Partie 2 : exigences

Partie 2 : lien vers la présentation

https://drive.google.com/file/d/1dbgRVV5FwGLhWjL4iOy97qFDSpuxaVpx/view?usp=sharing

Architecture logicielle et exigences

Deux types d'exigences : fonctionnelles et non fonctionnelles

- exigence fonctionnelle : exigence issue de la compréhension du besoin métier

- exigence non fonctionnelle : tout ce qui n'est pas lié directement aux fonctionnalités principales du système mais néanmoins indispensable au bon fonctionnement du système.
  les exigences non fonctionnelles comprennent les aspects performance, sécurité, réseau, disponibilité, exploitation, charge, terminal utilisateur

Architecture logicielle et exigences

Les exigences non fonctionnelles doivent toujours être décrites en termes clairs, précis et non ambigus. Elles doivent être quantifiées.

Exemple : le système doit pouvoir traiter 100000 utilisateurs simultanément avec un temps de réponse inférieur à 2 secondes par utilisateur.

Les termes génériques et subjectifs sont à proscrire ou à définir clairement. Par exemple, les termes « ergonomique », « qualitatif », « de niveau acceptable pour l'utilisateur » ne veulent rien dire s'ils ne sont pas expliqués et quantifiés.

Exigences non fonctionnelles : exemples (1 / 8)

- Contraintes pesant sur le projet :
  - prix maximum de la solution à développer, budget
  - ressources humaines et matérielles imposées,
  - standards internes à l'entreprise
  - deadline liée à un événement commercial

Exigences non fonctionnelles : exemples (2 / 8)

- Conformité du système à un environnement :
  - normes réglementaires (ex: RGPD, SOX, BALE)
  - documentaires,
  - conformité aux licences acquises

Exigences non fonctionnelles : exemples (3 / 8)

- Maintenabilité du système : est-ce que le logiciel requiert peu d'effort à son évolution par rapport aux nouveaux besoins ?
    - Facilité d'analyse : identification dans le logiciel de l'origine d'un défaut constaté
    - Facilité de modification
    - Stabilité
    - Testabilité

Exigences non fonctionnelles : exemples (4 / 8)

- Rendement et efficacité : est-ce que le logiciel requiert un dimensionnement rentable et proportionné de la plate-forme d'hébergement en regard des autres exigences ?
    - Comportement temporel : temps de réponse, taux de transactions
    - Utilisation des ressources : mémoire, processeur, disque et réseau

Exigences non fonctionnelles : exemples (5 / 8)

- Fiabilité du système : est-ce que le logiciel maintient son niveau de service dans des conditions précises et pendant une période déterminée ?
    - Maturité (faible fréquence d'apparition des incidents)
    - Tolérance aux pannes
    - Facilité de récupération : capacité d'un logiciel défectueux à retourner dans un état opérationnel complet (données et connexions réseaux incluses)
    - Disponibilité

Exigences non fonctionnelles : exemples (6 / 8)

- Portabilité du système : est-ce que le logiciel peut être transféré d'une plate-forme ou d'un environnement à un autre ?
    - Facilité d'adaptation à des changements de spécifications ou d'environnements opérationnels
    - Facilité d'installation
    - Coexistence
    - Interchangeabilité : utilisation de greffons (plugins)

Exigences non fonctionnelles : exemples (7 / 8)

- Sécurité du système :
  - traçage des mises à jour des données dans le système,
  - gestion de la confidentialité,
  - gestion de l'intégrité des données,
  - protection des données personnelles

