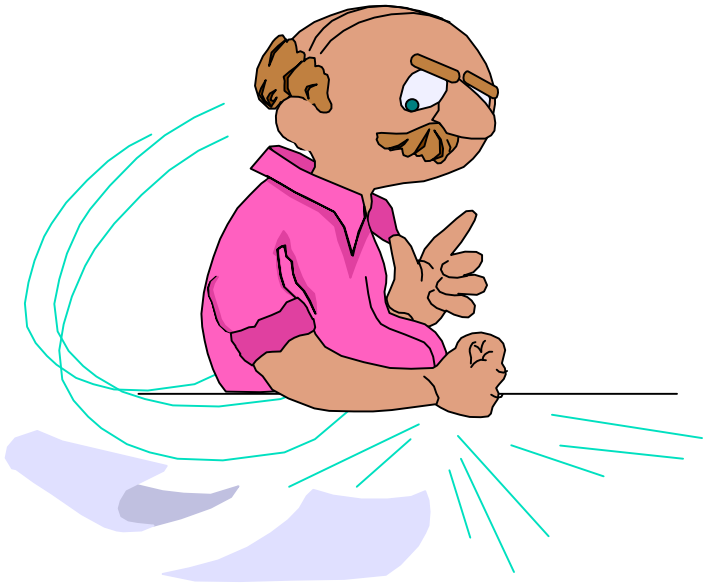
Chap 2: Qualité et Cahier de Charge



Dr. Bah Moussa Demba

La qualité en génie logiciel?

La qualité d'un logiciel est son aptitude à satisfaire les besoins des utilisateurs [Martin, 1987]

- Le logiciel est la clé de différenciation des produits industriels
- Mais on le maîtrise mal...
  - De nombreux projets informatiques n'ont jamais abouti
  - Ou ont été des catastrophes économiques
- ... contrairement au génie civil (ponts autoroutes, tunnels,...)
- Le zéro défaut n'existe pas en matière de logiciel

Personne ne sait aujourd'hui créer de logiciel sans défaut!

