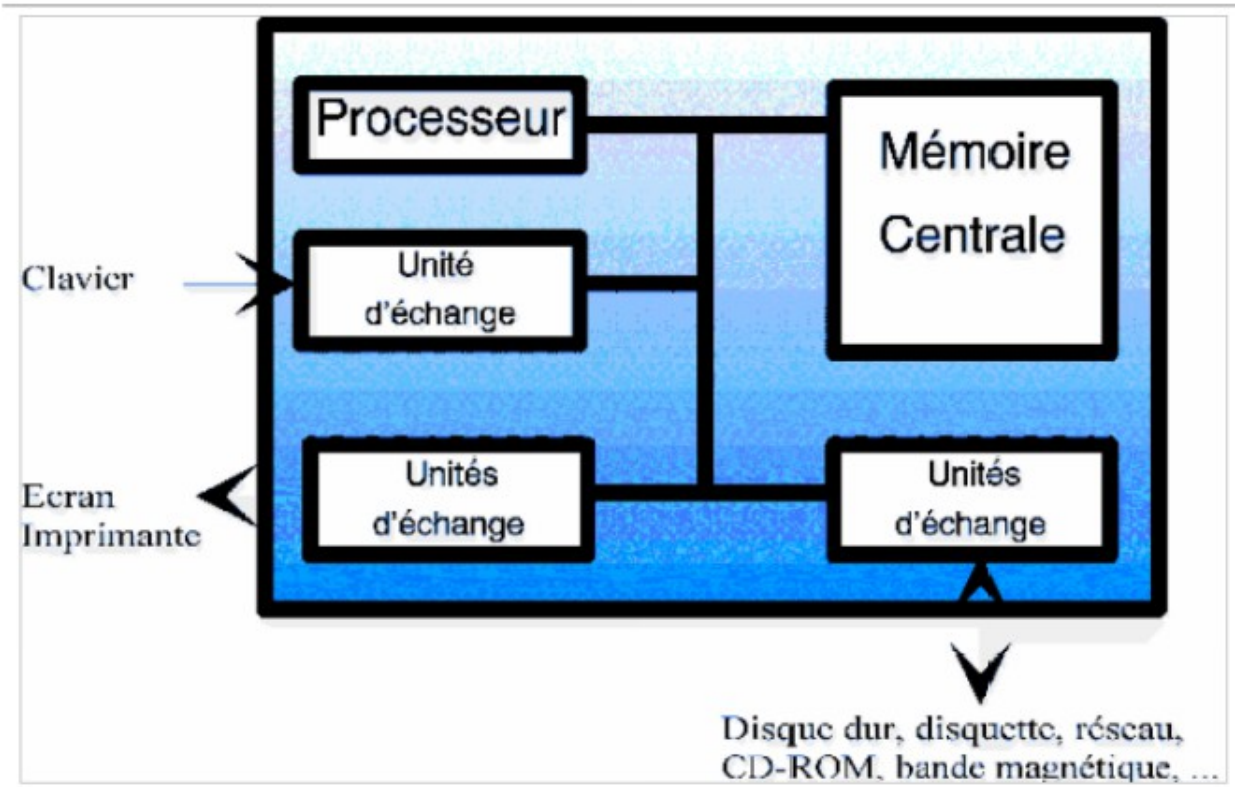
Le port pour brancher l'*ordinateur* à *Internet* ou sur un *réseau*.

Les prises son : pour brancher enceintes, caisson de basses, micro.

Un branchement *DVI* (blanc rectangulaire) pour brancher les nouveaux écrans et un *VGA* (bleu rectangulaire) pour les anciens écrans.

Bien entendu, ces branchements peuvent varier d'un ordinateur à l'autre selon son ancienneté.

En résumé, voici le schéma structurel et fonctionnel d'une unité central :



On vient de voir la structure du matériel, On parle aussi de HARDWARE (qu'on a francisé en matériel) ; En anglais, ce mot veut dire "quincaillerie", et il désigne dans le jargon informatique tout ce qui est matériel, par opposition à tout ce qui est logiciel ou SOFTWARE, qu'on peut traduire par le « mou », c'est-à-dire les données présentes sous formes de courants électriques et de pistes magnétiques. Toute cette partie est désignée par le terme "software".

1.1 Les Logiciels

Introduction

Une fois le PC[2] branché, vous appuyez sur un gros bouton pour la mise en route. Et Vous entendez certains bruits dus à des moteurs en rotation, et effectivement, il y a plusieurs choses qui tournent dans votre ordinateur, les ventilateurs, le disque dur, et tout cela chauffe pas mal.
Au moment du lancement, le processeur dit aussi "puce", ou "CPU" ("Central Processor Unit"), demande au disque dur de charger le « système d'exploitation » dans la mémoire vive ou RAM.
S'il n'y a rien sur le disque dur, eh bien, il ne va rien se passer.

2 PC : Personal Computer

1.2 .1 - LE SYSTEME D ' EXPLOITATION

Nous disons qu'il faut avoir au préalable installé sur l'ordinateur un « système d'exploitation», un ensemble des programmes de base permettant d'utiliser et gérer les ressources matérielles de la machine ainsi que les échanges avec le monde extérieur (les périphériques, le réseau Internet….)
Cette installation est en général faite par le vendeur de l'ordinateur à partir d'un CD-ROM qu'il vous remettra. Le système d'exploitation est cette première couche de programme qui vous permet de lancer ensuite les programmes avec lesquelles vous travaillez habituellement, comme le traitement de texte ou le programme de gestion d'e-mails. Les données qui composent le système d'exploitation sont lourdes et nombreuses, c'est pourquoi au démarrage, elles viennent se loger, en partie, dans la mémoire vive, petites barrettes enfichées sur la carte mère, support du processeur, qui permet des transferts de données beaucoup plus rapides que le disque dur.
Notons cependant un inconvénient majeur de la mémoire vive, est tel que, dès que l'électricité est coupée, la mémoire vive se vide complètement, contrairement au disque dur, d'où la nécessité de ce chargement au démarrage.

Le Système d'exploitation ou Operating System (en anglais) ou OS les plus répandus sont :

A. **WINDOWS** le plus répandu puisque fonctionnant sur PC, le matériel le plus répandu sur la planète. Ses trois dernières versions sont Seven, Windows 8 et Windows 10. (Windows est la propriété de MICROSOFT, du célèbre Bill GATES)

B. **MAC-OS** pour les ordinateurs MAC-INTOSH ; (il est la propriété de MAC-INTOSH, du non moins célèbre Steve JOBS)

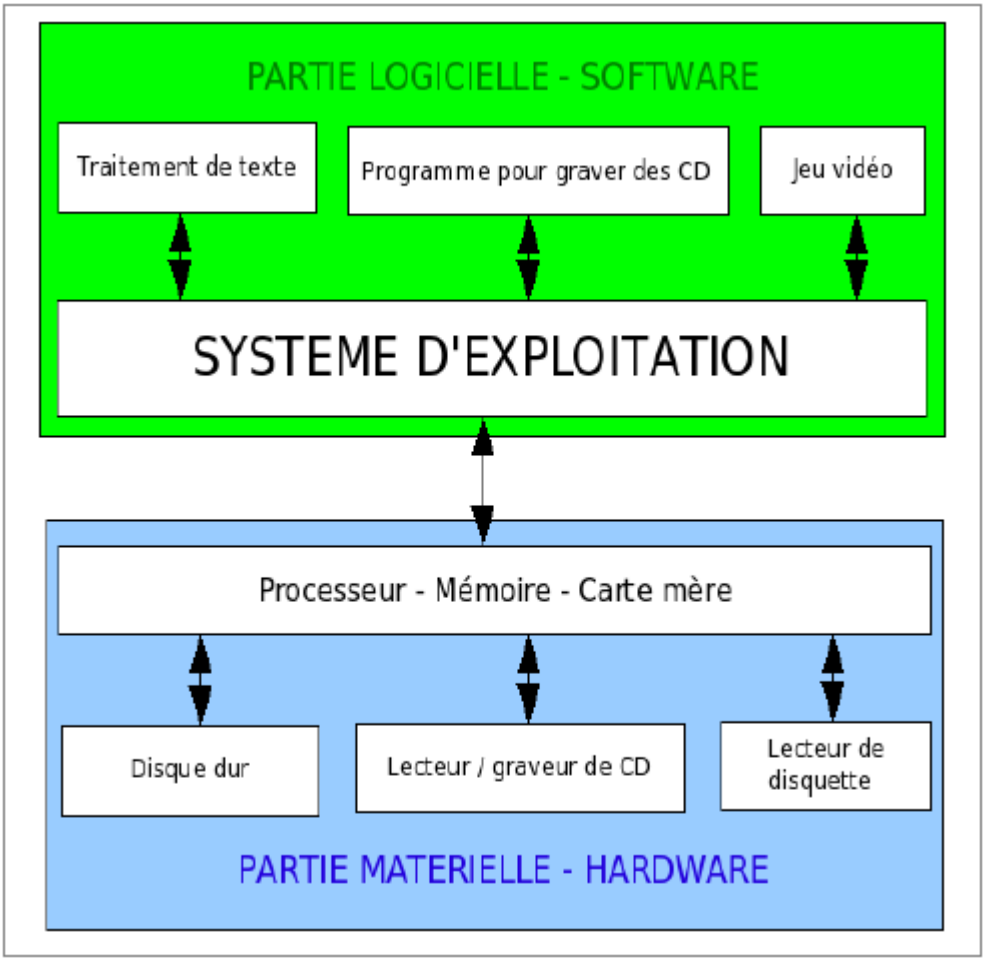
C. **LINUX** plus récent, donc de conception plus moderne, et qui possède deux versions, une pour PC et une pour MAC. Linux est un logiciel dit « libre » en ce sens qu'il est fourni avec tous les éléments permettant de le modifier, c'est-à-dire de le personnaliser. WINDOWS et MAC-OS sont dits « propriétaires » donc dépourvus de moyens d'adaptation. En plus, c'est interdit !...Pour compléter, savoir que LINUX (qui existe en deux versions) peut très bien cohabiter sur une machine avec WINDOWS pour les PC et MAC-OS pour les MAC

N.B : Sur un PC on installe WINDOWS, et sur un MAC, on installe MAC-OS

1.2. 2- LES AUTRES LOGICIELS

Le système d'exploitation n'est donc que la première couche. La deuxième, ce sont les programmes, appelés aussi applications, logiciels ou exécutables, qui la constituent.
Les systèmes d'exploitation de type Linux intègrent des centaines de programmes dès leur installation, et sont donc complètement opérationnels immédiatement. On peut, si l'on veut, ajouter des programmes très spécialisés. Les systèmes d'exploitation Windows fournissent des programmes (donc des outils) spécifiques à Windows, certains assez complets, d'autres plus rudimentaires ; par exemple le traitement de textes Wordpad (pour une utilisation assez simple)

mais dès que vous voulez un bon traitement de texte, il faut se tourner vers un programme spécifique. C'est là que beaucoup de gens s'imaginent qu'il va falloir mettre la main au portefeuille, alors qu'il n'en est rien. Car on trouve désormais, beaucoup de programmes gratuits qui vous donnent accès à des fonctions déjà très élaborées. Les équivalents payants existent, mais ils ne sont souvent pas meilleurs, ni plus complets... et les gens les achètent quand même... comprenne qui pourra.
Voici, pour finir, une illustration des flux de données à l'intérieur d'un ordinateur. J'espère que cela vous aidera à mieux saisir les explications de cette page.
La figure suivante montre la structure du système d'information dans un ordinateur.



