

## **Chapitre III : Les Techniques de Spécification en Génie Logiciel.**

### **I. Introduction sur les techniques de spécification.**

De manière générale, tout produit logiciel doit être spécifié le plus clairement possible comme c'est le cas dans d'autres domaines de l'ingénierie (l'automobile, le bâtiment, la chimie...).

Dans le domaine informatique, le terme 'spécification' peut être utilisé à différents niveaux du développement du logiciel :

- spécification des exigences (Besoins),
- spécification d'implantation,
- spécification technique d'un module,
- spécification d'une propriété,
- spécification de tests,
- spécification de procédures de tests,
- spécification de données,
- spécification de traitements.

Ces spécifications peuvent être considérées comme un **contrat** entre un client (la collectivité qui veut réaliser le produit) et un producteur (l'entreprise de développement).

Il s'agit de ce fait d'une activité dont le but est de préciser (clairement) un aspect.

Dans ce cours, nous nous intéressons essentiellement à la Spécification des besoins (ou exigences).

### **II. Définitions**

- a) Artefact : terme qui désigne toute chose produite ou consommée par une étape du processus de développement. Exemple : un diagramme, du texte, du code, fichiers, Documents...etc.
- b) Besoins : conditions auxquelles un système et plus généralement un projet doivent satisfaire.
- c) Spécifications Fonctionnelles : C'est une description des fonctions que le logiciel doit réaliser. Elles ne décrivent que le comportement externe du logiciel.
- d) Cahier des charges : c'est un document qui vise à définir simplement les « spécifications de base » d'un produit ou d'un service à réaliser.

### **III. Analyse des exigences et spécification**

#### **A) Les Types d'exigences**

Selon le standard IEEE 830 (en 1998) on distingue les exigences (besoins) liées aux systèmes informatiques suivantes :

- **Exigences d'interfaces externes** : comment le logiciel interagit-il avec les individus, avec l'OS, avec d'autres logiciels, avec le hardware ?
- **Exigences de performance** : vitesse d'exécution, temps de réponse, disponibilité, temps de détection d'erreur, temps de recouvrement d'erreur...
- **Exigences de qualité** : portabilité, correction, maintenabilité, sécurité.

- *Exigences imposées à l'implantation* : langage, politique de gestion de données, de gestion de ressources, l'OS, consommation d'énergie, encombrement mémoire, poids.

## **B) Les Processus d'analyse des exigences/besoins**

Les activités d'analyse des exigences sont souvent regroupées en quatre familles d'activités : étude de la faisabilité du système, identification et analyse des exigences, *spécification* des exigences et, *validation* de ces exigences.

### **1. Etude de la faisabilité**

L'objectif de cette phase est de répondre aux questions suivantes :

- ✓ Est-ce que le système va contribuer au développement de l'organisation ? Apporte-t-il un plus ?
- ✓ En quoi va-t-il aider à résoudre des problèmes s'il y en a ?
- ✓ Est-ce que le système peut être réalisé avec la technologie actuelle, dans les délais requis et avec les coûts indiqués ?
- ✓ Est-ce que le système peut être intégré aux autres systèmes déjà existants ?

### **2. Identification des exigences et leur analyse**

L'identification des exigences est liée à chaque domaine de systèmes en fonction des secteurs d'activités. Il s'agit ici de répertorier les exigences, de les classer par catégories, de leur donner des priorités, de déterminer les liens et les conflits entre les exigences.

### **3. La Spécification des exigences**

Après les phases d'identification et classification des exigences, on passe à la spécification (description) ces exigences en vue d'une utilisation dans les phases suivante du cycle de développement en utilisant des techniques formelles.

### **4. Validation des exigences**

Il est toujours conseillé de valider les exigences avec le client (l'utilisateur) avant de passer aux phases suivantes. Pendant l'étape de validation, plusieurs contrôles doivent être effectués :

- *Contrôle de validité* : est-ce les fonctions prévues sont prévues pour être utilisées par les bons acteurs, aux bons endroits, dans les bonnes conditions ?
- *Contrôle de cohérence* : est-ce que certaines exigences ne sont pas en conflit ou contradictoires avec d'autres ?
- *Contrôle de complétude* : est-ce tous ce les besoins ont été pris en compte pour concevoir et réaliser le système ?
- *Contrôle de réalisme* : est-ce les objectifs fixés peuvent être atteints avec les technologies ciblées ?
- *Vérifiabilité* : pour éviter les malentendus avec le client, contrôler si tout a été écrit de manière à pouvoir vérifier assertions (affirmations, dires, discours...) des uns et des autres.

