Cours – CDSI – Ingénierie Logiciel

# Définition des besoins

Définition d’un terme informatique : Standard Glossary Software

Definition d’un besoin : A condition or capability needed by a user to solve a problem or achieve an objective. Une condition ou une capacité nécessaire par un utilisateur pour résoudre un problème ou

3 types de besoin :

* Au niveau utilisateur : compréhension par tout le monde
* Au niveau système. : pour les développeurs du projet
* Spécifications Logiciel : pour

Sujet : Gestion du bar 2017

* Suivit d’une forme standard
* Éviter d’imposer le choix aux personnes qui sont chargés de les faire.

Besoin de l’utilisateur :

* Assez global

Besoin Système :

* Plus précis dans le besoin.
* Pas comment le faire mais ce que l’on souhaite avoir.

Propriété du besoin :

* Vérification unitaire
* Complet
* Consistent
* Non Combinés
* Traçable
* Courent (encore un sens à un moment donnée)
* Ambiguë
* Vérifiable

Pour qui fait-on les besoins ?

* Pour toute personne ou organisation qui est concernée par le logiciel et qui en a intérêt
* Types de partie prenantes :
  + Utilisateurs finaux
  + Gestionnaire du système informatique
  + Propriétaire du système
  + D’autres personnes

Besoins fonctionnels et non-fonctionnels ?

* Besoin fonctionnel
  + Services que rend le logiciel
  + L’ensemble de services que le logiciel doit rendre, comment réagir à une action générale et dans les cas particuliers
  + Ce que ne doit pas le faire.
* Besoin Non-fonctionnel
  + Condition dans lesquels ils vont le rendre
  + Appareil hébergeant le logiciel
  + Système d’exploitation
  + Le langage de programmation
* Besoin de Domaine
  + Environnement dans lequel le logiciel doit évoluer

Soucis et imprécision

* Les développeurs peuvent répondre aux besoins d’une manière différentes que celle voulue.

Le besoin doit être complet et consistent.

* L’utilisabilité est compliquée
* Efficacité : C’est l’utilisation des ressources disponibles.
* Sureté de fonctionnement : Qu’est-ce que l’on tolère en panne ?
* Sécurité : L’authentification, les autorisations, l’intégrité des données, stabilité des données
* Organisationnelle : Environnement d’exécution du système
* Opérationnelle : Qui va se charger de la mise en œuvre ?
* Développement : Utilisation d’une base de données particulière, langage particulier ?
* Éthique : Le plus économe possible par exemple.

On s’axera plus sur certains points que l’on jugera plus important.

Certains besoins non fonctionnels peuvent entrainer un besoin fonctionnel.

Le système doit être stable quand après la formation l’utilisateur n’a plus besoin d’annuler souvent une action.

Mesure :

* Vitesse
* Taille :
* Facilité d’usage
  + Nombre d’écran d’aide
* Sureté de fonctionnement
  + Temps moyen entre les pannes
  + Probabilité indisponibilité
  + Taux échec
  + Indisponibilité de l’application
* Résistance d’un logiciel
  + Temps de redémarrage après une panne
  + Pourcentage d’évènement causant une panne
  + Taux de données corrompue

Besoin non fonctionnel : Un moyen de le tester

# Ingénierie du besoin

## Processus

Le processus utiliser dépend du domaine de l’application.

Une vue en spirale permet de retourner les 3 grands axes.

Les différentes étapes sont :

* Spécifications du besoin.
* Étude de faisabilité
* Spécification des besoins des utilisateurs
* Prototypage
* Incitation du
* Phase de Spécifications et modélisation
* Phase de revues

## Cahier des charges et analyse

Identification des besoins :

* Faire appel à des domaines différents à ceux que l’on utilise
* Aller voir des gens, toutes les parties prenantes, personne de la maintenance, ingénieur, technicien, utilisateurs, syndicat, …

Les étapes :

* Découverte des besoins
* Classification et organisation des besoins
* Priorisation et négociation des besoins
* Spécification des besoins
  + Mise en forme dans un document.

Ces étapes ne sont pas linéaires.

Problème du cahier :

* Les parties prenantes ne savent pas ce qu’elles veulent.
  + Parfois ne savent pas ce qui existe.
* Elles n’expriment pas toujours de la bonne manière
  + Elles ont le langage de leur propre métier et pas forcément le même que celui de la personne qui développe.
* Les différentes parties prenantes peuvent avoir des besoins conflictuels.
  + Le conflit peut aussi être de pouvoir avec certaines personnes qui perdent leur pouvoir.
* Des facteur organisationnel et politique qui peuvent influencer les besoins du système.
* Les besoins changent durant le processus l’analyse. De nouvelle parties prenantes peuvent débarquer et l’environnement de business peut changer.

## Découverte des besoins

* Bien comprendre ce que veut le client
* Plusieurs solutions :
  + Interview
    - Formel ou informel avec les parties prenantes
    - Ouvert ou fermée
      * Soit des questions libres
      * Soit des questions précises,
      * Il faut garder l’esprit ouvert et faire attention à ses préjugés
    - Enregistrer l’interview (si la personne est d’accord)
    - Les explications sur leur travail sont toujours dans le cas idéal, comme le voudrait le patron. Parfois, la personne ne veut pas car elle pense qu’on l’évalue bien que cela ne soit pas le cas
    - Ne pas hésiter à poser des questions sur le métier de la personne et/ou lire des ouvrages sur le métier
  + Méthodes ethnographiques[[1]](#footnote-1)
    - Comprendre comment les gens travailles
    - Voir les gens sur place en tant qu’observateur, voir le faire même. Pour pailler le problème de l’interview. Les gens font même preuve d’initiative lors de leur travail et dont parfois certains éléments sont même possibles de mettre dans le logiciel que l’on crée.