1.3 Quelques remarques importantes dans l'utilisation de l'ordinateur

Au début, l'utilisateur est un peu "inhibé" par :

A- la peur de mal faire:

Principe: la machine ne sait rien faire toute seule donc dans tous les cas, il faut faire quelque chose, et la solution dans tous les cas, exécuter une commande, soit avec la souris, soit avec le clavier. Prendre l'habitude de lire les messages affichés, répondre aux questions

B- le vocabulaire:

Parfois en anglais, pas toujours explicite (papier-peint, tonner, navigateur..), on s'y habituera puisqu'on va utiliser ces notions.

C- la lecture des messages:

Toujours lire les messages, toujours y répondre, attention aux tournures de phrases! ne pas répondre « oui » systématiquement (bien lire la question..)

N.B: « annuler » n'est pas répondre

D- l'organisation:

Peut-être complexe, utilise des termes spécifiques, évolue avec le temps ; et la solution, connaître peu à peu l'organisation de sa machine ; être rigoureux dans le rangement de ses dossiers utiliser des noms simples et explicites, ne pas multiplier les noms semblables, cette organisation sera vue dans les séances pratiques.

1.4 Représentation des données

Dans un PC, on ne connaît que 2 chiffres : 0 et 1 qui représentent l'état d'un circuit électrique, soit : 0 = éteint 1 = allumé

Les données sont donc représentées, en mémoire, sur un disque, par des 0 et des 1

……1100011010100100101001111000 01

Avec ces deux symboles, il va falloir représenter les nombres, l'alphabet, les signes de ponctuation et d'accentuation….. Les couleurs et les sons.

Observation

Dans notre système décimal, on a dix symboles (de 0 à 9 avec lesquels on sait écrire tous les nombres). On est dans un système dit à base 10 (ou décimal). En informatique, on a 2 symboles (0 et 1 avec lesquels il va falloir écrire tous les nombres). On est dans un système dit à base 2 (ou binaire).

En décimal, quand on arrive à 9 on fait

```
  9
 +1
----
 10
```

"On dit : 9+1= zéro et je retiens 1" qui se lit « dix »

En binaire, c'est pareil.

Quand on arrive au dernier chiffre, le 1, on fait

```
  1
 +1
---
 10
```

"On dit : 1+1 = zero et je retiens 1" qui se lit « un zero »

Représentation des nombres en décimal et en binaire

En numération décimale (base 10), chaque chiffre d'un nombre a un poids selon sa position. Ce poids est un multiple de 10 (puissance de 10) (1 10 100 1000....).
Chaque position "pèse" la précédente multipliée par 10

1- Représentation décimale (base 10)

poids décimal	1000	100	10	1	
nombre décimal	0	2	9	5	
valeur du nombre : (en décimal)	0*1000 +	2*100 +	9*100 +	1*5	= 295

2- Représentation binaire (base 2)
On fera la même chose, sauf que le poids de chaque position sera celui de la précédente multiplié par 2 (et non par 10)
Par exemple :

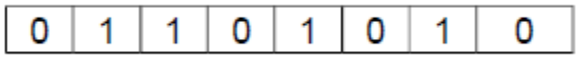
Poids binaire	128	64	32	16	8	4	2	1	
nombre binaire	0	1	0	1	0	1	1	0	
valeur du nombre : (en décimal)	0*128+	1*64+	0*32+	1*16+	0*8+	1*4+	1*2+	0*1	=86

Soit, plus rapidement : 64+16+4+2 = 86

maintenant, on va compter en binaire :

nbre décimal		nombre binaire
0	**---->**	**0 0 0 0**
+1		+1
=1	**---->**	**0 0 0 1**
+1		+1
=2	**---->**	**0 0 1 0**
+1		+1
=3	**---->**	**0 0 1 1**
+1		+1
=4	**---->**	**0 1 0 0**
+1		+1
=5	**---->**	**0 1 0 1**
+1		+1
=6	**---->**	**0 1 1 0**
+1		+1
=7	**---->**	**0 1 1 1**
+1		+1
=8	**---->**	**1 0 0 0**
+1		+1
=9	**---->**	**1 0 0 1**
+1		+1
=10	**---->**	**1 0 1 0**
+1		+1
=11	**---->**	**1 0 1 1**

En informatique, chacune de ces positions binaires (0 ou 1) s'appelle un bit noté **b** , abréviation de **bi**nary digi**t.**



Les bits ont été groupés par huit. Un tel groupe s'appelle un **Byte** ou **Octet,** le Byte (ou Octet) est la taille de la plus petite unité adressable d'un ordinateur, les tailles mémoire et disque se calculent en octets (ou bytes)

les multiples de l'octet :

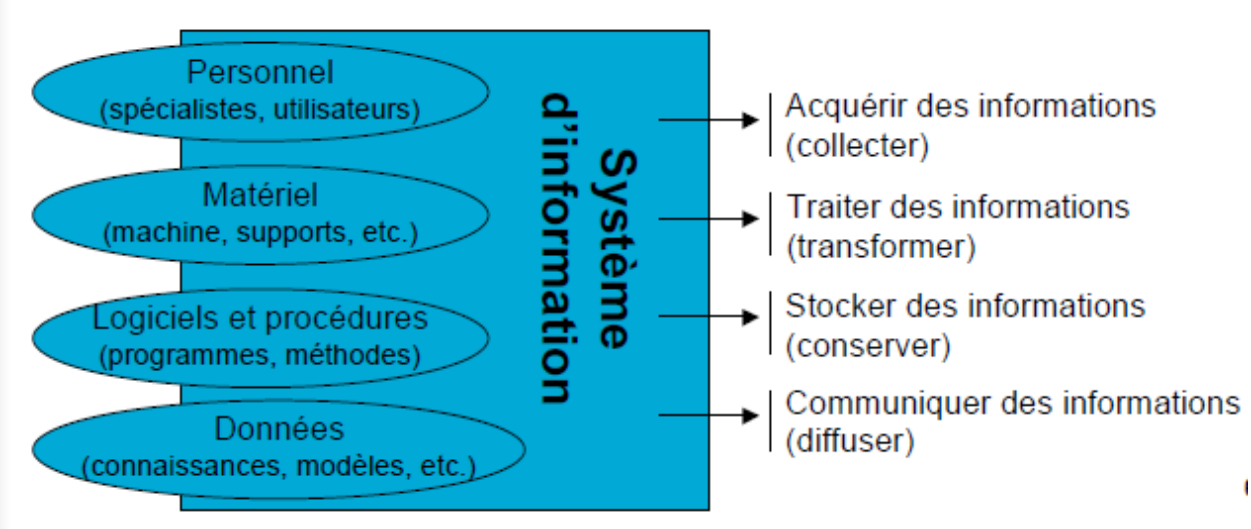
1 Ko (kilo octets) est égal à	1 000 octets	10^3 octets
1 Mo (mega octets) est égal à	1 000 000 d'octets	10^6
1 Go (giga octets) est égal à	1 milliard d'octets	10^9
1 To (tera octets) est égal à	1000 milliard d'octets	10^12

Quelques points de repères :

Une page (texte Word), c'est 30 Ko
Une photo (prise à 3 millions de pixels), « pèse » ~3Mo, soit 3 000 KO
Une minute de musique, c'est ~10 Mo soit 10 000 000 ko 10 000 KO
Donc, sur un CD (700 Mo de capacité) on pourra mettre : 700 000 KO ; entre 300 et 350 photos de ~ 2 Mo ; ou ~ une bonne heure de musique (non compressée)
Un disque dur contient de 500 Go (500 milliards octets)
Une barrette de mémoire RAM contient 1Go (100 000 KO)
Les disques externes de 1 Téra octets (1000 milliards) sont sur le marché depuis peu.
Les vitesses de transmission se calculent en kilobits par seconde (ex : 56 Kbits/s)
Et pour finir :
La puissance du processeur se mesure par le nombre de cycles exécutés par seconde.
Un processeur de 1,66 Mégahertz/s effectue 1,6 milliards de cycles par seconde.

Chap. II. Système d'information

L'organisation peut mesurer ses performances grâce aux différents indicateurs à sa disposition. Afin de piloter l'organisation, tenté d'atteindre la performance globale et prendre des mesures de correction de certaines actions, l'organisation doit mettre en place un système d'information qui rassemble l'ensemble **des indicateurs de performance**[3] **et leurs résultats**.
Un SI est un ensemble organisé de ressources : matériel, logiciel, personnel, données, procédures… permettant d'acquérir, de traiter, de stocker des informations (sous formes de données, textes, images, sons, etc.) dans et entre des organisations.



L'organisation est à la base de l'action collective. Dès qu'une activité dépasse la capacité d'un seul individu, l'organisation constitue la réponse appropriée. Elle se caractérise donc par :
- un ensemble d'individus : participants, acteurs ;
- un accord, implicite ou explicite, sur un ou plusieurs objectifs partagés par les divers participants ;
- une division du travail, définissant le rôle de chaque participant ;
- une coordination plus ou moins formalisée, qui assure les objectifs communs en dépit de la division du travail.

1. Le rôle du système d'information et de décision

Le système d'information et de décision permet de stocker, traiter, restituer ou diffuser les informations nécessaires au fonctionnement de l'organisation au moyen de l'informatique et des nouvelles technologies de l'information et de la communication (NTIC) comme les réseaux intranet, Internet…

2. Les productions du système d'information et de décision

3 Appelé aussi KPI : Key performance indicator

Grâce aux informations transmises et traitées, le système d'information et de décision va restituer des informations essentielles à la prise de décision.

A. Les documents comptables

Les documents comptables permettent d'évaluer la santé de l'organisation et de prendre des décisions en conséquence (bilan, compte de résultat, calcul du seuil de rentabilité, chiffre d'affaires, niveau de trésorerie…).

B. Les tableaux de bord

Les tableaux de bord regroupent toutes les informations issues des indicateurs de performance et vont permettre de **mettre à jour les dysfonctionnements** et donc de les corriger (délais, coûts, prise de risque…).

Ils reposent sur la définition d'**objectifs à atteindre**, sur la réalisation et sur les écarts recensés entre les objectifs et la réalisation.

C. Le bilan social

Le bilan social est mis en place dans les structures qui comptent au moins 300 salariés. C'est un document qui présente l'organisation en sept catégories :

• l'emploi,
• les salaires,
• les conditions d'hygiène et de sécurité,
• les conditions de travail,
• la formation,
• les relations professionnelles,
• et autres conditions de vie liées à l'organisation.

Le système d'information et de décision doit permettre un pilotage efficace de l'organisation en amenant des mesures de correction des actions déjà menées.

