ANALYSE ET SPECIFICATION DES BESOINS
1ère Etape du cycle de vie

1. DEFINITION

C'est le processus visant à établir quelles fonctionnalités le système doit fournir et les contraintes auxquelles il sera soumis. Répondre au **quoi et non au comment ?**

2. OBJECTIFS

- Comprendre la nature exacte du problème. **Par l'analyste et par le client**
- Fournir un document compréhensible par le client et les concepteurs du système : **Le fondement du contrat**
- Définir la base pour la validation des étapes ultérieures. La validation utilise la spécification comme base pour répondre à la question principale suivante : **A t'on conçu ce qui a été demandé ?**
- Préciser les contraintes de réalisation : les outils et environnement de développement à utiliser,
- Prendre des décisions : après une première analyse, on peut prendre la décision finale de développer le système, ou d'acheter un système existant et l'adapter, etc
- Réduire les coûts de développement du système.
  Une spécification rigoureuse des besoins permet d'éviter les oublis pouvant apparaître tardivement dans le cycle de vie. En effet de tels oublis nécessitent des retours en arrière, ce qui augmente les coûts de développement : **Eviter l'oubli, les allers retours dans le cycle de vie**

3. LES ETAPES

-**Analyse des besoins** (Requirement Analysis) : Rassembler toutes les informations nécessaires à la compréhension du problème

-**Spécification des besoins** (Requirement Specification) : Spécifier de manière claire et précise le résultat de l'analyse en utilisant des méthodes de modélisation (voire étape de spécification).

Nous allons d'abord préciser ce que nous qualifions de besoins en les classons en deux catégories. Ensuite, nous donnons quelques éléments permettant d'aborder l'analyse puis la spécification.

Les éléments clés de l'analyse est de la spécification sont donc les besoins que l'on peut classer en deux catégories :

- **Besoins fonctionnels (exigences fonctionnelles)**: les services que l'utilisateur attend du système. Il s'agit des activités que le système doit exécuter. Dans premier temps, ces besoins peuvent être exprimés en langage naturel accompagné éventuellement de diagrammes graphiques.
  Exemple : le système de distribution automatique de billets de banque (DAB).
  - Le système doit vérifier la validité de la carte.
  - Le système doit mettre à jour le compte du client après un retrait ou un dépôt.
- **Besoins non fonctionnels (spécifications techniques)**
  Les contraintes sous lesquelles le système doit rester opérationnel et les standards auxquels il doit se conformer. Par exemple :
  - Les besoins en performance
    Exemple : le système doit répondre à un client au bout de 2 secondes après l'insertion de la carte.
  - Les contraintes de développement (processus de développement)

Exemple : le système doit être implanté sur une machine supportant le système d'exploitation UNIX.

4. ANALYSE DES BESOINS

Acteurs : regroupent tous les intervenants qui constituent la source d'information pour l'élaboration des besoins. Un intervenant peut appartenir à l'une des 4 catégories suivantes : 1) les utilisateurs qui vont exploiter le système et utiliser ses fonctionnalités 2) les clients qui investissent dans le développement du système et qui en sont propriétaires 3) le personnel technique qui est responsable du bon fonctionnement du système dans l'environnement de l'organisation 4) éventuellement les entités extérieures, tels les clients de l'organisation.
La première tâche de l'analyste est d'identifier chacun des types d'intervenants concernés par le système à construire.

Procédé :

-Séries de questions posées aux intervenants (clients, utilisateurs, ….)
-Éventuellement, les documents fournis par le client
-Analyse de la tâche, généralement quand il s'agit d'automatiser une activité manuelle.
-Création de modèles pour représenter et communiquer les besoins. Par exemple, la modélisation des besoins fonctionnels par des cas d'utilisation (UML) et des données par une première ébauche d'un diagramme de classes.

Problèmes rencontrés : divers et en particulier :

-Comment organiser et structurer les informations obtenues ?
-Comment résoudre les contradictions pouvant exister entre les différents interlocuteurs ?

