Génie Logiciel - Cours 01

Régis WITZ

Définitions

Le projet

Un projet est un processus **unique**, qui consiste en un ensemble d'**activités** coordonnées et maîtrisées comportant des dates de début et de fin, entrepris dans le but d'atteindre un objectif conforme à des exigences spécifiques de **qualité** avec des contraintes de **coûts** et de **délais**.

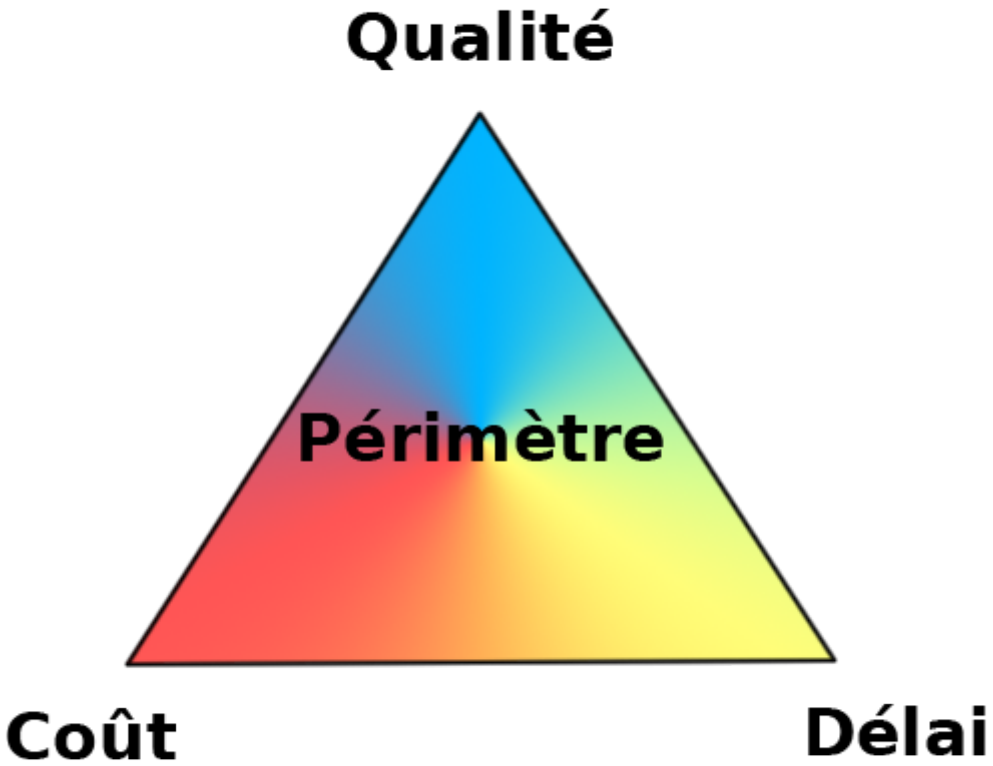

Figure 01:Qualité, Budget, Délai sont interdépendants

Ces contraintes de qualité, de coûts et de délais fixent le **périmètre** du projet. Elles sont interdépendantes : "gagner" sur l'une fait souvent "perdre" sur l'une ou l'autre.