II.1. Présentation : du système d'information à la gestion de projet

Le système d'information permet la **circulation d'information** au sein de l'organisation, il **est amené à évoluer régulièrement** de façon à permettre à l'organisation de pouvoir **s'adapter** aux variations de son environnement socio-économique.

Pour pouvoir faire évoluer le système d'information la direction du système d'information (DSI) de l'organisation doit définir régulièrement des **projets d'adaptation du SI** tant au niveau d'une **étude de l'existant** (notion d'audit) et des **perspectives d'évolution** d'un point de vue technologique (souvent informatique).

2. Le vocabulaire de la gestion de projet d'un SI

• **Un modèle** (référence à la modélisation du SI) est une représentation de la réalité. Le modèle permet d'**observer un système d'information**, en permettant de **tester sa réaction** aux divers événements de la vie de l'organisation. La conception d'un ensemble de modèles au sein d'une organisation revient à concevoir l'état futur du système d'information.
• **Une méthode** est la mise en œuvre d'**un certain nombre d'étapes** (méthodologiques) :

- une démarche,

- des principes,

- des outils (traces papiers standardisées, informatiques),

- un vocabulaire.

• **L'analyse** consiste d'une part à **comprendre et modéliser le fonctionnement d'un domaine d'étude** du système d'information de l'organisation, et d'autre part à **préparer et concevoir la solution informatique** adéquate.

On distingue deux types d'analyse :

- l'analyse de l'existant (étude du domaine),

- l'analyse conceptuelle (préparation d'une solution technologique).

• **Le domaine d'étude** au sens de « domaine de gestion » correspond à une **division du système d'information** de l'organisation que l'on étudie en **sous-ensembles cohérents** (domaine financier et comptable, domaine commercial, domaine gestion du personnel...) ; le domaine d'étude pourra être défini comme un **processus** (gestion financière comptable, gestion commerciale, gestion du personnel...).

3. Le projet et ses objectifs

Le projet correspond à la réponse à un **besoin** (amélioration de la productivité, adaptation à l'évolution des lois, objectifs stratégiques d'adaptation au marché…) de l'organisation (décideurs et utilisateurs) répondant à **l'analyse d'un domaine d'étude** précisément défini (activité ou processus) utilisant l'une ou des **méthodes spécialisées** dans la gestion de projet. Le projet est un **ensemble d'étapes méthodologiques** qui doit permettre à partir de la réalité du domaine d'étude (**étude de l'existant/abstraction**) :
- de définir précisément le besoin par l'élaboration d'un cahier des charges (**spécification**),
- d'élaborer méthodologiquement une modélisation du système d'information étudié (**conception**),
- de construire l'application technique correspondant aux besoins (**développement**),
- de valider en fonction des besoins exprimés par le cahier des charges (**test**),
- de la livrer en production aux utilisateurs (**exploitation**).

Remarque : il existe de très nombreuses méthodes de gestion de projet qu'elles soient généralistes ou spécialisées en informatique ; le programme de terminale utilise un certain nombre d'outils issus de méthodes ou langages tels que Merise, UML ou encore de gestion ou de planification (comme le diagramme de Gantt).

Le projet se définit à partir de **3 contraintes** :

• **la qualité** qui consiste à répondre de façon optimale aux besoins exprimés,

• **le coût** qui consiste à gérer au milieu le budget,

• **le délai** qui consiste à optimiser la gestion du temps de réalisation.

4. Le projet informatique

La gestion d'un projet d'évolution du système d'information d'une organisation aboutit le plus souvent à la **définition d'un projet d'informatisation** du domaine d'étude ciblé par le projet. Un projet informatique utilise une ou des **méthodes spécialisées** dépendant du « génie logiciel ».
Le génie logiciel comprend l'ensemble des techniques théoriques et pratiques qui interviennent dans l'élaboration et le développement d'un logiciel. La notion de « Logiciel » incluant les catégories de logiciels de tous les domaines possibles.
Les méthodes de gestion de projets informatiques issus du « génie logiciel » intègrent toujours les **objectifs** suivants :
• la fiabilité,
• la performance,
• la maintenance facile,
• un temps de production raisonnable,
• la réduction des coûts de production.
On note qu'il est **impossible d'atteindre l'ensemble de ces objectifs**, certains étant en conflits avec d'autres. L'élaboration d'un projet informatique devra prendre en compte un juste compromis entre les différents objectifs à atteindre.
Exemple : la recherche de la fiabilité d'une application informatique se fera au détriment de la réduction des coûts ou encore la recherche de la performance d'une application informatique se fera au détriment d'une réduction du temps de production alors que la fiabilité de l'application informatique entraînera une maintenance facilitée.

5. Modélisation d'un système d'information

Dans la conception d'un système d'information, la **modélisation des données** est l'analyse et la conception de l'information contenue dans le système.

Il s'agit essentiellement d'identifier les entités logiques et les dépendances logiques entre ces entités. La modélisation des données est une représentation abstraite, dans le sens où les valeurs des données individuelles observées sont ignorées au profit de la structure, des relations, des noms et des formats des données pertinentes, même si une liste de valeurs valides est souvent enregistrée. Le modèle de données ne doit pas seulement définir la structure de données, mais aussi ce que les données veulent vraiment signifier (sémantique).

1. Présentation : les objectifs de la modélisation de processus

Rappel : un **processus organisationnel** regroupe un **ensemble d'activités coordonnées** permettant de **répondre à un besoin** de l'organisation en **mettant à disposition de chaque acteur/service les informations** lui permettant d'effectuer au mieux son travail.

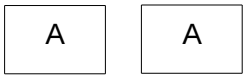
La modélisation d'un processus correspond à la **représentation** sous forme **verbale puis graphique** des **acteurs** et des **tâches** qu'ils accomplissent dans le cadre du déroulement du processus. Cette **représentation** issue de l'analyse du système d'information du processus permet d'avoir une **vision synthétique « universelle » simple** reconnue par tous les acteurs/services de l'organisation (uniformité du langage) et permet :

• aux **acteurs/services** de **communiquer** leurs connaissances de l'environnement du processus,
• à la **DSI** (direction du système d'information) de **comprendre** la structure et l'organisation du processus,
• à la **DSI** d'**analyser** l'existant (représentation de la réalité du fonctionnement) et d'évaluer les évolutions à mettre en œuvre pour l'améliorer,
• à la **DSI** (service informatique interne ou SSII externe) de **développer** l'application informatique représentant tout ou partie du processus.

2. Modélisation de processus

Retenons :

Ou : Pour définir le début d'un processus



: Un point d'arrêt ainsi que son point de connexion correspondant

a. Les acteurs du processus

L'acteur interne réalise ses tâches dans le cadre du domaine d'étude du processus ; il participe à la production du résultat du processus.

: Toutes les données « acteurs internes » seront présents dans ce symbole

L'acteur externe est créateur (événements entrants) ou récepteur (événements résultats) d'information participant au processus mais externe au domaine d'étude ; il n'effectue aucune tâche (opération ou activité) du processus.

: Toutes les données « acteurs externes » seront présents dans ce symbole

Attention ! Un acteur externe peut-être un membre de l'organisation.
Exemple : dans un établissement scolaire le cuisinier qui est un membre de l'organisation est un acteur externe du processus « gestion des absences élèves » quand il signale que sa fille sera absente en cours.

b. Les événements

L'événement est un **flux d'information** qui représente éventuellement un **flux matériel**. Il existe deux types d'événement :

• **l'événement déclencheur** qui entraîne la mise en œuvre d'une activité (opération ou tâche) du processus et qui représente soit :
- un flux d'information,
- la représentation d'un flux matériel,
- une représentation du temps (début de journée, à 15 h, début d'année…).

: Symbole prenant en charge les données « déclencheurs » à passer au traitement.

Exemple : après l'appel téléphonique (flux d'informations) du client ou à la réception de sa commande (représentation d'un flux matériel : le bon de commande) et à partir de 10 h (représentation du temps) on préparera l'enregistrement des commandes (l'acticité/tâche/opération)

• **l'événement résultat** est le résultat des traitements d'une tâche (activité ou opération) exemple : lors de l'enregistrement des commandes si les stocks de produits sont corrects, on éditera un bon de préparation de commande (résultat) dans le cas contraire, on avertira le client de la rupture de stocks (résultat).

: Symbole prenant en charge les données « Résultats » à passer au traitement.

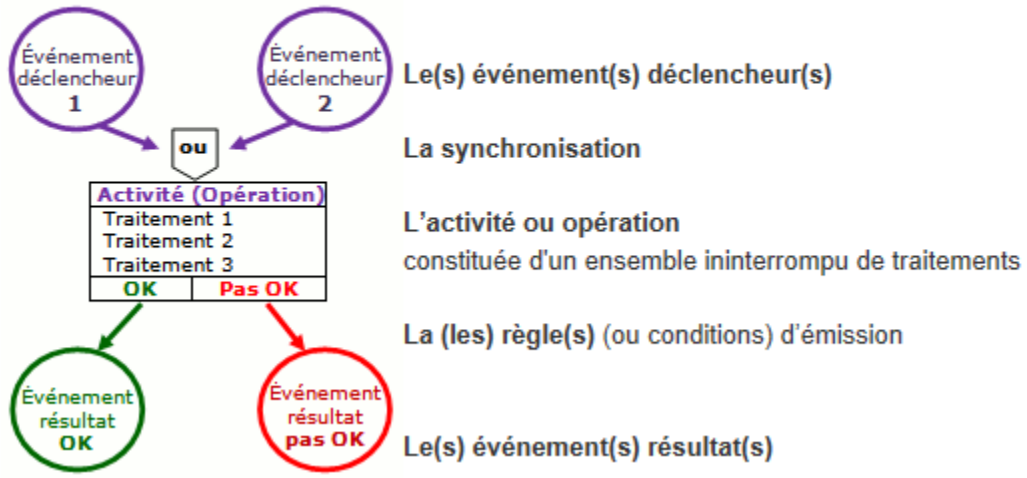
c. Les activités ou opérations

L'activité est une tâche (ou traitement) qui est gérée par un acteur interne et qui se déroule sans interruption (pas d'événements déclencheurs ou résultats intermédiaires).
Exemple : dans un processus « gestion des commandes clients » l'enregistrement de la commande d'un client est une activité (tâche ou opération).

3. **Le schéma événements/résultats**

a. Le formalisme

Schéma événements/résultats générique



d. La **synchronisation**

Est la combinaison d'événements qui va permettre **l'enclenchement d'une activité** (ou opération). La synchronisation est représentée par une **expression conditionnelle** sur les événements intervenant dans le déclenchement de l'activité (ou l'opération) ; le résultat est booléen (vrai/faux).

e. La **règle** (ou condition) d'émission
Est **la condition** qui détermine les événements résultats d'une activité (ou d'une opération), quand il n'y a pas de condition, on indique ***toujours*** ou ***rien***.