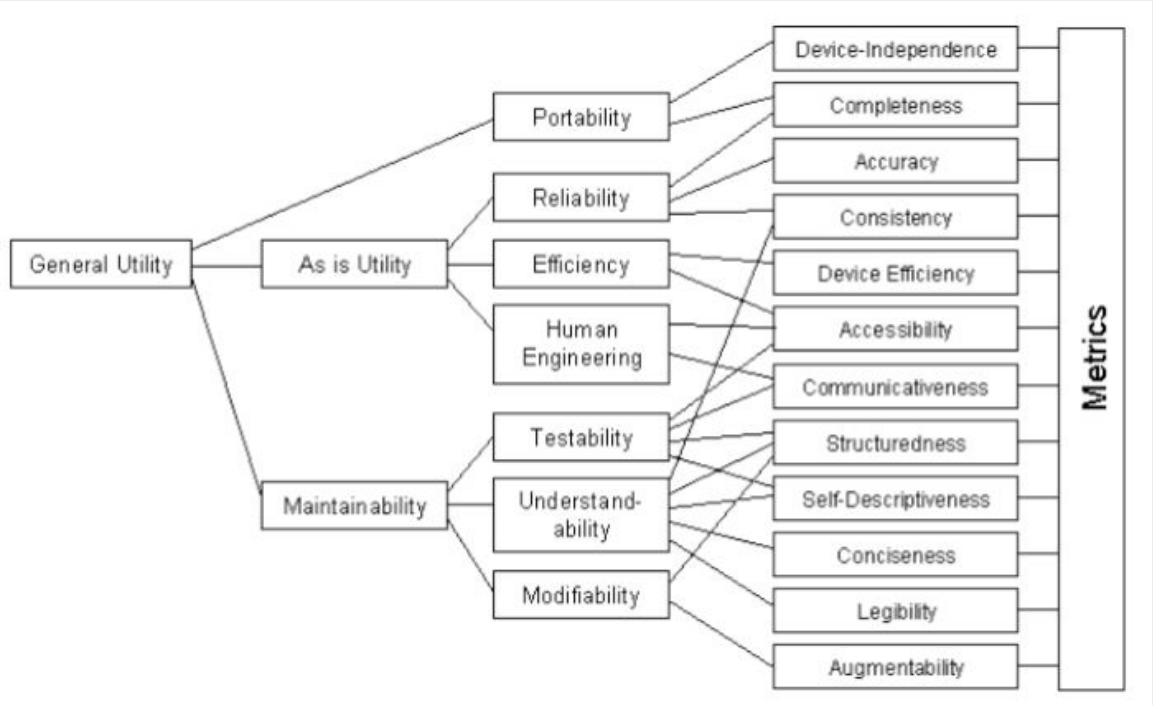
Exigences non fonctionnelles : exemples (8 / 8)

- Utilisabilité : est-ce que le logiciel requiert peu d'effort à l'utilisation ?
    - facilité d'utilisation en limitant le nombre de clic à maximum 3 clics pour finaliser la transaction,
    - facilité de compréhension,
    - facilité d'apprentissage,
    - facilité d'exploitation

Exigences non fonctionnelles : quelques normes

- Plusieurs normes et référentiels existent
  - arbre de qualité de Boehm
  - norme ISO 9126 puis ISO 25010
  - classification de Sommerville
  - CMMI : Capability Maturity Model Integration : dépasse le cadre des exigences.
    CMMI définit la capacité des organisations à mener des projets informatiques