Problème de l’observation :

* Des couts élevés
* Peut se trouver intrusif
* Peut se révéler intéressant.

La mise en forme du besoin peut passer par des scénarios qui décrit le processus que l’on souhaite faire. Cette méthode est surtout utilisée dans les méthodes Agiles. Cette méthode est surtout orientée coté utilisateur.

## Spécification des besoins

Écriture du document sous la forme Utilisateur (Simplifié) et les besoins système. Il sert aussi de contrat entre le Client et l’entreprise qui fournit le développement du logiciel.

Les différents moyens pour l’écriture d’un tel document :

* Le langage naturel :
  + Écriture de phrase pour décrire le système
  + Langage naturel structurée : Tous les besoins seront écrits de la même manière. Par exemple :
    - En tant qu’utilisateur, je dois pouvoir …
* Langages informels
  + Objectif : depuis le besoin, générer le une description.
* Langages de description
  + Graphique
  + Mathématique

Normalement, les spécifications ne doivent pas décrire des éléments du système. Certains éléments sont inséparables :

* L’architecture peut être mis
* L’interopération du système
* Les éléments que l’on veut

Le cahier des charges est souvent rédigé en langage naturel structuré. Cela ne doit pas être littéraire. Les répétitions ne sont pas gênantes. Il faut conserver le même langage tous le long du document technique.

Règles d’écriture :

* Même format tous le long du document
* Pas d’utilisation de synonyme pour les mêmes choses
* On doit utiliser doit permettre / devrait permettre pour hiérarchiser les choses
* Utiliser le gras pour les choses standards
* Ne pas utiliser du vocabulaire technique
* Pourquoi on exprime ses besoins ?

Problème avec le langage naturel :

* La clarté n’est pas toujours là
* Confusion besoin fonctionnel / besoin non-fonctionnel
* Plusieurs besoins dans un seul besoin
  + Ne pas hésiter de se faire relire dans une équipe. (Avec une peur d’empathie)

Dans le cas de système plus spécifique, le langage naturel n’est pas forcément le plus adaptée :

* Dans certains types de documents, le langage naturel n’est pas forcément adapté (description de la sortie d’un train d’atterrissage)

Besoin sous forme de formulaire :

(DIAPO 54)

* Définition de la fonction ou de l’entité

Besoin sous forme de tableau :

* Ajout langage naturel
* Pour être plus spécifique dans un élément
* Exemple :
  + Une pompe à insuline.

Le document :

* Décris ce que fait le document
* De nombreuses personnes lisent le document mais n’ont pas forcément les mêmes compétences.
  + Les clients du système
  + Les gestionnaires (ceux qui paient)
  + Ceux qui vont développer
  + Ceux qui vont faire les tests.
    - Il faut trouver dans le cahier des charges comment faire les tests.
  + Système de maintenance ingénieur (TMA)

Suivant ce que l’on fait, le cahier des charges va changer. Les spécifications seront plus précises avec des contraintes physiques par exemple alors qu’un logiciel de paie sera plus sur du droit.

La méthode Agile, ce ne sont que des petits bouts au fur et a mesure.

Template disponible sur git.

Phase de validation :

* Faire de la relecture de croisée
* Importance : Une erreur dans le cahier de charge, plus tard elle est découverte, plus chère, elle coutera.

Pour chaque besoin : (Dipo 67)

* Valide
  + Revue du besoin
  + Prototypage
  + Écriture de tests
* Pas de conflit
* Complétude
* Le réalisme
* La vérifiabilité

La méthode la plus classique : La revue du vahier des charges, elle sera faite par le client et par les personnes concernées. Elle peut-être formel ou informel.

Dans le projet, on aura :

* Verifiable
* Stable
* Compris
* Adaptable
* Traçable (pourquoi on a fait ce besoin ?
* Faire attention à ce que l’on voulait dire

Besoin de changement :

* Le business et l’environnement technique du système a toujours changé après l’installation
* La personne qui paie pour un système and les utilisateurs du système sont rarement les mêmes personnes.
* Les procédures peuvent être plus ou moins compliquée à être mises à en place en fonction du logiciel
* Mesure de l’impact du changement sur les parties utilisant le logiciel.
* Planning :
  + Il y a des outils pour suivre l’avancement des mises à jour.
  + Ces mises à jour

UML

* Réflexion du coté logiciel (plutôt orienté objet)
* Générer un maximum de code depuis ces spécifications
* Définition
  + UML se définit comme étant un langage de modélisation graphique et textuel, destinée à
    - Comprendre et décrire des besoins
    - Communiquer,
    - Spécifier,
    - Documenter les systèmes,
    - Esquisser des architectures logicielles

13 types de diagramme.

* Diagramme des cas d’usages
  + Peut-être utilisé dans un cahier des charges
  + Ils représentent l’expression des besoins fonctionnels de l’applications
    - Partie
* Diagramme de classe :
  + Description
    - Nom
    - Visibilité
    - Type
    - Cardinalité
    - Valeur par défaut
    - Autres propriétés

1. Étude des cultures, comprendre leur mode de vie, leur religion, … [↑](#footnote-ref-1)