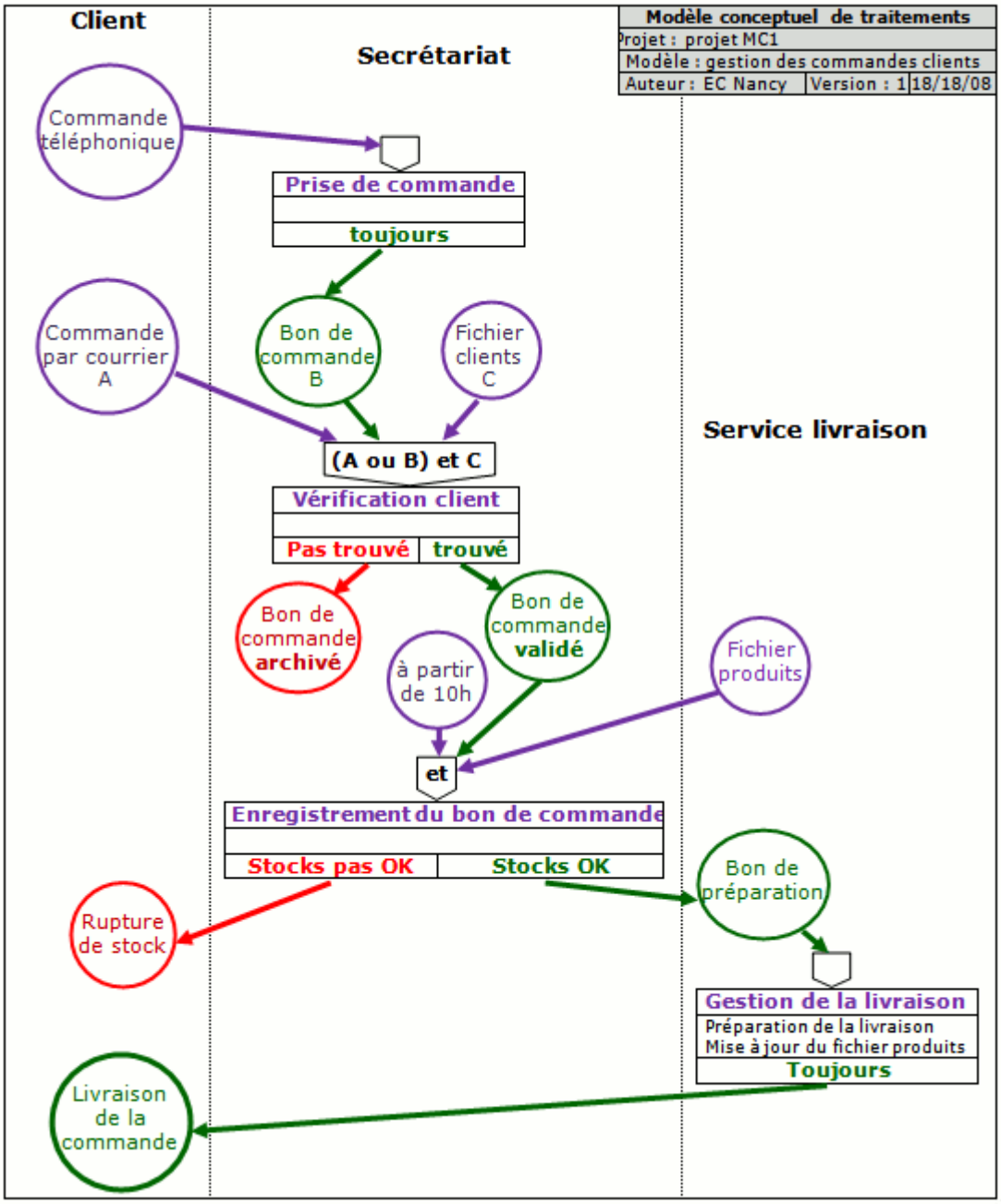
: Symbole d'une condition qui connecte le processus à un point donné.

Remarque : toute activité possède au moins **un événement déclencheur** et **un événement résultat.**
F. Illustration
• **Représentation textuelle du processus**
On veut gérer les commandes clients à partir des informations collectées suivantes : Un client peut passer une commande par téléphone auprès du secrétariat qui remplit alors un bon de commande ou l'envoyer par courrier. Le secrétariat vérifie par la suite de l'existence du client à partir du fichier des clients, si le client n'existe pas, on archive le bon de commande sinon on le garde pour enregistrement.

À partir de 10 h, le secrétariat enregistre les bons de commande client en vérifiant les stocks à partir du fichier des produits, si le stock est suffisant, on édite un bon de préparation qui est transmis au service livraison, dans le cas contraire, on avertit le client de la rupture de stock. À la réception d'un bon de préparation, le service livraison **prépare la commande**, met à jour des fichiers produits et livre le client.

• **Analyse de l'information**
- Le **processus** : la gestion des commandes clients
- Les **acteurs** : le client, le secrétariat, le service livraison
- Les **activités** (ou opérations) : la prise de commande par téléphone, l'enregistrement de la commande, la livraison

• Représentation finale graphique (schéma événements/résultats)



On note que **l'acteur client** ne possède aucune activité il est donc **externe**, les **acteurs secrétariat** et **service livraison** sont **internes**

L'essentiel

La modélisation d'un processus correspond à la représentation par le schéma événements/résultats des acteurs et des activités (tâches/opérations) qu'ils accomplissent dans le cadre du déroulement du processus. Elle permet une vision synthétique compréhensible par tous les acteurs de l'enchaînement des activités (tâches/opérations).

L'objectif de la modélisation des processus est de permettre l'observation de l'existant et l'évolution adaptative des traitements pour satisfaire les besoins des clients.
Deux méthodes de modélisation peuvent être retenues dans le cadre de ce cours :
- Modélisation UML (Unified Modelling Language) : est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé en développement logiciel et en conception orientée objet.
- Modélisation MERISE : Est une méthode d'analyse, de conception et de gestion de projet informatique.
Dans le cadre de ce cours, nous laisserons de côté la modélisation UML, propre aux professionnels du métier informatique, nous prendrons la modélisation MERISE qui se présente beaucoup plus pratique dans le cadre de la formation d'un gestionnaire.

4. MODELISATION MERISE

Il est difficile de modéliser un domaine sous une forme directement utilisable par un système informatique. Une ou plusieurs modélisations intermédiaires sont donc utiles, le modèle entités-associations constitue l'une des premières et des plus courantes. Ce modèle permet une description naturelle du monde réel à partir des concepts d'entité et d'association. Basé sur la théorie des ensembles et des relations, ce modèle se veut universel et répond à l'objectif d'indépendance données-programmes. Ce modèle, utilisé pour la phase de conception, s'inscrit notamment dans le cadre d'une méthode plus générale et très répandue : *Merise*.

La conception d'un système d'information n'est pas évidente car il faut réfléchir à l'ensemble de l'organisation que l'on doit mettre en place. La phase de conception nécessite des méthodes permettant de mettre en place un modèle sur lequel on va s'appuyer. La modélisation consiste à créer une représentation virtuelle d'une réalité de telle façon à faire ressortir les points auxquels on s'intéresse. Ce type de méthode est appelé *analyse*. Il existe plusieurs méthodes d'analyse, la méthode la plus utilisée en France étant la méthode **MERISE**.

4.1 Présentation de la méthode MERISE

MERISE (Méthode d'Étude et de Réalisation Informatique pour les Systèmes d'Entreprise) est certainement le langage de spécification le plus répandu dans la communauté de l'informatique des systèmes d'information, et plus particulièrement dans le domaine des bases de données. Une représentation Merise permet de valider des choix par rapport aux objectifs, de quantifier les solutions retenues, de mettre en œuvre des techniques d'optimisation et enfin de guider jusqu'à l'implémentation. Reconnu comme standard, Merise devient un outil de communication. En effet, Merise réussit le compromis difficile entre le souci d'une modélisation précise et formelle, et la capacité d'offrir un outil et un moyen de communication accessible aux non-informaticiens.

Un des concepts clés de la méthode Merise est la séparation des données et des traitements. Cette méthode est donc parfaitement adaptée à la modélisation des problèmes abordés d'un point de vue fonctionnel. Les données représentent la *statique* du système d'information et les traitements sa *dynamique*. L'expression conceptuelle des données conduit à une modélisation des données en **entités et en associations**. Dans ce cours, nous écartons volontairement la modélisation des

traitements puisque nous ne nous intéressons à la méthode Merise que dans la perspective de la modélisation du système informatique.

Merise propose une démarche, dite par niveaux, dans laquelle il s'agit de hiérarchiser les préoccupations de modélisation qui sont de trois ordres : la conception, l'organisation et la technique. En effet, pour aborder la modélisation d'un système, il convient de l'analyser en premier lieu de façon globale et de se concentrer sur sa fonction : c'est-à-dire de s'interroger sur ce qu'il fait avant de définir comment il le fait. Ces niveaux de modélisation sont organisés dans une double approche données/traitements. Les trois niveaux de représentation des données, puisque ce sont eux qui nous intéressent, sont détaillés ci-dessous :

Niveau conceptuel :

Le *modèle conceptuel des données (MCD)* décrit les entités du monde réel, en termes d'objets, de propriétés et de relations, indépendamment de toute technique d'organisation et d'implantation des données. Ce modèle se concrétise par un *schéma entités-associations* représentant la structure du système d'information, du point de vue des données.

Niveau logique :

Le *modèle logique des données (MLD)* précise le modèle conceptuel par des choix organisationnels. Il s'agit d'une transcription (également appelée dérivation) du MCD dans un formalisme adapté à une implémentation ultérieure, au niveau physique, sous forme de base de données relationnelle ou réseau, ou autres. Les choix techniques d'implémentation (choix d'un SGBD) ne seront effectués qu'au niveau suivant.

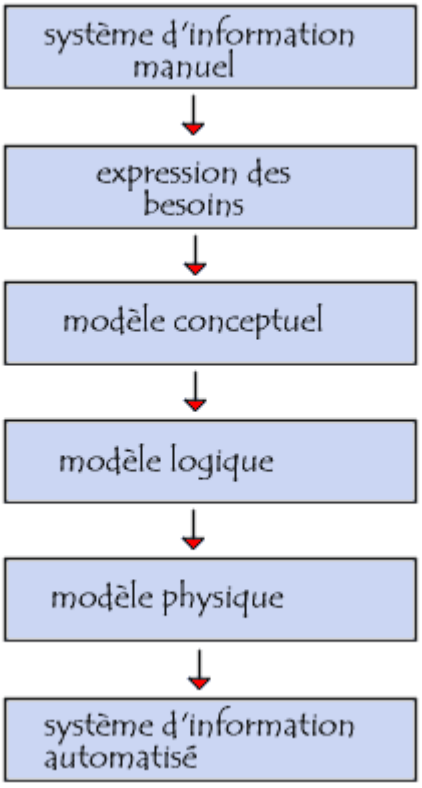
Niveau physique :

Le *modèle physique des données (MPD)* permet d'établir la manière concrète dont le système sera mis en place.