Exigences non fonctionnelles : arbre de qualité de Boehm

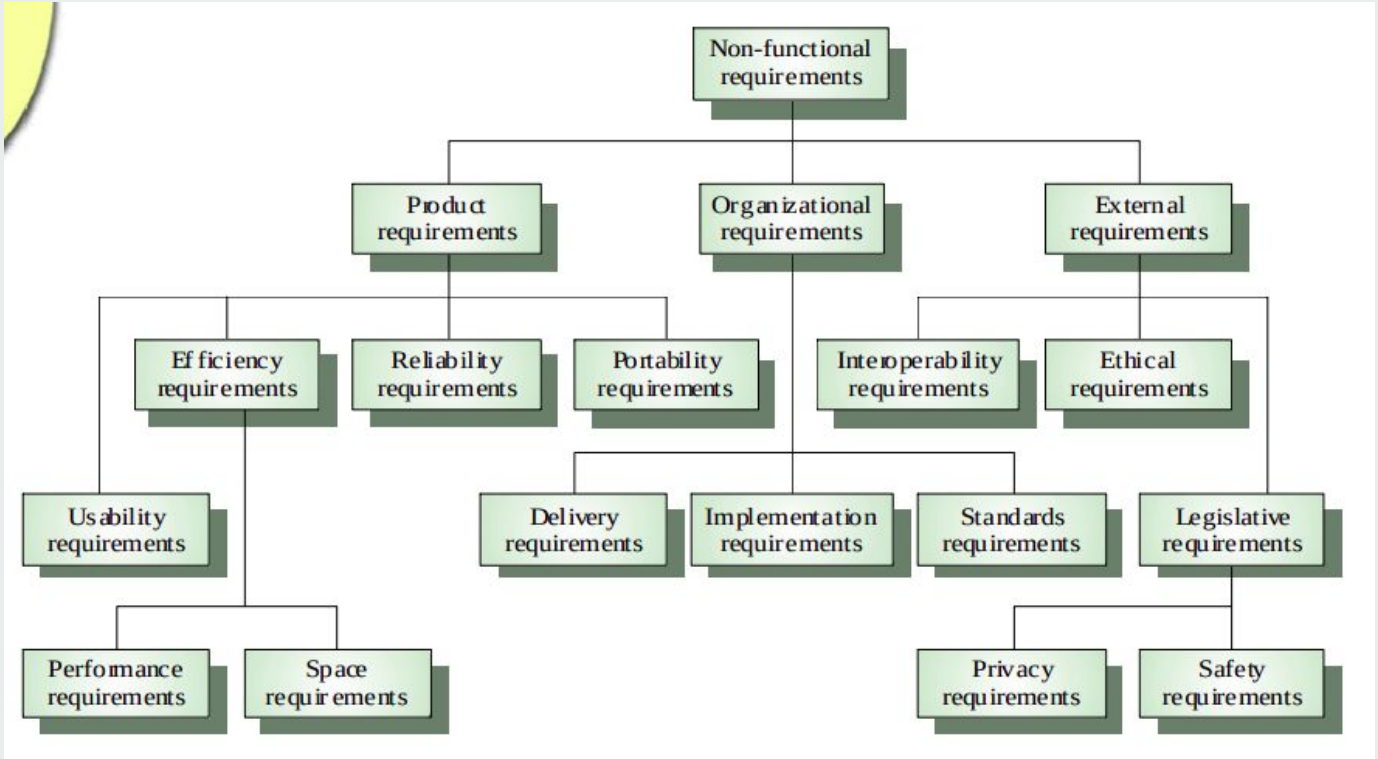


Exigences non fonctionnelles : ISO 25010

SOFTWARE PRODUCT QUALITY								
FUNCTIONAL SUITABILITY	PERFORMANCE EFFICIENCY	COMPATIBILITY	INTERACTION CAPABILITY	RELIABILITY	SECURITY	MAINTAINABILITY	FLEXIBILITY	SAFETY
FUNCTIONAL COMPLETENESS FUNCTIONAL CORRECTNESS FUNCTIONAL APPROPRIATENESS iso25000.com	TIME BEHAVIOUR RESOURCE UTILIZATION CAPACITY	CO-EXISTENCE INTEROPERABILITY	APPROPRIATENESS RECOGNIZABILITY LEARNABILITY OPERABILITY USER ERROR PROTECTION USER ENGAGEMENT INCLUSIVITY USER ASSISTANCE SELF-DESCRIPTIVENESS	FAULTLESSNESS AVAILABILITY FAULT TOLERANCE RECOVERABILITY	CONFIDENTIALITY INTEGRITY NON-REPUDIATION ACCOUNTABILITY AUTHENTICITY RESISTANCE	MODULARITY REUSABILITY ANALYSABILITY MODIFIABILITY TESTABILITY	ADAPTABILITY SCALABILITY INSTALLABILITY REPLACEABILITY	OPERATIONAL CONSTRAINT RISK IDENTIFICATION FAIL SAFE HAZARD WARNING SAFE INTEGRATION

https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

Exigences non fonctionnelles : Sommerville



CMMI (1 / 3)