**Cette vérifiabilité est facilitée par l'utilisation de méthodes de spécification formelles.**

## **C) La Classification des exigences**

Une catégorisation des exigences existe pour faciliter l'identification des exigences :

- **exigences utilisateur** : indiquent ce que le logiciel doit faire et sous quelles conditions de fonctionnement ;
- **exigences système** : indiquent des éléments sur le fonctionnement du logiciel (ergonomie des interfaces, aspects graphiques...) ;
- **exigences logiciel** : apportent des détails sur des contraintes de conception et implantation.

Un autre critère est celui qui consiste à séparer les exigences en deux :

- **Exigences fonctionnelles** : elles indiquent ce que le système doit faire et comment il doit réagir aux différentes situations (événements) qui peuvent se présenter. Elles indiquent aussi les liens entre les entrées et les sorties du système.
- **Exigences non fonctionnelles** : elles indiquent des contraintes sur la manière dont le service du système est rendu. Ces besoins incluent notamment : les contraintes de temps, l'espace mémoire, le débit, la fiabilité, l'ergonomie, l'utilisation de standards et de règles de développement, de sécurité, de lisibilité, de maintenabilité, de coûts, d'éthique.

*Certains parlent de besoins techniques au lieu de non-fonctionnels.*

## IV. Les Techniques de Spécification des exigences

Deux critères *perpendiculaires* sont généralement utilisés pour classer les techniques de spécification avec des méthodes formelles: (Voir Schéma en fin du Cours, Page 9)

- ✓ **Le caractère formel** (utilisation de formalisme). On distingue :
  - des **spécifications informelles** (en langue naturelle),
  - des **Spécifications semi formelles** (souvent graphiques, dont la sémantique est plus ou moins précise),
  - Les **Spécifications formelles** (quand la syntaxe et la sémantique sont définies formellement par des outils mathématiques).
- ✓ **Le caractère opérationnel ou déclaratif** : les **spécifications opérationnelles** décrivent le *comportement désiré*; par opposition, les **spécifications déclaratives** décrivent seulement les *propriétés désirées*.

### A) Les spécifications en langue naturelle

Leurs caractéristiques sont les suivantes :

- ✓ très souples,
- ✓ conviennent pour tous les aspects,
- ✓ très facilement communicables à des non-spécialistes.
- ✓ manquent de structuration, de précision et sont difficiles à analyser.

### B) Les spécifications dans des langages spécialisés ou des langages informatiques de haut niveau

Ce sont des Spécifications qui sont mises en œuvre à l'aide des langages semi-formels qui comportent des sections et champs prédéfinis, ce qui force à une certaine structuration.

Des langages de haut niveau sont aussi utilisés comme des pseudo-codes pour décrire au moins les fonctionnalités attendues.

### **C) La Spécification avec des diagrammes de flots de données**


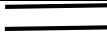


Il s'agit d'une technique semi-formelle et opérationnelle. Les DFD (*Design Flow Datagrams*) décrivent des collections de données manipulées par des fonctions. Les données peuvent être persistantes (dans des stockages) ou circulantes (flots de données).

Un DFD Indique la circulation des données à travers un ensemble de composants qui peuvent être des tâches, des composants logiciels.

Généralement, des conventions graphiques sont utilisées pour représenter :

- ✓ les fonctions (ou processus) du système ;
- ✓ les flots de données (le plus souvent la direction d'un flux de données est matérialisée par une flèche) ;
- ✓ les entités externes qui interagissent avec le système mais qui ne sont pas spécifiées dans leur détail ;
- ✓ les points de stockage d'informations.