4.2 Cycle d'abstraction de conception des systèmes d'information

La conception du système d'information se fait par étapes, afin d'aboutir à un système d'information fonctionnel reflétant une réalité physique. Il s'agit donc de valider une à une chacune des étapes en prenant en compte les résultats de la phase précédente. D'autre part, les données étant séparées des traitements, il faut vérifier la concordance entre données et traitements afin de vérifier que toutes les données nécessaires aux traitements sont présentes et qu'il n'y a pas de données superflues.

Cette succession d'étapes est appelée *cycle d'abstraction pour la conception des systèmes d'information* :



4.3 Éléments constitutifs du modèle entités-associations

La représentation du modèle entités-associations s'appuie sur trois concepts de base :

- L'objet ou entité,
- L'association,
- la propriété.

L'objet est une entité ayant une existence propre. L'association est un lien ou relation entre objets sans existence propre. La propriété est la plus petite donnée d'information décrivant un objet ou une association.

4.3.1 Entité

Figure 2.1: Représentation graphique d'un exemple de type-entité.

Définition 1 -***entité***- *Une entité est un objet, une chose concrète ou abstraite qui peut être reconnue distinctement et qui est caractérisée par son unicité.*

Exemples d'entité : Michelle, Joseph André, le livre que je tiens entre les mains, la Mercedes qui se trouve dans mon garage, etc. Les entités ne sont généralement pas représentées graphiquement.

4.3.2 Attribut ou propriété, valeur

Figure 2.2: Représentation graphique d'un exemple de type-entité comportant trois attributs

Définition 3 -***attribut, propriété***- *Un attribut (ou une propriété) est une caractéristique associée à un type-entité ou à un type-association.*

Exemples d'attribut : le nom d'une personne, le titre d'un livre, la puissance d'une voiture.

Définition 4 -***valeur***- *Au niveau du type-entité ou du type-association, chaque attribut possède un domaine qui définit l'ensemble des valeurs possibles qui peuvent être choisies pour lui (entier, chaîne de caractères, booléen). Au niveau de l'entité, chaque attribut possède une* ***valeur*** *compatible avec son domaine.*

La figure 2.2 montre la représentation graphique d'un exemple de type-entité (*Personne*) avec trois attributs.

Règle 1 *Un attribut ne peut en aucun cas être partagé par plusieurs type-entités ou type-associations.*
Règle 2 *Un attribut est une donnée élémentaire, ce qui exclut des données calculées ou dérivées.*
Règle 3 *Un type-entité et ses attributs doivent être cohérents entre eux (*i.e. *ne traiter que d'un seul sujet).*

Par exemple, si le modèle doit comporter des informations relatives à des articles et à leur fournisseur, ces informations ne doivent pas coexister au sein d'un même type-entité. Il est préférable de mettre les informations relatives aux articles dans un type-entité *Article* et les informations relatives aux fournisseurs dans un type-entité *Fournisseur*. Ces deux type-entités seront probablement ensuite reliés par un type-association.

4.3.3 Identifiant ou clé

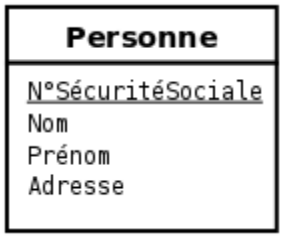


Figure 2.3: Représentation graphique d'un exemple de type-entité comportant quatre attributs dont un est un identifiant : deux personnes peuvent avoir le même nom, le même prénom et le même âge, mais pas le même numéro de sécurité sociale.

Définition 8 -***identifiant, clé***- *Un identifiant (ou clé) d'un type-entité ou d'un type-association est constitué par un ou plusieurs de ses attributs qui doivent avoir une valeur unique pour chaque entité ou association de ce type.*

Il est donc impossible que les attributs constituant l'identifiant d'un type-entité (respectivement type-association) prennent la même valeur pour deux entités (respectivement deux associations) distinctes. Exemples d'identifiant : le numéro de sécurité sociale pour une personne, le numéro d'immatriculation pour une voiture, le N° de téléphone pour individu d'une société connue.

Règle 4 *Chaque type-entité possède au moins un identifiant, éventuellement formé de plusieurs attributs.*

Ainsi, chaque type-entité possède au moins un attribut qui, s'il est seul, est donc forcément l'identifiant.

Dans la représentation graphique, les attributs qui constituent l'identifiant sont soulignés et placés en tête (cf. figure 2.3).

4.3.4 Association ou relation

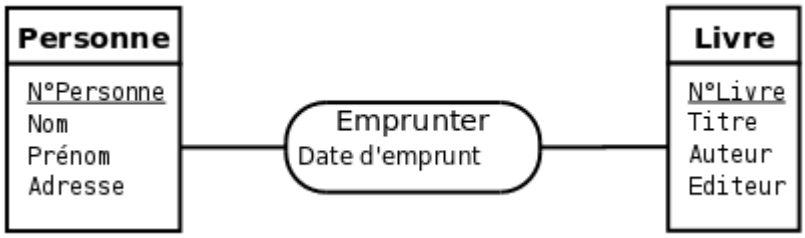


Figure 2.4: Représentation graphique d'un exemple de type-association liant deux type-entités.

Définition 10 -***association***- *Une association (ou une relation) est un lien entre plusieurs entités.*

Exemples d'association : l'emprunt par l'étudiante Gabrielle du 3ème exemplaire du livre «Mon SI ».

Les associations ne sont généralement pas représentées graphiquement.

Définition 11 -***type-association***- *Un type-association (ou un type-relation) désigne un ensemble de relations qui possèdent les mêmes caractéristiques. Le type-association décrit un lien entre plusieurs type-entités. Les associations de ce type-association lient des entités de ces type-entités.*

Comme les type-entités, les type-associations sont définis à l'aide d'attributs qui prennent leur valeur dans les associations.

Règle 12 *Un attribut peut être placé dans un type-association uniquement lorsqu'il dépend de toutes les entités liées par le type-association.*

Un type-association peut ne pas posséder d'attribut explicite et cela est relativement fréquent, mais on verra qu'il possède au moins des attributs implicites.

Exemples de type-association : l'emprunt d'un livre à la bibliothèque.

Une association est souvent nommée occurrence ou instance de son type-association.

La figure 2.4 montre la représentation graphique d'un exemple de type-association.

Par abus de langage, on utilise souvent le mot association en lieu et place du mot type-association, il faut cependant prendre garde à ne pas confondre les deux concepts.

Définition 13 -***participant***- *Les type-entités intervenant dans un type-association sont appelés les participants de ce type-association.*
Définition 14 -***collection***- *L'ensemble des participants d'un type-association est appelé la collection de ce type-association.*

Cette collection comporte au moins un type-entité, mais elle peut en contenir plus, on parle alors de type-association *n-aire* (quand *n*=2 on parle de type-association binaire, quand *n*=3 de type-association ternaire, etc.).

4.3.5 Cardinalité

Figure 2.5: Représentation graphique des cardinalités d'un type-association. Dans cet exemple pédagogique, on suppose qu'un livre ne peut posséder qu'un auteur.

Définition 17 -***cardinalité***- *La cardinalité d'une patte reliant un type-association et un type-entité précise le nombre de fois minimal et maximal d'interventions d'une entité dans une association. La cardinalité minimale doit être inférieure ou égale à la cardinalité maximale.*

Exemple de cardinalité : une personne peut être l'auteur de 0 à *n* livre, mais un livre ne peut être écrit que par une personne (cf. figure 2.5).

Règle 18 *L'expression de la cardinalité est obligatoire pour chaque patte d'un type-association.*
Règle 19 *Une cardinalité minimal est toujours 0 ou 1 et une cardinalité maximale est toujours 1 ou n.*

Ainsi, si une cardinalité maximale est connue et vaut 2, 3 ou plus, alors nous considérons qu'elle est indéterminée et vaut *n*. En effet, si nous connaissons *n* au moment de la conception, il se peut que cette valeur évolue au cours du temps. Il vaut donc mieux considérer *n* comme inconnue dès le départ. De la même manière, on ne modélise pas des cardinalités minimales qui valent plus de 1 car ces valeurs sont également susceptibles d'évoluer. Enfin, une cardinalité maximale de 0 n'a pas de sens car elle rendrait le type-association inutile.

Les seuls cardinalités admises sont donc :

0,1 : Une occurrence du type-entité peut exister tout en étant impliquée dans aucune association et peut être impliquée dans au maximum une association.
0,n : C'est la cardinalité la plus ouverte ; une occurrence du type-entité peut exister tout en étant impliquée dans aucune association et peut être impliquée, sans limitation, dans plusieurs associations.
1,1 : Une occurrence du type-entité ne peut exister que si elle est impliquée dans exactement (au moins et au plus) une association.
1,n : Une occurrence du type-entité ne peut exister que si elle est impliquée dans au moins une association.

Une cardinalité minimale de 1 doit se justifier par le fait que les entités du type-entité en questions ont besoin de l'association pour exister. Dans tous les autres cas, la cardinalité minimale vaut 0. Ceci dit, la discussion autour d'une cardinalité minimale de 0 ou de 1 n'est intéressante que lorsque la cardinalité maximale est 1. En effet, lors de la traduction vers un schéma relationnel non prévu dans le cadre de ce cours, lorsque la cardinalité maximale est *n*, nous ne ferons pas la différence entre une cardinalité minimale de 0 ou de 1.

Remarques

La seule difficulté pour établir correctement les cardinalités est de se poser la question dans le bon sens. Pour augmenter le risque d'erreurs, il faut noter que, pour les habitués, ou les futurs habitués, du modèle UML, les cardinalités d'un type-association sont « à l'envers » (par référence à UML) pour les type-associations binaires et « à l'endroit » pour les n-aires avec $n>2$.

La notion de cardinalité n'est pas définie de la même manière dans le modèle Américain et dans le modèle Européen (Merise). Dans le premier n'existe que la notion de cardinalité maximale.

4.4 Règles de bonne formation d'un modèle entités-associations

La *bonne formation* d'un modèle entités-associations permet d'éviter une grande partie des sources d'incohérences et de redondance. Pour être bien formé, un modèle entités-associations doit respecter certaines règles et les type-entités et type-associations doivent être normalisées. Un bon principe de conception peut être formulé ainsi : **« une seule place pour chaque fait »**.

Bien que l'objectif des principes exposés dans cette section soit d'aider le concepteur à obtenir un diagramme entités-associations bien formé, ces principes ne doivent pas être interprétés comme des lois. Qu'il s'agisse des règles de bonne formation ou des règles de normalisation, il peut exister, très occasionnellement, de bonnes raisons pour ne pas les appliquer.

4.4.1 Règles portant sur les noms

Règle 21 *Dans un modèle entités-associations, le nom d'un type-entité, d'un type-association ou d'un attribut doit être unique.*

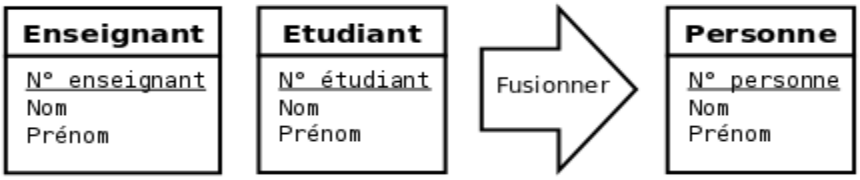


Figure 2.15: La présence des deux type-entités *Enseignant* et *Etudiant* est symptomatique d'une modélisation inachevée. A terme, ces deux type-entités doivent être fusionnés en un unique type-entité *Personne*. Référez-vous à la règle 25 pour plus de précisions concernant cette erreur de modélisation.