Eléments de solution :

-Utiliser les méthodes d'analyse : SA, SADT, Merise, OOA (UML)...
-Qualités de l'analyste en matière de communication
-Appliquer les trois principes suivants :

1-**Décomposition** : Appréhender le problème en le décomposant en plusieurs parties.
Cas d'un système d'exploitation : difficulté d'appréhender tout le système dans sa globalité.
==> Le décrire à travers ces composants :

- système de gestion de fichiers
- système de gestion de la mémoire
- système de gestion de processus.

Après la décomposition, on peut étudier chaque partie séparément.

2-**Abstraction** : Appréhender le problème à un bon niveau d'abstraction. Eviter les détails.
Par exemple, pour le système de gestion des fichiers d'un système d'exploitation, décrire les fonctions de haut niveau sans se soucier dans un premier temps de l'organisation des fichiers en mémoire secondaire :

- création et destruction de fichier
- ouverture et fermeture de fichier
- gestion des répertoires
- interface utilisateur

3-**Projection :** Etudier le système à partir de plusieurs points de vue.
Cas d'un système d'exploitation. Analyser selon :

- le point de vue de l'utilisateur
- le point de vue du programmeur
- le point de vue de l'administrateur

Pour chaque point de vue, identifier les besoins qui lui sont associés.

5- Spécification des besoins

Il s'agit de décrire de manière précise et à travers des modèles les besoins du système : rendre plus formalisés les résultats de l'étape d'analyse. Le document qui en résulte est le document de spécification.

Caractéristiques d'un bon document de spécification.

- Doit spécifier uniquement le comportement externe du système et les réponses aux événements non désirables.
- Doit spécifier les contraintes de réalisation.
- Doit être facile à mettre à jour.
- Doit contenir des indications concernant les étapes ultérieures.
- Doit servir d'outil de référence pour la validation et la maintenance.
- Doit être complet et cohérent.

Les principaux composants d'une spécification

Peuvent être classés en 3 catégories : les besoins fonctionnels, les besoins non fonctionnels et les interfaces externes. Nous allons les décrire séparément.

5.1 Les besoins fonctionnels

Définition: Les services ou les fonctionnalités que l'utilisateur attend du système. Pour chaque service :

- les données nécessaires à l'exécution du service
- les résultats qu'il produit
- une description générale de son rôle
- le traitement des erreurs

L'expression des besoins fonctionnels peut prendre plusieurs formes selon la méthode et le langage de spécification choisis et le niveau de détail à atteindre. Nous allons explorer les différentes formes d'expression possibles tout en soulignant leurs avantages et leurs inconvénients. Il s'agit ici d'une présentation synthétique : nous consacrerons une bonne partie des cours suivants à l'étude détaillée des deux dernières méthodes et langages de spécification (méthodes semi-formels très utilisées : UML, SA, SADT et méthodes formelles : spécifications algébriques, langage Z et B). Les deux premiers langages que nous présentons ci-dessous (langage naturel, langage structuré) sont simples à comprendre et constituent deux moyens d'expressions pouvant être utilisés pour documenter les spécifications construite à l'aide de méthodes formelle ou semi-formelles.

Méthodes et Langages d'expression des besoins fonctionnels

5.1.1. Spécification en Langage naturel

Dans ce cas, les fonctions (ou services) du système sont définies en utilisant le langage naturel éventuellement en introduisant des schémas et des tableaux.

Exemple de quelques fonctions du système TPE (développement d'un terminal de paiement électronique)

Fonction : <<Fournir un ticket>>

- Fournir une preuve d'achat au client
- Fournir une preuve de la transaction (cas paiement accepté)

Fonction : <<Communiquer avec le système de paiement>>

- permettre au TPE de dialoguer avec le système de paiement par carte.

Avantages de ce mode d'expression :

- plus expressif.
- compréhensible à la fois par les clients et les développeurs.

Inconvénients :
- besoins fonctionnels imprécis à cause de l'ambiguité inhérente au langage naturel.
- mélange des besoins fonctionnels et non fonctionnels.
- la spécification est beaucoup trop flexible : le risque d'exprimer des besoins voisins dans des termes totalement différents.
- les besoins ne sont pas cloisonnés : les conséquences d'un changement ne peuvent être évaluées qu'après un examen de chaque besoin, et non d'un groupe de besoins voisins.