#### **1. Les Outils de Représentation Graphique d'un DFD**

Outils Représentés	Symboles de représentation
les fonctions par des cercles	
les stockages par des boîtes ouvertes	
les flots par des flèches	
les entités à l'interface par des rectangles	

Les DFD peuvent apparaître à différents niveaux d'abstraction du système (niveau le plus élevé, niveau des interfaces, niveau d'implantation réelle), selon les besoins. On aussi de 'Diagramme de Contexte'.

### **Exemples de représentation de DFD**

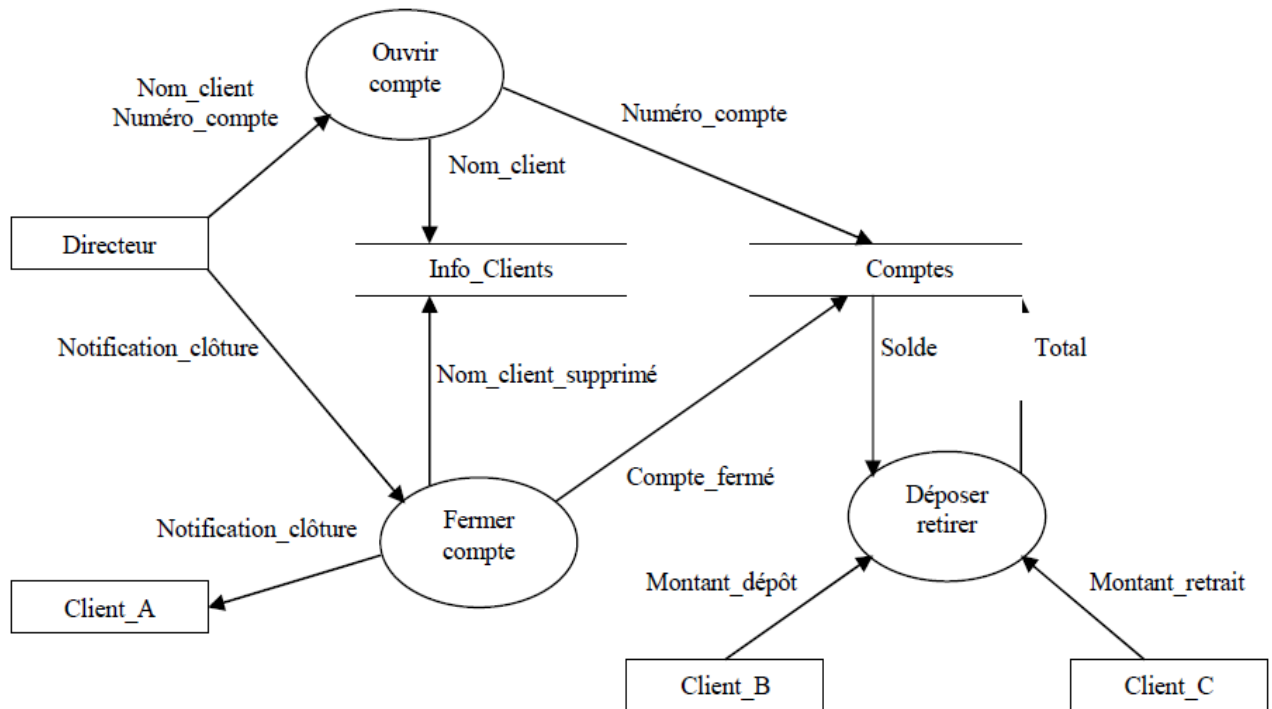
#### **Exemple 1**

La figure ci-dessous montre un DFD qui illustre partiellement le fonctionnement d'une banque. On distingue :

- ✓ **Quatre entités externes** (Directeur, Client\_A, Client\_B et Client\_C),
- ✓ **Trois processus** (Ouvrir\_Compte, Fermer\_Compte, Deposer\_Retirer),
- ✓ **Deux stockages** (Info\_Clients et Comptes),
- ✓ **8 flux** (Nom\_client, Numéro\_compte, Notification\_clôture, Montant\_Retrait, Montant\_Dépôt, Solde, Total, Nom\_client\_supprimé, Compte\_fermé).