Figure 2.16: Ici, les attributs *Adresse de facturation* sont redondants. Cette situation doit être évitée à tout prix car elle entraîne un gaspillage d'espace mémoire mais aussi et **surtout un grand risque d'incohérence**. En effet, que faire si, dans le cadre d'une occurrence du type-association *Correspondre*, les valeurs des deux attributs *Adresse de facturation* diffèrent ?

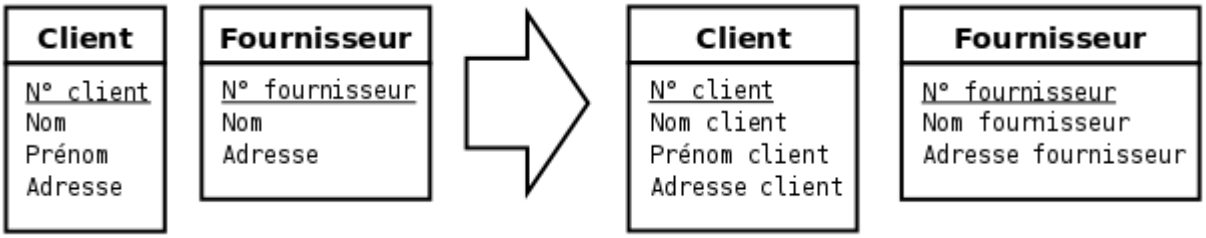


Figure 2.17: Dans cette situation, les deux attributs *Adresse* doivent simplement être renommés en *Adresse client* et *Adresse fournisseur*. Il en va de même pour les deux attributs *Nom*.

Lorsque des attributs portent le même nom, c'est parfois le signe d'une modélisation inachevée (figure 2.15) ou d'une redondance (figure 2.16). Sinon, il faut simplement ajouter au nom de l'attribut le nom du type-entité ou du type-association dans lequel il se trouve (figure 2.17). Il faut toutefois remarquer que le dernier cas décrit n'est pas rédhibitoire et que le système informatique s'accommode très bien de relations comportant des attributs de même nom. L'écriture des requêtes sera tout de même plus lisible si les attributs ont tous des noms différents.

4.4.2 Normalisation des type-entités et type-associations

Introduction

Les formes normales sont différentes de qualité afin d'éviter la redondance, source d'anomalies. La normalisation peut être aussi bien effectuée sur un modèle entités-associations, où elle s'applique sur les type-entités et type-associations, que sur un modèle relationnel.

Il existe 5 formes normales[4] principales et deux extensions. Plus le niveau de normalisation est élevé, plus le modèle est exempte de redondances. Une modélisation rigoureuse permet généralement d'aboutir directement à des type-entités et type-associations en forme normale de Boyce-Codd.

Première forme normale (1FN)

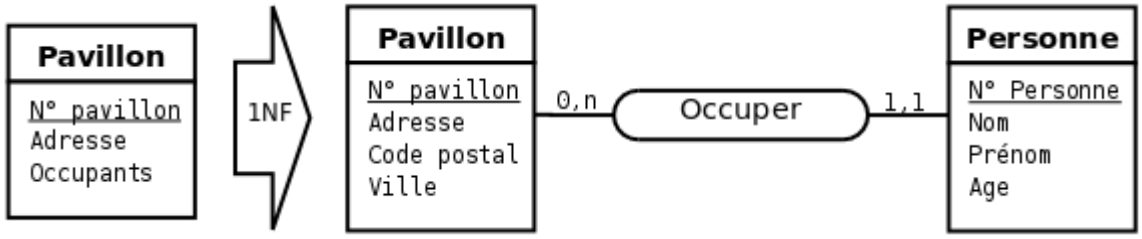


Figure 2.30: Exemple de normalisation en première forme normale.

4 Dans le cadre de ce cours nous ne présenterons que les 3 formes normales utiles à l'implémentation d'un système informatique.

Définition 30 -***Première forme normale (1FN)***- *Un type-entité ou un type-association est en première forme normale si tous ses attributs sont élémentaires, c'est-à-dire non décomposables.*

Un attribut composite doit être décomposé en attributs élémentaires (comme l'attribut *Adresse* sur la figure 2.30) ou faire l'objet d'une entité supplémentaire (comme l'attribut *Occupants* sur la figure 2.30.)

L'*élémentarité* d'un attribut est toutefois fonction des choix de gestion. Par exemple, la propriété *Adresse* peut être considérée comme élémentaire si la gestion de ces adresses est globale. Par contre, s'il faut pouvoir considérer les codes postaux, les noms de rues, il convient d'éclater la propriété *Adresse* en *Adresse* (au sens numéro d'appartement, numéro et nom de rue,), *Code postal* et *Ville*. En cas de doute, il est préférable (car plus général) d'éclater une propriété que les regrouper.

Deuxième forme normale (2FN)

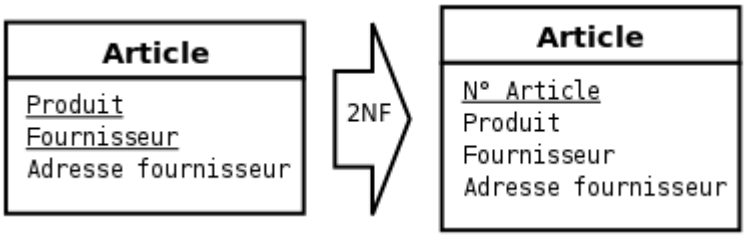


Figure 2.31: Exemple de normalisation en deuxième forme normale. On suppose qu'un même fournisseur peut fournir plusieurs produits et qu'un même produit peut être fourni par différents fournisseurs.

Définition 31 -***Deuxième forme normale (2FN)***- *Un type-entité ou un type-association est en deuxième forme normale si, et seulement si, il est en première forme normale et si tout attribut n'appartenant pas à la clé dépend de la totalité de cette clé.*

Autrement dit, les attributs doivent dépendre de l'ensemble des attributs participant à la clé. Ainsi, si la clé est réduite à un seul attribut, ou si elle contient tous les attributs, le type-entité ou le type-association est, par définition, forcément en deuxième forme normale.

La figure 2.31 montre un type-entité *Article* décrivant des produits provenant de différents fournisseurs. On suppose qu'un même fournisseur peut fournir plusieurs produits et qu'un même produit peut être fourni par différents fournisseurs. Dans ce cas, les attributs *Produit* ou *Fournisseur* ne peuvent constituer un identifiant du type-entité *Article*. Par contre, le couple *Produit/Fournisseur* constitue bien un identifiant du type-entité *Article*. Cependant, l'attribut *Adresse fournisseur* ne dépend maintenant que d'une partie de la clé (*Fournisseur*). Opter pour une nouvelle clé arbitraire réduite à un seul attribut *N° article* permet d'obtenir un type-entité *Article* en deuxième forme normale. On va voir dans ce qui suit que cette solution n'a fait que déplacer le problème.

Troisième forme normale (3FN)

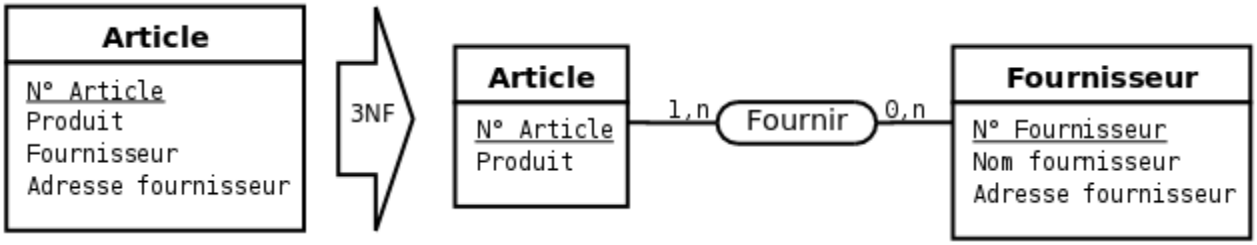


Figure 2.32: Exemple de normalisation en troisième forme normale. Dans cet exemple, l'attribut *Adresse fournisseur* dépend de l'attribut *Fournisseur*.

Définition 32 -***Troisième forme normale (3FN)***- *Un type-entité ou un type-association est en troisième forme normale si, et seulement si, il est en deuxième forme normale et si tous ses attributs dépendent directement de sa clé et pas D'autres attributs.*

Cette normalisation peut amener à dés imbriquer des type-entités cachées comme le montre la figure 2.32.

Un type-entité ou un type-association en deuxième forme normale avec au plus un attribut qui n'appartient pas à la clé est, par définition, forcément en troisième forme normale.

4.5 Élaboration d'un modèle entités-associations

4.5.1 Étapes de conceptions d'un modèle entités-associations

Pour concevoir un modèle entités-associations, vous devrez certainement passer par une succession d'étapes. Nous les décrivons ci-dessous dans l'ordre chronologique. Sachez cependant que la conception d'un modèle entités-associations est un travail non linéaire. Vous devrez régulièrement revenir à une étape précédente et vous n'avez pas besoin d'en avoir terminé avec une étape pour commencer l'étape suivante.

Recueil des besoins
C'est une étape primordiale. Inventoriez l'ensemble des données à partir des documents de l'entreprise, d'un éventuel cahier des charges et plus généralement de tous les supports de l'information. N'hésitez pas à poser des questions.

Tri de l'information
Faites le tri dans les données recueillies. Il faut faire attention, à ce niveau, aux problèmes de synonymie/polysémie. En effet, les attributs ne doivent pas être redondants. Par exemple, si dans le langage de l'entreprise on peut parler indifféremment de *référence d'article* ou de *n° de produit* pour désigner la même chose, cette caractéristique ne devra se concrétiser que par un unique attribut dans le modèle. Inversement, on peut parler d'adresse pour désigner l'adresse du fournisseur et l'adresse du client, le contexte permettant de lever l'ambiguïté. Par contre, dans le modèle, il faudra veiller à bien distinguer ces deux caractéristiques par deux attributs distincts.

Un autre exemple est celui d'une entreprise de production fabricant des produits à destination d'une autre société du même groupe. Il se peut que dans ce cas, le prix de production (*i.e.* le coût de revient industriel) soit le même que prix de vente (aucune marge n'est réalisée). Même dans ce cas où les deux caractéristiques sont identiques pour chaque entité (prix de production égale prix de vente), il faut impérativement les scinder en deux attributs au niveau du type-entité

Produit. Sinon, cette égalité factuelle deviendrait une contrainte imposée par le modèle, obligeant alors l'entreprise de production à revoir son système le jour où elle décidera de réaliser une marge (prix de production inférieure au prix de vente).