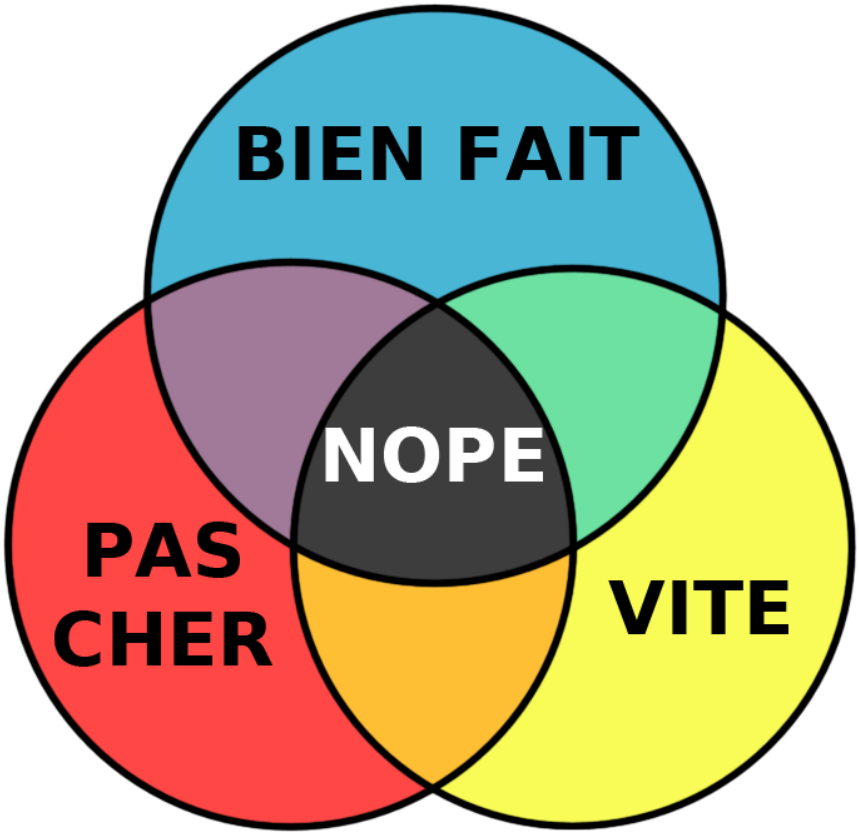

Figure 02:« Je veux le meilleur, le moins cher et je ne veux pas attendre ! »

Les activités d'un projet informatique créent des **livrables** : outils, méthodes ou services informatiques.

Les acteurs et leurs rôles

Le projet est réalisé avant tout par des hommes et des femmes. Ces personnes physiques sont réunies en *entités* qui sont autant de personnes morales (une entreprise, un service ou une équipe particuliers, …).

Chacune de ces entités assume une ou plusieurs responsabilités (planifier, concevoir, développer, tester, …) qui, si elles sont correctement menées, aboutiront au succès du projet.

Durant la phase de réalisation d'un projet, on peut distinguer deux entités particulières :

- le **client**: aussi appelé *Maîtrise d'ouvrage* (*MOA*) ou *Project owner*, il s'agit de celui qui **exprime le besoin**.
- le **fournisseur**: aussi appelé *Maîtrise d'œuvre* (*MOE*), il s'agit de celui qui **répond au besoin**.

En plus du client et de son fournisseur, d'autres entités peuvent être **partie prenante** dans le projet. Il s'agit des utilisateurs finaux de la solution, des organismes finançant le projet, etc.

La Maîtrise d’OuvrAge (MOA)

La maîtrise d’ouvrage est à l’origine du projet. C’est elle qui en fixe les objectifs que sont le **besoin** à remplir, ainsi que les **coûts** et les **délais** du projet.

C’est elle qui détient la **connaissance fonctionnelle** du domaine dans lequel s’inscrit le projet, tel que le cadre dans lequel doit exister le projet, et l’attente des fonctionnalités qu’il aportera.

Il s’agit du **propriétaire** au final du projet. La MOA est donc **celui qui paye**.

La Maîtrise d’Œuvre (MOE)

La maîtrise d’œuvre fournit le produit attendu par la maîtrise d’ouvrage. C’est elle qui assure la qualité, les coûts et les délais.

Elle peut se faire aider par un ou plusieurs fournisseurs. C’est alors à elle qu’incombe la responsabilité de coordoner et de superviser ses prestataires.

La MOE est **celui qui réalise**.

Qualité d’un logiciel

Lorsqu’on parle de la qualité d’un logiciel, on parle en fait de plusieurs **caractéristiques** différentes, elles-mêmes décomposées en sous-caractéristiques (ou attributs).

Toutes les caractéristiques d’un logiciel n’ont pas forcément la même importance pour une organisation donnée. C’est pourquoi chaque organisation définit son propre **modèle de qualité**, avec parfois plusieurs modèles applicables à ses différents produits logiciels. En particulier, chaque organisation définit ses propres méthodes (métriques) pour évaluer le **degré de présence** des différents attributs de qualité.

Les principales **normes** utilisées pour la définition de modèle(s) de qualité : ISO/IEC 9126 (ancienne version) et ISO/IEC 25010 (nouvelle version).