5.1.2. Spécification en Langage structuré

Utilisation contrôlée du langage naturel et de formulaires standards de spécification. Les formulaires peuvent être structurés autour :
- des objets manipulés par le système.
- des fonctions qu'il remplit (ce mode de structuration est le plus utilisé lorsqu'on se base sur un langage structuré).
- des évènements qu'il doit traiter.

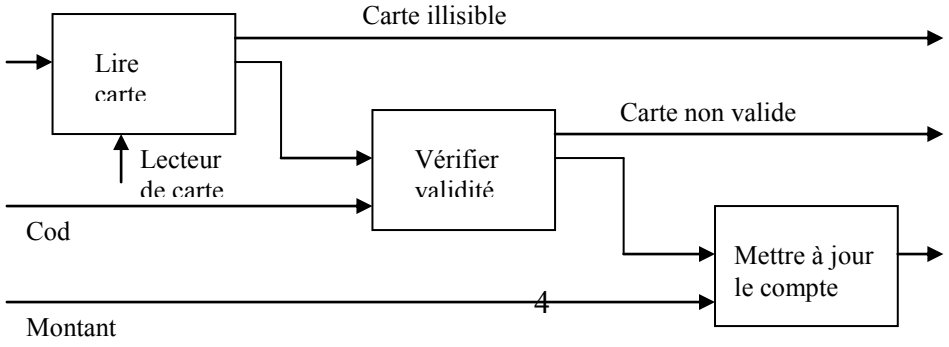
Exemple :
Fonction : Vérifier_validité_carte
Description : Cette fonction vérifie que la carte introduite par un utilisateur provient d'une banque reconnue, qu'elle est à jour, et qu'elle contient des informations appropriées ainsi que des détails sur les dates et les montants des précédents retraits.
Entrées : identification de la banque, numéro de compte, date d'expiration, date et nature de la dernière transaction.
Sorties : Etat de la carte = (OK, invalide)
Besoins : Liste des banques, format du compte, date du jour
Pré-condition : carte introduite et bande magnétique lue
Post-condition : Identification de la banque dans la liste des banques **et** Numéro de compte au bon format **et** date d'expiration >= date d'aujourd'hui **et** Etat de la carte = OK
ou (si l'un de ces tests est faux) Etat de la carte = invalide

Cette spécification utilise les pré et post conditions pour décrire le rôle de la fonction. Les objets utilisés (Liste des banques,....) doivent être définis dans un dictionnaire de données. Notons que cette spécification n'introduit pas d'information sur le séquencement dans la définition des pré et post conditions.

- Avantages de ce mode d'expression :
  - évite un certain nombre de problèmes de structurations.
  - peut être supporté par des outils.
- Inconvénients : demeure le problème de l'ambiguïté dû à l'utilisation en partie du langage naturel.

5.1.3. Spécifications semi-formelles

Généralement associées à des méthodologies (SA, SADT, UML,…), ces spécifications sont exprimées dans des langages graphiques. Elles sont très couramment utilisées et bénéficient d'environnements logiciels d'aide à la spécification. Nous consacrerons une bonne partie du cours à ces méthodes. A titre d'exemple, voici une spécification exprimée en SADT (Structured Analysis and Design Techniques).



SADT est une méthode qui analyse un systèmes en mettant l'accent sur les fonctions qu'il doit réaliser. Chaque fonction peut être décomposée en plusieurs autres fonctions (décomposition fonctionnelle d'un système). Un rectangle (appelé actigramme) représente une fonction du système. Les flèches entrantes (respectivement sortantes) représentent les données (respectivement les résultats). La flèche en dessous de la fonctionnalité « Lire carte » précise le support physique nécessaire à l'exécution de cette fonction.

5.1.4 . Spécifications formelles

se sont des spécifications exprimées dans des langages possédant une sémantique formelle permettant de raisonner sur la spécification pour faire des preuves de propriétés. Proche des mathématiques, ces langages sont généralement fondés sur la logique du premier ordre, la théorie des ensembles ou les systèmes formels. Nous donnons ci-dessous un exemple de tels langages : la spécification algébrique des types abstraits de données.