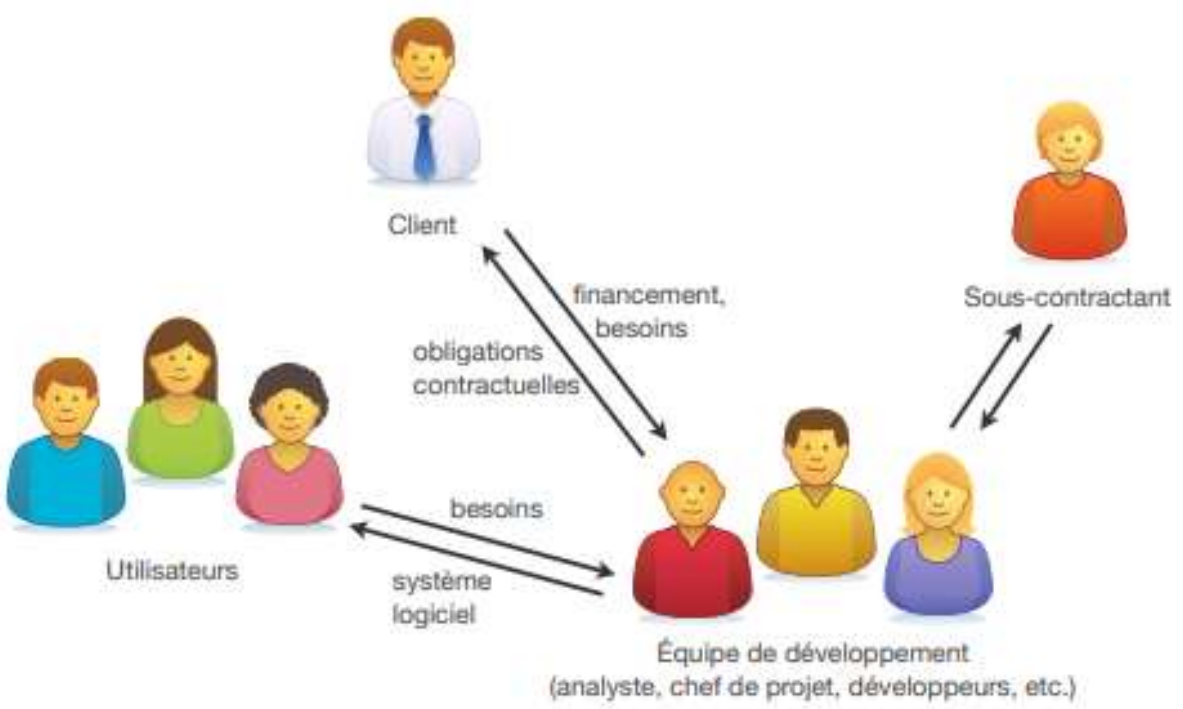
Fiabilité des logiciels

C’est la probabilité pour qu’une panne du logiciel ait lieu.

➔ Un système peu fiable entraine des pertes d’infos, pertes de temps et d’argent pour les reconstruire

Indicateurs	•Fiabilité •Maintenabilité •Performance •Portabilité •Utilisabilité

Les acteurs du génie logiciel



Principes de développement : DRY

DRY: **Don't Repeat Yourself**. *Ce principe vise à éviter la redondance au travers de l'ensemble de l'application*. Cette redondance est en effet l'un des principaux ennemis du développeur.
Elle a les consequences néfastes suivantes :

- Augmentation du volume de code.
- Diminution de sa lisibilité.
- Risque d'apparition de bugs dûs à des modifications incomplètes.

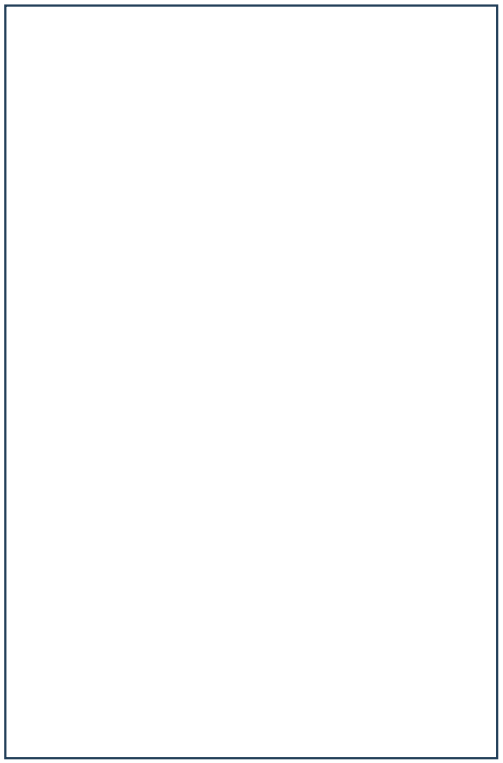
Elle prend souvent la forme d'un ensemble de lignes de code dupliquées à plusieurs endroits pour répondre au même besoin.

Exemple:

Principes de développement : DRY

```
function A() {
  // ...

  // Code dupliqué

  // ...
}

function B() {
  // ...

  // Code dupliqué

  // ...
}
```

La solution: factoriser les lignes auparavant dupliquées.



Attention:

Trop factoriser ➔ creation des applications inutilement complexes

Principes de développement : KISS

KISS: **Keep It Stupid Simple** et qu'on peut traduire par "*Ne complique pas les choses ou garde ca super simple*“ avec l'idée que « *même un idiot pourrait comprendre* ». Ce principe vise à privilégier autant que possible la simplicité lors de la construction d'une application.

Il consiste à affirmer qu'une application doit être créée *selon l'ordre de priorité* ci-dessous :

1. ***Make it work.***
2. ***Make it right.***
3. ***Make it fast***.

Principes de développement : YAGNI

Il signifie **You Ain't Gonna Need It** signifie *«vous n'en aurez pas besoin »*. Il consiste à dire que les *programmeurs* ne devraient pas ajouter de fonctionnalité à un logiciel tant que celle-ci n'est pas absolument nécessaire.

Ron Jeffries recommande par ailleurs : « *mettez toujours en œuvre les choses quand vous en avez effectivement besoin, pas lorsque vous prévoyez simplement que vous en aurez besoin* ».