Dans ces normes, le terme *logiciel* est employé au sens large. Il peut désigner à la fois le code source, les exécutables, les librairies, les documents d’architecture, et ainsi de suite. En conséquence, la notion d'*utilisateur* peut elle aussi s’appliquer, suivant la caractéristique et/ou le logiciel considéré, à un client ou à un développeur, par exemple.

Caractéristiques

- **Capacité fonctionnelle** (Functional suitability)
  Est-ce que le logiciel répond aux besoins fonctionnels exprimés ?
  - *Pertinence* (Suitability / Functional appropriateness)
    Est-ce que toutes les fonctionnalités présentes sont adéquates ?
  - *Exactitude* (Accuracy / Functional Correctness)
    Est-ce que les résultats ou effets fournis sont corrects ?
  - *Complétude* (Functional completeness)
    Est-ce que les résultats ou effets fournis sont suffisants ?
  - *Conformité* (Functionality compliance)
    Est-ce que le logiciel respecte les normes et règlements en vigueur ?
- **Fiabilité** (Reliability)
  Est-ce que le logiciel maintient son niveau de service dans des conditions précises et pendant une période déterminée ?
  - *Maturité* (Maturity)
    Est-ce que la fréquence d'apparition des incidents est faible ?
  - *Tolérance aux pannes* (Fault tolerance)
    Est-ce que l'apparition d'un incident dégrade le comportement du logiciel ?
  - *Facilité de récupération* (Recoverability)
    Quel est la capacité du logiciel à retourner dans un état opérationnel complet après une panne ?
  - *Disponibilité (Availability)*
    Est-ce que les fonctionnalités du logiciel sont disponibles en permanence ?
  - *Conformité* (Reliability compliance)
    Est-ce que le logiciel respecte les normes et règlements en vigueur ?
- **Ergonomie** (Usability)
  Effectivité, efficacité et satisfaction avec laquelle des utilisateurs spécifiés accomplissent des objectifs spécifiés dans un environnement particulier.
  - *Facilité de compréhension* (Understandability / appropriateness recognizability)
    Est-ce qu'il est (intuitivement) facile pour un utilisateur de comprendre et retenir comment réaliser une opération ?
  - *Facilité d'apprentissage* (Learnability)
    Est-ce qu'il est facile de former de futurs utilisateurs à l'utilisation du logiciel ?
  - *Facilité d'exploitation* (Operability)
    Est-ce qu'il est facile pour l'utilisateur d'accomplir sa tâche ?
  - *Protection contre les erreurs d'utilisation* (User error protection)
    Comment réagit le logiciel si son utilisateur a un comportement inatendu ?
  - *Accessibilité* (Accessibility)
    Est-ce que le logiciel est utilisable par des personnes en situation de handicap ?

- *Attrait* (Attractiveness / user interface aesthetics)
  Est-ce que l'utilisateur a envie d'utiliser l'interface ?
- *Conformité* (Usability compliance)
  Est-ce que le logiciel respecte les normes et règlements en vigueur ?

- **Rendement** (Performance Efficiency)
  Est-ce que le logiciel requiert un dimensionnement rentable et proportionné de la plate-forme d'hébergement en regard des autres exigences ?
  - *Comportement temporel* (Time behaviour)
    Est-ce que le temps de réponse du logiciel varie en fonction de son usage ?
  - *Utilisation des ressources* (Resource utilization)
    Est-ce que les resources (mémoire, processeur, …) sont correctement utilisées par le logiciel ?
  - *Capacité* (Capacity)
    Est-ce que les ressources fournies par le logiciel le sont au bon moment et en quantité adéquate ?
  - *Conformité* (Efficiency compliance)
    Est-ce que le logiciel respecte les normes et règlements en vigueur ?

- **Maintenabilité** (Maintainability)
  Est-ce que faire évoluer le logiciel est aisé en cas d'anomalie ou de nouveaux besoins ?
  - *Facilité d'analyse* (Analyzability)
    Est-ce qu'il est facile d'identifier les déficiences du logiciel ou leur cause ?
  - *Facilité de modification* (Modifiability)
    Quel est l'effort nécessaire pour remédier à ces déficiences ?
  - *Testabilité* (Testability)
    Quel est l'effort nécessaire pour valider le comportement du logiciel ?
  - *Modularité* (modularity)
    Est-ce que modifier un composant du logiciel impacte peu les autres composants ?
  - *Ré-utilisabilité* (Reusability)
    Est-ce qu'il est facile de combiner des composants existants pour obtenir une nouvelle fonctionnalité ?
  - *Conformité* (Maintainability compliance)
    Est-ce que le logiciel respecte les normes et règlements en vigueur ?