Exemple: Spécification algébrique du type Pile

type Pile (ELT)
opérations : décrit l'interface du type en donnant ses opérations et leurs interfaces
creer : ---> Pile , chaque opération est définie par son domaine de départ et son domaine d'arrivée
empiler : Pile * ELT ---> Pile
depiler : Pile ---> Pile
vide : Pile ---> Booléen
sommet : Pile ---> ELT
Axiomes : décrivent les propriétés des opérations ou services proposés.
depiler (créer()) = créer()
depiler (empiler (p, e)) = P
vide (créer) = vrai
vide (empiler (p , e)) = faux
sommet (empiler (p , e)) = e
sommet (créer()) = indéfini
fin type

Avantages :
- spécification précise et sans ambiguïté
- possibilité de faire des preuves de complétude et de consistance
- peut être traitée de manière automatique grâce à des outils
- possibilité d'avoir des spécifications exécutables

Inconvénients :
- nécessite un grand effort de formation
- non compréhensible généralement par les clients

Conclusion sur les méthodes de spécification :

Nous venons d'examiner les différentes classes de méthodes et d'expression des spécifications. Mais alors laquelle choisir ?

Il n'y a pas de réponse unique et valable pour tous les systèmes. Disons que dans la pratique et pour de nombreux systèmes, on utilise les méthodes semi-formelles que l'on documente à l'aide du langage naturel et du langage structuré (vous ferez l'expérience à travers le projet que vous allez réaliser). Bien qu'elles soient très peu utilisées, les méthodes formelles ont montré dans l'industrie leur efficacité dans la spécification de certains systèmes critiques (une panne peut provoquer une catastrophe). Grâce

à la possibilité de prouver formellement les propriétés d'un système avant sa mise en œuvre, ces méthodes constituent un pas vers la certification formelle de logiciels.

5.2 Les besoins non fonctionnels

Ils constituent les buts opérationnels liés à l'environnement au matériel et aux logiciels de l'organisation. Ils décrivent aussi les restrictions ou les contraintes qui pèsent sur le système et en particulier sur ses services. On peut les subdiviser en plusieurs parties que nous présentons ci-dessous.

Besoins en performances

- Temps de réponse exigé (à une requête par exemple)
- Taille des informations manipulées et espace mémoire nécessaire
- Restrictions sur la représentation des données
- Nombre moyen de transactions par seconde
- Nombre de terminaux connectés au système

Voici un tableau de métrique qui résume et donne d'autres éléments liés aux performances.

Propriété	Métrique
Vitesse	Nombre de transactions par seconde Temps de réponse à l'utilisateur/à un événement Temps de rafraîchissement de l'écran
Taille	K octets, nombre de puces de RAM
Facilité d'utilisation	Durée de formation, nombre d'écran d'aides
Fiabilité	Temps moyen entre deux pannes, probabilité de non disponibilité, Taux d'apparition des pannes
Robustesse	Temps de redémarrage après une panne, Pourcentage d'événements causant des pannes, Probabilité de corruption de données sur une faute
Portabilité	Pourcentage d'instructions dépendant de la cible

Les contraintes de conception

expriment les contraintes sur le processus de développement

- utilisation de méthodes standards
- langage de programmation
- système d'exploitation
- matériels
- structure de la documentation (format de rapports)

Exemple de contrainte de conception :
"Le processus de développement du système et les documents devront être conformes à la norme DEF-STAN OO-65"

5.3 Les interfaces externes

expriment les relations entre le système et :

- Les utilisateurs du système : les messages d'erreurs (clarté, longueur), la description des formes et éditions sur papier et écrans
- Le matériel : lorsque le système à développer interagit avec un système matériel, il faut décrire l'interface entre les deux
- Le logiciel : liaison avec une base de données, des bibliothèques mathématiques, graphiques, les procédures d'échange de messages.

6. Structure d'un document de spécification

Cette structure est issue de la norme IEEE, que j'ai adaptée et commentée.

Sommaire: plan du document

1 - Introduction

Buts et destinataires du document

- Définir en quelques lignes s'il s'agit de développer ou d'améliorer un système
- La manière dont le système s'insère dans les objectifs commerciaux et stratégiques de l'organisme commanditaire.