Identification des type-entités
Le repérage d'attributs pouvant servir d'identifiant permet souvent de repérer un type-entité. Les attributs de ce type-entité sont alors les attributs qui dépendent des attributs pouvant servir d'identifiant.

Attention, un même concept du monde réel peut être représenté dans certains cas comme un attribut et dans D'autres cas comme un type-entité, selon qu'il a ou non une existence propre. Par exemple, la marque d'une automobile peut être vue comme un attribut du type-entité *Véhicule* de la base de données d'une préfecture mais aussi comme un type-entité *Constructeur automobile* dans la base de données du Ministère de l'Industrie.

Lorsqu'on ne parvient pas à trouver d'identifiant pour un type-entité, il faut se demander s'il ne s'agit pas en fait d'un type-association. Si ce n'est pas le cas, un identifiant arbitraire numérique entier peut faire l'affaire.

Identification des type-associations
Identifiez les type-associations reliant les type-entités du modèle. Le cas échéant, leur affecter les attributs correspondant.

Il est parfois difficile de faire un choix entre un type-entité et un type-association. Par exemple, un mariage peut être considéré comme un type-association entre deux personnes ou comme un type-entité pour lequel on veut conserver un numéro, une date, un lieu, et que l'on souhaite manipuler en tant que tel.

Étudiez également les cardinalités des type-associations retenus. Lorsque toutes les pattes d'un type-association portent la cardinalité 1,1, il faut se demander si ce type-association et les type-entités liés ne décrivent pas en fait un seul type-entité (cf. règle 2.25).

Vérification du modèle
Vérifiez que le modèle respecte bien les règles que nous avons énoncés et les définitions concernant la normalisation des type-entités et des type-associations. Le cas échéant, opérez les modifications nécessaires pour que le modèle soit bien formé.

Remarque :

Pour faciliter la lecture du schéma, il est assez courant de ne pas y faire figurer les attributs ou de ne conserver que ceux qui font partie des identifiants. Les attributs cachés doivent alors absolument être spécifiés dans un document à part.

4.5.2 Conseils divers

Concernant le choix des noms

Pour les type-entités, choisissez un nom commun décrivant le type-entité (ex : Étudiant, Enseignant, Matière). Certain préfèrent mettre le nom au pluriel (ex : Étudiants, Enseignants, Matières). Restez cependant cohérents, soit tous les noms de type-entité sont au pluriel, soit ils sont tous au singulier.

Pour les type-association, choisissez un verbe à l'infinitif, éventuellement à la forme passive ou accompagné d'un adverbe (ex : Enseigner, Avoir lieu dans).

Pour les attributs, utilisez un nom commun au singulier éventuellement accompagné du nom du type-entité ou du type-association dans lequel il se trouve (ex : nom de client, numéro d'article).

Concernant le choix des identifiants des type-entités

Évitez les identifiants composés de plusieurs attributs (comme, par exemple, un identifiant formé par les attributs *nom* et *prénom* d'un type-association *Personne*) car :

- ils dégradent les performances du système informatique,
- mais surtout l'unicité supposée par une telle démarche finit généralement, tôt ou tard, par être démentie !

Évitez les identifiants susceptibles de changer au cours du temps (comme la plaque d'immatriculation d'un véhicule). Évitez les identifiants du type chaîne de caractère. En fait, il est souvent préférable de choisir un identifiant arbitraire de type entier pour les type-entités. Cet identifiant deviendra une clé primaire dans le schéma relationnel et le SGBD l'incrémentera automatiquement lors de la création de nouvelles instances. L'inconvénient de cette pratique est qu'il devient possible de se retrouver avec deux instances du type-entités représentant le même objet mais avec deux numéros différents. Malgré cette inconvénient, cette politique de l'identifiant reste largement avantageuse dans la pratique et permet, en outre, de s'affranchir (en la satisfaisant automatiquement) de la deuxième forme normale.

Bien distinguer les concepts de données et de traitements

La modélisation conceptuelle des données exclut la représentation des traitements futurs sur ces données. Toutefois, elle nécessite la connaissance de ces traitements pour prévoir les données élémentaires indispensables à ceux-ci. En conséquence, il existe une confusion fréquente entre les concepts de données et de traitements. Par exemple, la facturation est un traitement qui nécessite de connaître toutes les caractéristiques d'une commande. Par contre, la facturation ne se traduit ni par un type-entité, ni par un type-association dans le schéma entités-associations.

CHAP III. GESTION DES PROJETS

III.1 Introduction

Un projet est un ensemble de tâches **indissociables** permettant de répondre à un **besoin exprimé.** Il comprend également les **ressources** nécessaires à sa réalisation ; Il a une **durée finie**, caractérisée par une date de début et une date de fin ; Il peut être multi technique, mono technique, collectif ou individuel

Nous reconnaitrons trois principes fondamentaux :

- **Instauration systématique d'une phase d'exploration en amont des projets**
  - Evaluer l'opportunité du projet
  - Préciser ses objectifs de coût, de délai et de performance
  - Identifier les marges de manœuvre : coût, délais, volume
- **Maîtrise d'œuvre responsable nommément désignée**
  - Mobiliser et coordonner l'ensemble des compétences nécessaires
- **Maîtrise d'ouvrage forte animée par un pilote**

- Fixer des objectifs clairs
- Réorienter si nécessaire

III. 2 Les phases de la vie

- **La pré-étude**

Elle permet au travers d'une identification première des actions, des acteurs, des coûts, des charges et des gains de toutes natures, de bâtir un dossier qui contribue à l'étude d'opportunité et de faisabilité ; Une pré-étude n'est jamais perdue, quelles qu'en soient les suites

-

- **L'étude**
  - Une fois le projet retenu, elle conduit à figer de manière précis les contours du projet
- **La réalisation**
  - Elle s'accompagne d'un suivi et d'un bilan de réalisation

- **L'exploitation**
  - idem mais avec des rapports d'étape
- **Le désinvestissement**
  - Il peut faire partie de la pré-étude du projet suivant

Idée

Autorisable
Autorisé
Pré-étude
Arbitrage ?
Lancement ?

Orange
Rejeté

III. 3 Les acteurs concernés

le chef de projet : Il fédère l'équipe projet, il est en mesure à tous les stades d'informer le comité de pilotage, Il informe également ses partenaires, et dispose de moyens et d'obligations de tenir ses objectifs.

Ses missions principales sont :

- **Définition du projet**
- **Planification du projet**
- **Pilotage du projet**
- **Négociations internes et externes au projet**
- **Animation des équipes**
- **Reporting interne et externe**
- **Gestion du fond documentaire**

Il doit avoir le sens :

Imagination, raisonnement, Savoir-faire, Expertise, Curiosité, Sensibilité, Ecoute, Ouverture d'esprit, communication, relationnel, motivation, influence, solidarité, responsabilité, synthèse.
Confiance, Créativité, Méthodologie, Initiative ; Capacité à défendre une idée ; Capacité d'interpellation.

III. 4 Le dossier de la pré-étude

Il présente les avantages suivants :

- **Un instrument de réflexion et de communication**
  - Décloisonner, se poser les bonnes questions
- **Permettant de définir le contenu du projet**
  - Les actions, mais aussi les partenaires ou contributeurs

- Les marges de manœuvre pendant qu'elles donnent encore un degré de latitude. Il s'agira plus tard de gérer des contraintes...

- **Facilitant la validation de ce contenu**
  - Arbitrage, évaluation à partir de l'urgence, du retour financier et du retour opérationnel, éclairage sur les risques
  - Lettre de mission ... etc.
- **Préparant la mise en place et la structuration du projet**
  - Optimisation avant mise en œuvre

III. 5 le management du projet

- **L'inadéquation de l'environnement et de l'organisation**
  - Des structures organisationnelles stables
  - Des produits de plus en plus fluides
- **Garantir l'atteinte d'un objectif en réalisant le meilleur compromis par la bonne gestion des contraintes :**
  - Qualité technique
  - Délais
  - Coûts
- **Favoriser l'atteinte des résultats attendus**
  - Par l'optimisation de la planification et des moyens
  - Par l'optimisation de l'organisation
- **Les dix commandements du fonctionnement en projet**
  - 1 Le maître d'ouvrage fixe les orientations générales du projet et avalise toute modification de ses objectifs
  - 2 Le maître d'ouvrage est garant de la conformité du projet aux objectifs stratégiques et de sa rentabilité
  - 3 La maîtrise d'ouvrage et la maîtrise d'œuvre doivent être séparées
  - 4 Le maître d'œuvre est garant de la tenue des objectifs du projet (délais, coûts, qualité)
  - 5 Le maître d'œuvre détermine le découpage de son projet en sous-projets avec des résultats intermédiaires et des dates-clés
  - 6 Le maître d'œuvre fait résoudre les problèmes à tous les niveaux
  - 7 Chaque service contributeur est responsable vis-à-vis du maître d'oeuvre des objectifs de coûts, délais, performance
  - 8 Chaque contributeur informe systématiquement le maître d'œuvre
  - 9 Chaque contributeur communique sans contrainte avec les autres contributeurs du projet

- 10 Chaque responsable de centre de compétences garantit l'efficacité des moyens qu'il engage sur les projets

CHAP IV. La sécurité informatique

La sécurité informatique est l'ensemble des moyens techniques, organisationnels, juridiques et humains nécessaires et mis en place pour conserver, rétablir, et garantir la sécurité des systèmes informatiques. Elle est intrinsèquement liée à la sécurité de l'information et des systèmes d'information.

« La raison principale de l'existence de l'industrie de la sécurité informatique est que les produits et services informatiques ne sont pas naturellement sûrs. Si les ordinateurs étaient protégés des virus, il n'y aurait pas besoin de produits antivirus. Si le mauvais trafic réseau ne pouvait être utilisé pour attaquer les ordinateurs, personne ne s'inquiéterait d'acheter un pare-feu. S'il n'y avait plus de dépassement de tampon, personne n'aurait besoin d'acheter des produits pour se protéger contre leurs effets. Si les produits informatiques que nous achetons étaient sûrs par défaut, nous n'aurions pas besoin de dépenser des milliards chaque année pour les rendre plus sûrs. »

IV. 1 Aspects généraux de la sécurité informatique

IV.1 Les Menaces

L'ensemble des actions de l'environnement d'un système pouvant entraîner des pertes financières. les Menaces peuvent être regroupées en trois catégories

- Celles relevant de problèmes non spécifiques à l'informatique, tel l'incendie, l'explosion, l'inondation, la tempête, la foudre, le vol d'équipements matériels, la destruction d'équipements, la destruction de supports de sauvegarde, le départ du personnel, les grèves…
- Celles relevant de problèmes informatiques, tel les pannes et dysfonctionnements du matériel et ou logiciel de base ; oubli de sauvegarde, écrasement de fichiers, erreur de saisie, erreur de transmission, erreur d'utilisation.
- Celles qui sont intentionnelles : L'ensemble des actions malveillantes qui constituent la plus grosse partie du risque, qui devraient être l'objet principal des mesures de protection ; dans cette catégorie on en distingue deux :

1. **Menaces passives**

Détournement des données, l'écoute, les indiscrétions, (Exemples: espionnage industriel, espionnage commercial, violations déontologiques), détournement des logiciels (Exemple: copies illicites)

2. **Menaces actives**

Modifications des informations, (Exemple : La fraude financière, informatique, le sabotage des informations).