Tant que la fonctionnalité n'est pas réellement nécessaire, il est difficile de définir complètement ce qu'elle doit faire, et comment la tester. Si cette nouvelle fonctionnalité n'est pas correctement définie et testée, elle risque de ne pas fonctionner correctement lorsqu'elle sera un jour nécessaire.

Spécification fonctionnelle

- **Définition** :
  - définir ce que doit faire un logiciel ...
  - avant d'écrire ce logiciel !
  - ➔ le *quoi*, sans le *comment*

- **S'oppose à implémentation**:
  - comme le plan d'un pont à sa construction.
  - Sauf... qu'en informatique, plan et réalisation ne manipulent que de l'écriture...

Pourquoi spécifier ?

- Définir le travail de réalisation, avant de le faire ...

  la spécification est censée être moins coûteuse, plus rapide à obtenir
- Comment vérifier la correction d' un programme,
  - *si l'on ne sait pas ce qu'il est censé faire ...*
  - *la spécification est la référence pour toutes les activités de vérification et de validation (V&V) : tests, preuves, mesures,...*

- Obtenir une description stable, plus abstraite, moins dépendante des contingences du matériel, des systèmes, des fluctuations de l'environnement.

Qualités d'une spécification fonctionnelle

- Précision, non ambiguïté, non contradiction,
- Concision, abstraction,
- Complétude,
- Facilité d'utilisation : écriture, lecture, vérification
- Réalisable avant l'implémentation,
- Si possible à un coût réduit,

Spécifications fonctionnelles *vs* non-fonctionnelles

- En génie logiciel, on distingue :
  - spécification fonctionnelle : décrit le « fonctionnement » du logiciel, ***le quoi***
  - spécification non-fonctionnelle: les conditions de fonctionnement, ***le comment***
- Les spécifications techniques du logiciel : *coût, temps de réponse, performance, robustesse, capacité de charge, consommation de ressource, confort d'utilisation, ergonomie ... sont appelées :*
  - qualités de service (QoS)
  - spécifications non fonctionnelles

Spécification *vs* Implémentation

- **Spécification**
  - décrit ce que doit faire le logiciel, si possible très tôt dans le processus logiciel, à un coût faible, indépendamment des « détails » :
    - ➔ *type de machine, plate-forme, langage utilisé, conditions d'utilisation,...*
  - Une spécification fonctionnelle doit se concentrer sur les fonctionnalités, sans considération de performance ou de qualités de services (spécifications non fonctionnelles)
- **Implémentation**
  - doit correspondre aux fonctionnalités décrites (idée de correction),
  - mais de manière efficace, performante, en tenant compte:

    ➔des conditions réelles ou prévues d'utilisation : *langage, spécificité du langage utilisé, volumes des données traitées...*
  - des contraintes exprimées dans les spécifications non fonctionnelles

Spécification *vs* Implémentation

- spécifier, c'est définir

  sans se soucier des contingences du monde réel...

- implémenter, c'est optimiser, réifier

  c'est tenir compte de la réalité...

Correction

- un logiciel est correct, si dans des conditions normales d'utilisation, il se comporte comme cela est attendu par ses utilisateurs
- Si la spécification traduit bien ce que veulent les utilisateurs, alors la correction revient à vérifier (prouver, tester) que l'implémentation est conforme à sa spécification

Comment rédiger les spécifications fonctionnelles ?

Étape 1 : définir le périmètre fonctionnel

Après avoir recueilli et reformulé *l'expression des besoins du client*, définissez le périmètre fonctionnel grâce à *l'**impact mapping*** en vous posant des questions telles que :

1. Quel est l'objectif du site/logiciel ?
2. Qui seront les utilisateurs finaux ?
3. Quels bénéfices recherchent-ils à travers son utilisation ?
4. Quelles fonctionnalités permettront d'y parvenir ?

Étape 2 : faire une arborescence

Structurez les spécifications à l'aide d'un ***graphique*** et faites ***l'arborescence*** du produit, pour voir comment les fonctionnalités s'articulent entre elles, et visualiser le parcours utilisateur (prémices de *l'UI design*, *design fonctionnel*).

Étape 3 : établir un niveau de priorité

Priorisez les fonctionnalités à développer en fonction de leur importance et de leurs interdépendances.