Maturité
+
5 OPTIMISÉ
Processus stable & flexible
4 QUANTIFIÉ
Processus mesuré et contrôlé
3 DÉFINI
Processus organisé & maîtrisé
2 GÉRÉ
Processus planifié, exécuté & évalué
1 INITIAL
Processus aléatoire
© www.manager-go.com
-

CMMI (2 / 3)

Niveau 1 Initial	Les processus sont aléatoires et « réactifs », ce qui accroît les risques de sous-qualité et de dérapages. La réussite des projets repose davantage sur les capacités individuelles des porteurs de projets que sur des efforts collectifs et coordonnés.
Niveau 2 Géré (managed)	Un certain niveau de management de projet a été atteint. Les projets sont planifiés, exécutés et évalués, mais de nombreux éléments restent à améliorer. Les processus commencent à devenir réplicables.
Niveau 3 Définis	Les organisations qui ont atteint ce niveau sont plus proactives que réactives. Des standards à l'échelle de l'organisation sont mis en œuvre pour guider les projets. Les différentes entités identifient leurs lacunes et leurs cibles d'amélioration.

CMMI (3 / 3)

Niveau 4 Gérés quantitativement	La mesure des écarts et leur pilotage ont été améliorés. L'organisation travaille à partir de données quantitatives pour établir des processus bien identifiés qui correspondent aux besoins des acteurs. Les risques sont anticipés et l'on dispose de davantage de données sur les déficiences éventuelles.
Niveau 5 Optimisé ou en cours d'optimisation	On atteint le stade ultime de l'amélioration continue, où les processus sont stables et flexibles. C'est le terreau idéal pour mettre en œuvre des pratiques « agiles » et innovantes, dans un environnement de mieux en mieux maîtrisé.

comparaison exigences fonctionnelles et non fonctionnelles (1 / 2)

Exigences fonctionnelles	Exigences non fonctionnelles
Une exigence fonctionnelle définit un système ou son composant.	Une exigence non fonctionnelle définit l'attribut de qualité d'un système logiciel.
Une exigence fonctionnelle « Que doit faire le système logiciel ? »	Une exigence non fonctionnelle impose des contraintes sur « Comment le système logiciel doit-il répondre aux exigences fonctionnelles ? »
L'exigence fonctionnelle est spécifiée par le donneur d'ordre.	L'exigence non fonctionnelle est spécifiée par des personnes techniques, par exemple l'architecte, les responsables techniques et les développeurs de logiciels.
Définies au niveau d'un composant.	Appliquées à un système dans son ensemble.

comparaison exigences fonctionnelles et non fonctionnelles (2 / 2)

Exigences fonctionnelles	Exigences non fonctionnelles
Aident à vérifier la fonctionnalité du logiciel.	Aident à vérifier les performances du logiciel.
Des tests fonctionnels tels que le système, l'intégration, de bout en bout, les tests d'API, etc. sont effectués.	Des tests non fonctionnels tels que les tests de performance, de stress, d'utilisabilité, de sécurité, etc. sont effectués.

Exigences non fonctionnelles

- L'identification des exigences non fonctionnelles est très importante pour la qualité du futur système.

- Certaines devront faire l'objet d'hypothèses de la part de l'équipe de développement, en particulier, de la part de l'architecte logiciel.

Les outils

- De nombreux outils de gestion des exigences (requirements management) existent, pouvant offrir toutes ou une partie des fonctionnalités suivantes :
  - Définition des priorités
  - Rapports et analyses
  - Suivi des statuts
  - Gestion des tâches
  - Gestion des changements
  - Gestion des documents
  - Budgétisation et prévision
  - Gestion du cycle de vie
  - Gestion des flux de travail

Les outils

- Quelques outils utilisés en entreprise :
  - ReqSuite RM
  - Visure Requirements
  - Monday Dev
  - Atlassian Confluence
  - TraceCloud
  - JetBrains Space
  - Atlassian Jira
  - IBM Doors
  - Enterprise Architect

Quiz 2

https://docs.google.com/forms/d/e/1FAIpQLSfri0SBikVttCwodFG8uvhDDDq54UHD9VwBSe0ZbPsmaIt24Q/viewform?usp=dialog