- **Portabilité** (Portability)
  Est-ce que le logiciel peut être transféré d'une plate-forme ou d'un environnement à un autre ?
  - *Facilité d'adaptation* (Adaptability)
    Est-ce qu'il est facile d'adapter le logiciel à des changements d'environnement opérationnel ?
  - *Facilité d'installation* (Installability)
    Est-ce qu'il est facile d'installer le logiciel dans un environnement prévu ?
  - *Interchangeabilité* (Replaceability)

Est-ce qu'il est facile d'utiliser ce logiciel à la place d'un autre dans un même environnement ?

- *Conformité* (Portability compliance)
  Est-ce que le logiciel respecte les normes et règlements en vigueur ?

- **Compatibilité** (Compatibility) Est-ce que le logiciel fonctionne bien avec d'autres ?
  - *Coexistence* (Co-existence)
    Est-ce que le fonctionnement du logiciel altère ou est altéré par un autre ?
  - *Interopérabilité* (Interoperability)
    Est-ce que le logiciel peut communiquer avec d'autres ?
- **Sécurité** (Security) Est-ce qu'on peut faire confiance au logiciel ?
  - *Confidentialité* (Confidentiality)
    Est ce que les données ne sont accessibles qu'à ceux qui y sont autorisés ?
  - *Intégrité* (Integrity)
    Est-ce que les donnés sont garanties contre une modification non autorisée ?
  - *Non-répudiation* (Non-repudiation)
    Est-ce que le logiciel peut prouver que chaque action a bien eu lieu ?
  - *Responsabilité* (Accountability)
    Est-ce que le logiciel peut relier chaque action à son auteur ?
  - *Authentification* (Authenticity)
    Est-ce que le logiciel peut empêcher une identité d'être usurpée ?

La « Sur-qualité »

On appelle parfois « sur-qualité » le fait de réaliser une fonctionnalité alors que celle-ci n'est pas spécifiée.

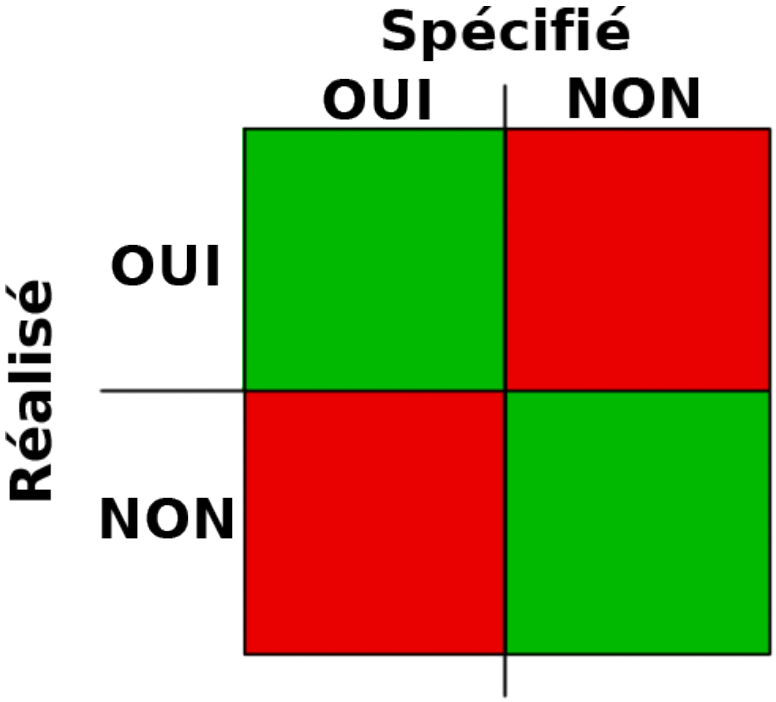

Figure 03:« Sur-qualité »

L'interprétation habituelle est alors la suivante :

- Réaliser une fonctionnalité demandée est normal : c'est ce en quoi consiste la qualité logicielle.
- Ne pas réaliser une fonctionnalité alors qu'elle est demandée dégrade la qualité du logiciel.
- Ne pas réaliser une fonctionnalité qui n'est pas demandée est normal.
- Mais réaliser une fonctionnalité alors que celle-ci n'est pas demandée est une perte ou un manque à gagner pour le fournisseur, étant donné que le client ne va pas payer pour obtenir quelque chose qu'il n'a pas demandé !