Définitions - Abréviations

- Les termes techniques utilisés
- Les abréviations

Présentation générale du document (comment il est organisé)

2 - Description générale

Environnement ou contexte du système

Décrit les relations du système avec d'autres systèmes et les principales interfaces avec ceux-ci.

- Donner un diagramme, faisant apparaître le système et les acteurs avec lesquels, il interagit. Le système est vu comme un grand cas d'utilisation.
- Commenter en langage naturel, les acteurs et leurs interaction avec le système.

Caractéristiques des utilisateurs

Identifie les différents types d'utilisateur du système en précisant leurs caractéristiques.

- Expérience et connaissances dans l'utilisation des systèmes informatiques.
- Utilisateurs réguliers ou occasionnels

Les contraintes principales de développement

Précisent les procédures, les normes et les standards à utiliser dans le développement.

Hypothèses de travail

Décrivent tous les facteurs susceptibles de remettre en cause tout ou une partie de la réalisation des spécifications ainsi que d'éventuelles solutions de repli.

3 - Besoins fonctionnels

-Décrire les cas d'utilisation du système, en les regroupant par acteur, ou par grande fonctionnalité.
-Commenter les cas d'utilisation en utilisant des formulaires tels que:

Nom du cas d'utilisation

-Description (Rôle, présentation générale)
-les acteurs concernés
-Les entrées et leurs provenances
-Le traitement
-Les sorties (messages d'erreurs éventuellement)

4- Spécification des structures de données

Donner un modèle décrivant les entités du système et leurs relations. Pour chaque entité ; préciser :

- Nom de l'entité
- But et essence
- Eventuellement les relations avec les autres entités, leurs cardinalités

5- Spécifications des interfaces externes

Interface matériel/logiciel

- Configuration minimale et optimale sur laquelle le système peut s'éxécuter
- Les caractéristiques des interfaces entre le système et les dispositifs particuliers :
- Capteurs, actionneurs, périphériques (lecteur de chèques, de code à barres, ...)

- Protocole d'échange (réseau local, ...)
- Le type de liaison (série, parallèle, ...)

Interface logiciel/logiciel

Pour chaque logiciel utilisé explicitement par exemple système de gestion de bases de données, bibliothèque de fonctions mathématiques, services du système d'exploitation, il faut préciser: le nom, son numéro de version, sa provenance, le but de son utilisation, et définir l'interface, éventuellement en renvoyant à une annexe ou un autre document.

Interface Homme/logiciel

- Spécification des formes des éditions sur papiers et sur écrans
- Spécification des menus, des messages d'erreur
- Définition d'un manuel d'utilisateur (préliminaire)

6- Les besoins en performance (Voir cours précédent)

- Nombre maximum de terminaux
- Nombre maximum de transactions simultanées
- Nombre de fichiers et leurs tailles
- Temps de réponse souhaité
- Les contraintes liées à l'environnement (temps réel)

7 - Les contraintes de développement (Voir cours précédent)

- Fiabilité et tolérance aux fautes
- Le comportement du système dans des situations anormales (les exceptions critiques)
- Sécurité
  - . Restriction sur l'utilisation de certaines commandes
  - . Contrôle des accès aux données
  - . Utilisation de mot de passe

-Utilisation de standards en ce qui concerne les méthodes, outils et langages de développement.

8 - Références

Décrit :

- les références bibliographiques
- les sources d'obtention des documents

Il faut donner les références de tous les documents utilisés dans l'élaboration de la spécification.

9 - Index

- Index alphabétique
- Index des fonctions

10 - Annexes

- Description rapide des méthodes utilisées
- Description rapide des systèmes , outils externes au système et qui sont utilisés dans les paragraphes précédents.

7. Bibliographie

-I. Sommerville. “Software Engineering”, 3rd edn. Wokingham : Addison-Wesley, 1989

-J. Satzinger, R.B. Jackson, S. Burd. “Analyse et conception des systèmes d’information”, Editions Rynold Goulet, 2002.

-I. Jacobson, G. Booch, J. Rumbaugh.“The Unified Software Developement Process”, Addison-Wesley, 1999.

-A. Koukam. “Software Engineering”, Course “Introduction to Software Engineering”, University of , University of Shanghai, China 2006.