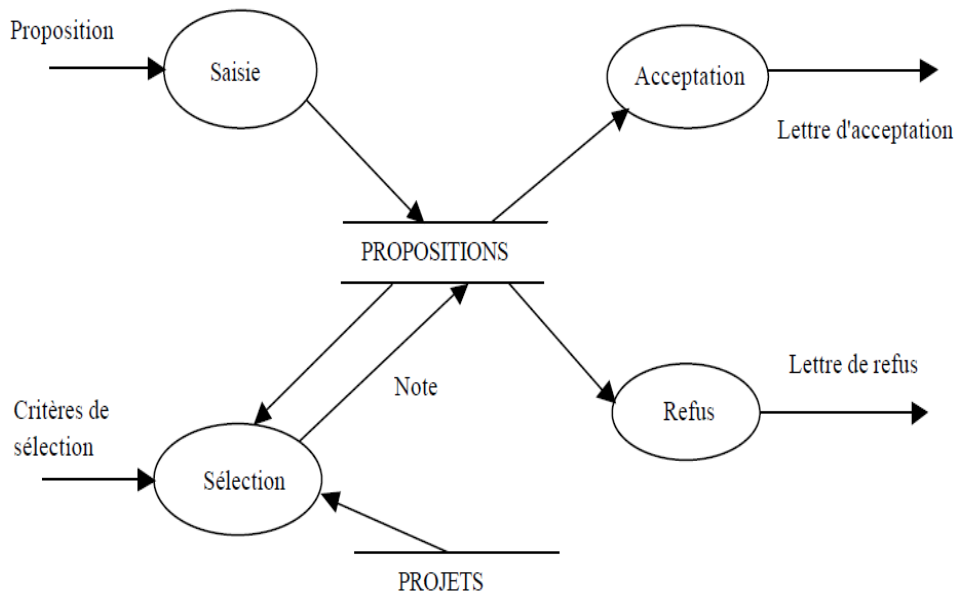
Les DFD sont simples et faciles à comprendre par des non informaticiens qui participent dans des projets de logiciel. Les DFD ont été intégrés à de nombreuses méthodes de conception comme **SA-RT, DE MARCO**.

**N.B.** Les DFD sont généralement insuffisants à eux seuls et doivent être complétés par d'autres moyens de spécification.

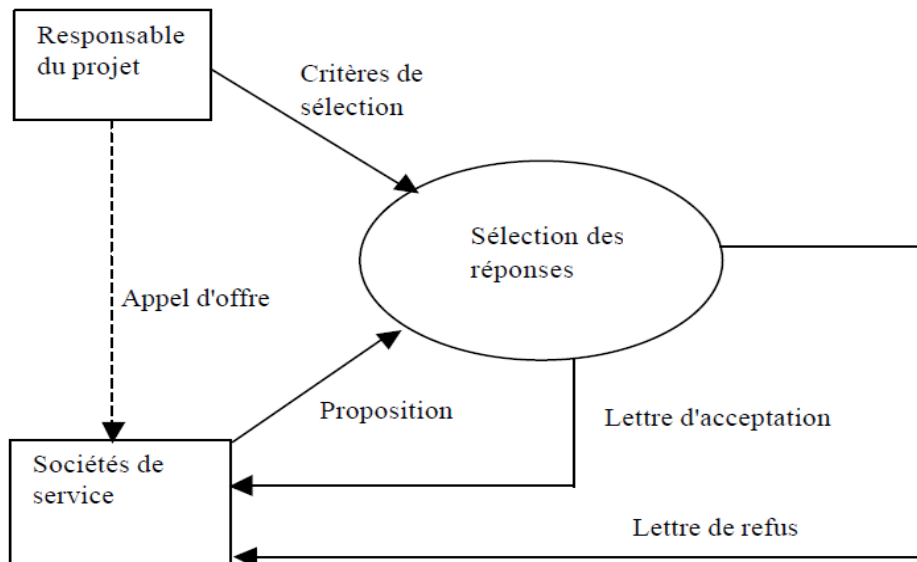


### **Exemple 2**

Concernant la « *sélection des réponses à un appel d'offres lancé dans une Entreprise* », le Diagramme de Contexte et Le Diagramme de Flots de Données raffiné sont représentés comme ci-dessous.