Bien que tout à fait défendable, cette façon de résumer la « sur-qualité » est limitative. Il est possible d'interpréter la situation de manière plus nuancée, en se posant la question : la fonctionnalité considérée est elle *utile* ou *inutile* pour le client ? Cette question exprime parfaitement la différence entre le besoin tel qu'il est *exprimé* par le client et le besoin *réel* du client (voir la phase d'analyse du besoin).

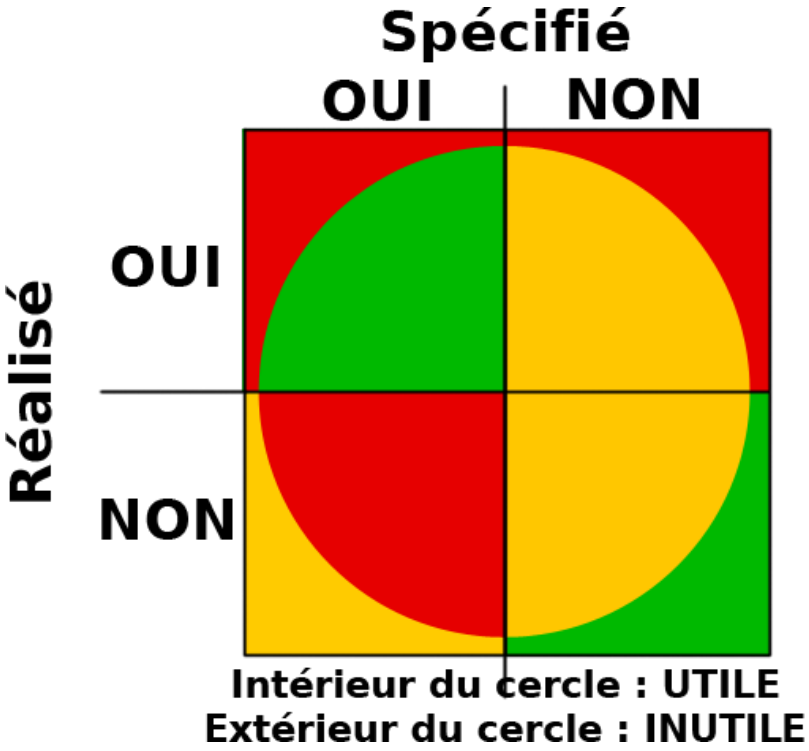

Figure 04:Une vision alternative de la « Sur-qualité »

On a alors l'interprétation suivante :

- Réaliser une fonctionnalité demandée est normal *si celle-ci est utile*.
- Réaliser une fonctionnalité demandée mais *inutile* peut signifier un manque lors de la phase d'analyse du besoin.
- Ne pas réaliser une fonctionnalitée alors que celle-ci est demandée est une faute du fournisseur.
    - Cependant, ci la fonctionnalité non réalisée s'avère inutile pour le client, celui-ci ne sera peut-être pas gêné par son absence.
- Ne pas réaliser une fonctionnalité qui n'est pas demandée est normal.
    - Mais si la fonctionnalité non réalisée aurait été finalement utile au client ? Il s'agit-là d'une opportunité d'amélioration du produit …
- Réaliser une fonctionnalité alors que celle-ci n'est pas demandée est:
    - une perte pour le fournisseur, effectivement, si la fonctionnalité en question est inutile.
    - une manière d'aller au devant du besoin du client, et donc une preuve de la qualité du service fourni par le fournisseur, qui apporte la preuve qu'il a mieux saisi le besoin réel du client que le client lui-même.

Avec ce point de vue alternatif, la « sur-qualité » peut donc devenir un atout commercial.