Cela permet au chef de projet de ***planifier les tâches*** et d'y ***attribuer les ressources*** nécessaires.

Étape 4 : rédiger la partie fonctionnelle de votre cahier des charges

Rédigez vos spécifications de façon structurée.

Cahier des charges en gestion de projet

Pour vous aider à établir le cahier des charges de votre projet, appvizer met à votre disposition un modèle. À vous de le compléter avec les informations concernant votre projet !

Plus d'infos : *https://www.appvizer.fr/magazine/operations/gestion-de-projet/etapes-gestion-projet*

Cahier des charges

1. Contexte du projet ou problématique à résoudre

Qu'est-ce qui motive le lancement de ce projet ? Quels sont les besoins ?

-
-

2. Objectif du projet

Quels sont les résultats attendus à l'issue du projet ?

-

-

3. Périmètre du projet

Quels sont les contours et les limites du projet ?

-

-

4. Cibles du projet

Quels sont les différents profils de vos cibles, les utilisateurs finaux ?

-

-

Cahier des charges

5. Budget du projet

Enveloppe budgétaire allouée pour le projet :

Postes de dépense à prendre en compte :

-

-

6. Planning du projet

Date de livraison du projet :

Jalons principaux :

-

-

Cahier des charges

7. Spécifications fonctionnelles

Fonctionnalité 1 :

- ➔ *Contexte*
- ➔ *Cas d'application*
- ➔ *Critères d'acceptation*
- ➔ *Niveau de priorité*

Fonctionnalité 2 :

- ➔ *Contexte*
- ➔ *Cas d'application*
- ➔ *Critères d'acceptation*
- ➔ *Niveau de priorité*

Arborescence

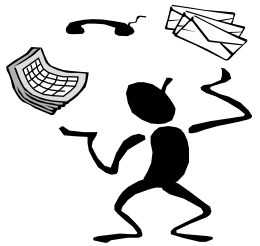
8. Spécifications techniques

Pour chaque fonctionnalité et sous-fonctionnalité :

- ➔ *Solution d'hébergement*
- ➔ *Architecture des serveurs*
- ➔ *Choix de la plateforme ou du CMS*
- ➔ *Outils d'administration*
- ➔ *Contraintes d'intégration*
- ➔ *Langage informatique*
- ➔ *Gestion de la sécurité des données*
- ➔ *Maintenance*
- ➔ *Migration*
- ➔ *Compatibilité avec les navigateurs, etc.*

9. Commentaires

Bibliographie



- J.P. Martin: « *La qualité des logiciels*», Association française de normalisation (AFNOR), Paris 1987.
- J.P. Martin: «*Qualité du logiciel et Système Qualité. L'industrialisation par la certificataion*», Masson, 1992.
- «ISO 9001 et le développement du logciel. Guide l'application », AFNOR, 1996
- C. H. Schmauch: «ISO 9000 for software developers», ASQC Quality Press, 1994
- T. Forse: «*Qualimétrie des systèmes complexes*», Les Editions d'Organisation, 1989.
- Philippe Collet:" *COO : Spécification du logiciel - OCL*", Cours Licence, universite Nice, Sophia Antipolis, 2012.
- https://www.appvizer.fr/magazine/operations/gestion-de-projet/etapes-gestion-projet

IMPACT MAPPING

4 questions essentielles

•**Pourquoi (objectif) ?**

Il s'agit de la plus importante question : Quel est l'objectif qu'on souhaite atteindre? La réponse à cette question nous permet de prendre des bonnes décisions concernant le budget, le planning…et nous permet de nous adapter correctement en cas d'imprévu. L'objectif doit être SMART (Spécifique, Mesurable, Atteignable, Réaliste et Temporellement défini).

•**Qui (ACTEUR)?**

Lister les acteurs impactés d'une manière plus ou moins directe par l'objectif précédemment identifié.

•**Comment (IMPACT) ?**

Il faut énumérer pour chacun des acteurs les modalités dans lesquelles ils peuvent être impactés ou peuvent participer pour atteindre l'objectif.

•**Quoi (fonctionnalités) ?**

L'idée est de décrire plus précisément les livrables ou les actions nécessaires pour réaliser les impacts.