**Diagramme de contexte**



Raffinement du DFD précédent

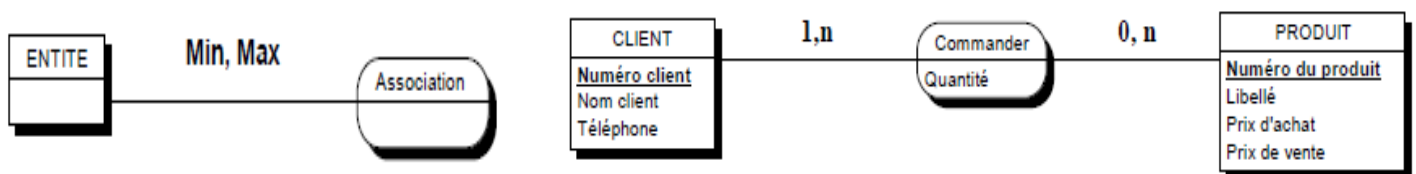
## D) La Spécification avec les machines à états finis

Ce sont des techniques les plus utilisées, essentiellement pour la spécification des comportements du fait qu'elles fondées sur la notion d'états-transitions : *automates à états finis*, *réseaux de Petri*, *Grafcet*...

## E) Spécification avec le modèle entités-relations selon la Méthode MERISE (Méthode de Réalisations Informatiques pour les Systèmes d'Entreprise)

Il s'agit d'une technique **semi formelle et déclarative**. Ce modèle permet de spécifier la *structure des données* et de *leurs relations*. Les concepts du modèle de base sont :

- a) les **entités**, qui sont des collections d'items partageant des propriétés communes (occurrences d'entités),
- b) les **associations** (ou *relations*), qui traduisent l'existence de liens entre entités (occurrences d'associations entre occurrences d'entités),
- c) les **attributs** (ou *propriétés*), attachés aux entités et aux associations et qui les caractérisent.
- d) **Les cardinalités** qui déterminent les nombre de fois qu'une occurrence d'une Entité » participe à une Association (Au minimum et au Maximum).



**RG** : Un client commande au moins 1 produit (sous-entendu ou plusieurs) et un produit peut ne pas encore avoir été commandé, comme il peut l'avoir été plusieurs fois.

## F) Style Particulier de spécification : Langage Z

De manière pratique, la notation Z n'est pas un langage de Programmation puisqu'il n'existe pas de compilateur permettant d'exécuter le code. Il s'agit uniquement d'un modèle simple, permettant à toutes les personnes travaillant sur un projet (les programmeurs, les testeurs, les personnes devant rédiger la documentation, et le client) de s'accorder sur le fonctionnement du programme sans entrer dans les détails de l'implémentation.

Pour faire abstraction de l'implémentation, la **notation Z** opère sur des objets mathématiques et les effets des fonctions du programme sont décrits par des formules du calcul des prédicats, et elle décompose le programme en **petits modules appelés schémas** (

### 1. Les notations mathématiques

#### a) Les Ensembles

La notation Z fournit comme ensembles de départ celui des entiers naturels **N** et de ses dérivés **Z** et **Q**.

L'ensemble vide est noté par  $\emptyset$ .

#### b) Les Opérations

Les opérations sur les ensembles sont notées de manières usuelles :

- l'inclusion :  $\subseteq$
- l'inclusion stricte :  $\subset$
- l'appartenance :  $\in$
- la non-appartenance :  $\notin$
- le produit cartésien :  $\times$
- l'union :  $\cup$
- l'intersection :  $\cap$
- la différence :  $\setminus$

#### c) Les Prédicats

Les principaux connecteurs sont les suivants :  $\neg$  (non),  $\wedge$  (et),  $\vee$  (ou),  $\Rightarrow$  (implique) et  $\Leftrightarrow$  (bi-implique ou équivalent).

#### d) Les Relations

Les relations sont une notion très importante en **Z**. Il s'agit d'ensembles de paires ordonnées. On utilise le symbole  $\leftrightarrow$  pour parler de relation.