I.2 Exemples de menaces malveillantes à caractère informatique

I.2.1 Déguisement

Pour rentrer dans un système on essaye de piéger des usagers et de se faire prendre pour quelqu'un d'autre.

Exemple: simulation d'interface, système sur écran, simulation de terminal à carte bancaire

I.2.2 Répétition ("replay")

Espionnage d'une interface, d'une voie de communication (téléphonique, réseau local) pour capter des opérations (même cryptées elles peuvent être utilisables),

Exemple: Plusieurs fois la même opération d'accréditement d'un compte bancaire.

I.2.3 Analyse de trafic

On observe **le** trafic de messages échangés pour en déduire des informations sur les décisions de quelqu'un.

IV. 2.2 Politique de sécurité

2.1 Définition

Une politique de sécurité est un ensemble de règles qui fixent les actions autorisées et interdites dans le domaine de la sécurité.

2.2 Étapes types dans l'établissement d'une politique de sécurité

- Identification des vulnérabilités

En mode fonctionnement normal (définir tous les points faibles).

En cas d'apparition de défaillances un système fragilisé est en général vulnérable : c'est dans un de ces moments intermédiaires qu'une intrusion peut le plus facilement réussir.

- Évaluation des probabilités associées à chacune des menaces
- Évaluation du coût d'une intrusion réussie
- Choix des contre-mesures
- Évaluation des coûts des contre mesure
- Décision

IV.3 Les méthodologies de sécurité

Réalisées par des grands utilisateurs de techniques de sécurité ou des groupes de travail elles sont applicables par des prestataires de service sous forme d'audit de sécurité, analyse de risques et propositions d'actions pour améliorer la situation

3.1 Méthode M.A.R.I.O.N

Méthode d'Analyse des Risques Informatiques et Optimisation par Niveau. (à partir de 1984)

Norme : **CLUSIF**: Club des Utilisateurs de La Sécurité Informatique Français

APSAD: Assemblée Plénière des Sociétés d'Assurances Dommages

Objectif: Mettre en place le schéma directeur de la sécurité des systèmes d'information SDSSI

Trois approches selon le sujet traité:

- Marion-AP (avant-projet) (Applicable aux grands comptes et aux compagnies d'assurance)
- Marion-PME
- Marion-RSX (Applicable aux réseaux)

La Méthode Marion renseigne Les six étapes d'élaboration du Schéma Directeur de Sécurité du Système d'Information :

a) Analyse des risques

Établissement de scénarios de risques courus par l'entreprise.

b) Expression du risque maximum admissible

Calcul de la perte maximale subie par l'entreprise face à des événements mettant sa survie en péril

c) Analyse des moyens de la sécurité existants

Identifier et qualifier les moyens de la sécurité (organisation générale, physique et logique)

d) Évaluation des contraintes techniques et financières

Recensement des contraintes générales, techniques, humaines et détermination d'un budget pour la prévention et la protection

e) Choix des moyens de sécurité

Moyens à mettre en oeuvre ou à améliorer pour supprimer les risques en fonction des contraintes et du coût parade/risque

f) Plan d'orientation

Phase de bilan définissant le plan technique détaillé et rédaction finale du SDSSI.

3.2 Méthode M.E.L.I.S.A

Délégation générale à l'armement 1985.

MELISA S - Confidentialité des données sensibles

MELISA P - Pérennité de fonctionnement du système

MELISA M - Sécurité micro mini-informatique

MELISA. R - Sécurité réseau

CHAP V. La programmation

Introduction

Une des questions récurrentes est "Comment débuter en programmation ?" ou parfois "Quel langage pour débuter ?". Je vais tenter d'y répondre, mais il faut bien savoir que ce genre de question est susceptible de déclencher des guerres de chapelles. Pour essayer d'avoir quelque chose de relativement objectif, je me suis basé sur les opinions des membres recueillies aux travers des réponses accordées à ces questions.

Alors tout d'abord, il n'y a pas de langage pour débuter la programmation, du moins la programmation commence sur papier, avec l'algorithmique.
Évidemment, pour tester les algorithmes, il peut être confortable de le faire avec un langage de programmation, mais dans ce cas, la seule chose que je recommanderai est un langage simple, non objet, non graphique. Les bons choix peuvent être Basic, Pascal, Python, néanmoins la syntaxe de ces derniers sera sans doute plus utile par la suite. Mais rien n'est vraiment à exclure ici, même le langage de calculatrice peut suffire à tester les algos simples. Ensuite, il est bon d'avoir une idée sur les différents type de langages, sur ce qu'est un compilateur et autres choses généralistes.

Lorsqu' on a déjà des bases saines pour commencer à programmer, un but, une idée, un logiciel à faire. Là, tout est fonction de ses gouts, de ses connaissances,... et là s'effectue les premiers choix techniques:
Langage objet, oui ou non ?
Multi plateforme, oui ou non ?
Avec machine virtuelle, oui ou non ?
Interface utilisateur, oui ou non ?
Accès à une base de données? Si oui laquelle?
…
Une fois ces choix faits, l'éventail des langages se retrouve singulièrement restreint. Alors passons en revue les langages les plus souvent cités pour débuter, avec toujours en tête le point de vue de l'apprentissage.

Ceux que l'on peut utiliser :

Delphi/Pascal

Une bonne solution, les compilateurs et environnements de développements se trouvent gratuitement (sous certaines conditions peu contraignantes). On peut commencer par faire des petits programmes dans des Shell en Pascal et ensuite passer (après avoir appris la notion d'objet) à des choses plus costaudes avec environnement graphique grâce à Delphi (et Kylix sous Linux). Le langage est relativement intuitif, rigoureux, la création d'interface graphique simple, on peut

vite se faire plaisir. La portabilité est médiocre mais existe vers certaines plateformes (Kylix...)

Java

Langage objet, très portable, syntaxe utilisée également pour des scripts internet (la syntaxe, pas le langage...). Il existe de nombreux environnements de développement gratuits sur le net : il faut juste choisir en fonction de ses goûts, néanmoins, ça n'a pas la facilité de Delphi au niveau des interfaces graphiques. Il s'agit là d'un langage "nouvelle génération" qui donnera aussi une expérience intéressante à ceux qui souhaitent travailler en informatique.

Python

Simple, rapide, donne de bonnes habitudes de programmation, léger (à tel point que repasser à un autre langage après semble vite pesant), ouvert, énormément de documentation sur internet. Il permet de vite se faire plaisir soit en faisant des scripts puissants (grâce notamment à la programmation objet). Il est également "nouvelle génération" et portable. Il fonctionne avec une machine virtuelle qui peut être empaquetée grâce à certaines classes disponibles sur le web (on en trouve de toutes sortes et à toutes fins...) dans un Exe pour distribuer l'application de manière simple et indépendante.

.Net
Alias le futur selon Microsoft, du bon et du mauvais dans cette plateforme, mais elle permet de programmer des applications graphiques de manière simple, ne tourne que sous Windows avec quelques tentatives adaptives sous Linux. Je conseillerai le C# qui est le langage "natif" de cette plateforme : sa syntaxe est proche du C et de java, pas de pointeurs, tout se passe à un niveau d'abstraction supérieur aux anciens langages Microsoft. C'est comparable à Java et à Python, sauf que ce n'est pas libre.

Depuis la version 2008 (Framework 3.0) Microsoft propose une version gratuite Visual Studio Express. Ces versions permettent de développer les mêmes applications que les versions payantes, en revanche elles ne sont pas dotées de nombreux outils pour simplifier le développement (seul ou en groupe), le déploiement ou encore le mutli-plateforme.

Ceux que l'on peut utiliser à la rigueur

C/C++

Ce sont de bons langages, mais avec une syntaxe compliquée qui ne facilite pas leur apprentissage. De plus l'utilisation des pointeurs rend leur utilisation, même par des professionnels, souvent ardue. L'avantage est que ces langages peuvent être utilisés sur plusieurs plateformes (après compilation) et que, une fois maîtrisés, ils font paraître les autres plus simples. Un avantage : elle confère de bonnes habitudes. Ce sont des langages encore très utilisés

dans le monde professionnel. A utiliser exclusivement dans le but de faire de l'informatique un métier.

PHP

Un langage initialement prévu pour le développement de pages web dynamiques. Sa syntaxe est proche du C++. PHP est assez "simple" à aborder et permet de communiquer assez facilement avec une base de données. Pour développer des pages web dynamiques, il requiert aussi une bonne connaissance du HTML et éventuellement du SQL si vous avez besoin d'une base de données.
Il permet notamment d'avoir rapidement une "visibilité" de son travail en le mettant en ligne, et ainsi d'avoir les commentaires/suggestions des internautes. On peut également développer des applications normales avec ce langage (hors web). Notons aussi que vous pouvez programmer de manière procédurale ou orientée objet, au choix, ça permet un bon tour d'horizon.
Ceci dit c'est un bon langage mais pas forcément le meilleur pour apprendre. Il est puissant et permissif si bien que vous risquez de prendre de mauvaises habitudes en apprenant avec. Notamment au niveau des pages web, il n'influence pas à programmer proprement de prime abord. Si vous chercher un langage facile à apprendre, qui permet d'apprendre l'orienté objet, et qui laisse derrière lui de bonnes habitudes, choisissez plutôt un langage comme python.

Visual Basic/Basic : D
Déconseillé formellement par tous les professionnels qui ont travaillé avec. Il donne de mauvaises habitudes de programmation (déclaration des variables non nécessaire, aucune notion de valeur/référence, tentative avortée de faire de l'objet,...), et de plus Microsoft (il s'agit d'un langage propriétaire) l'a laissé tomber au profit du Framework.Net.

PS : débuter en programmation, c'est aussi se trouver confronté à une épineuse question : mais pourquoi mon programme ne marche pas ? Les messages d'erreur fournis (ou non) par les environnement de compilation ou d'exécution (ou le comportement de l'ensemble) plongent parfois les débutants dans une perplexité sans fond qui peut devenir exaspérante. Une bonne méthode est de faire relire son code par quelqu'un d'expérimenté, et/ou d'expliquer à autrui votre problème. Parfois l'énoncé à haute voix de votre raisonnement vous mettra sur la bonne voie et fera de vous le vainqueur de votre chasse au fameux bug. Enfin, la capacité à (bien) programmer va (souvent) de pair avec celle de s'abstraire de ce monde matériel dans lequel nous sommes tous plongés. Bienvenue dans un monde où rien n'est réellement concret.

Si la programmation vous mord, vous en rêverez peut-être ... Ayez dans ces moment-là une feuille et un crayon à papier (l'arme absolue de l'informaticien) sur votre table de chevet, notez les solutions qui vous viennent à l'esprit et testez-les le matin venu, vous serez probablement surpris du conseil que la nuit vous a alors apporté.