### 2. Le découpage des programmes

Le principe de la spécification en **Z** est de découper le programme en sous-tâches. Ces sous-tâches sont ensuite spécifiées grâce à des schémas. C'est l'ensemble de ces schémas qui constitue le programme.

Il en existe de trois types :

a) Pour spécifier une sous-tâche :

titre[param]
déclarations
prédicats

b) Pour créer des objets de portée globale, c'est à dire valable dans tous les autres schémas de la modélisation :

déclarations
prédicats

c) Pour établir des définitions génériques, c'est à dire à instancier dans un (ou plusieurs) autre(s) schéma(s) :

[params]
déclarations
prédicats

**Exemple :** Enregistrement des passagers à bord d'un avion

1. **Déclaration des variables**

- [PERSONNE] : Ensemble des personnes identifiées de manière unique
- Capacité : N Capacité de l'avion

2. **Définition de l'état du système**

Avion
àBord : $\mathbb{P}$ PERSONNE
#àBord $\leq$ capacité

3. **Description de l'état initial**

Init
Avion
àBord = $\emptyset$

4. **Description de l'évolution du système**

$\Delta$ Avion
àBord : $\mathbb{P}$ PERSONNE
àBord' : $\mathbb{P}$ PERSONNE
#àBord $\leq$ capacité
#àBord' $\leq$ capacité

5. **Description des opérations**



Embarquer
$\Delta \text{Avion}$
$p? : \text{PERSONNE}$
$p? \notin \text{àBord}$
$\# \text{àBord} < \text{capacité}$
$\text{àBord}' = \text{àBord} \cup \{p?\}$

Débarquer
$\Delta \text{Avion}$
$p? : \text{PERSONNE}$
$p? \in \text{àBord}$
$\text{àBord}' = \text{àBord} \setminus \{p?\}$

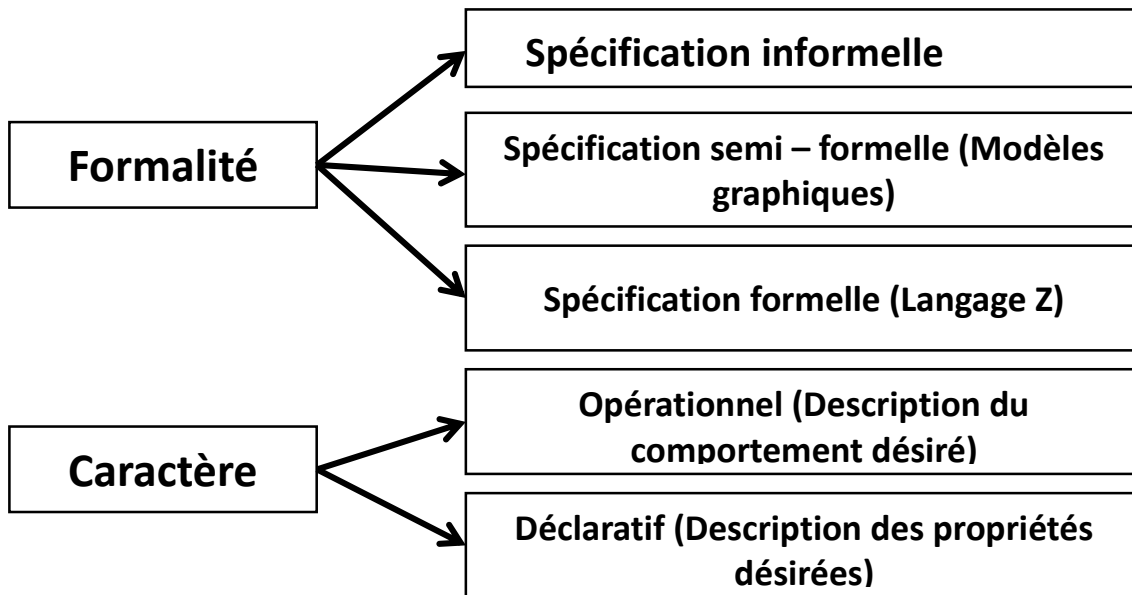


Schéma des différents Styles de Spécification