Modélisation des Systèmes d ’information

Yamouni Khelifi Nassima

Table des matières

I Ingénierie des exigences

1. Exigences

☐ *Définition* :

Exigence : **Condition** ou **capacité** que doit présenter un système pour satisfaire un contrat, un standard, une spécification ou tout autre document formel imposé ».

L'exigence est donc :

•un contrat entre un fournisseur et son client.

•doit être décrite sous la forme d'une action que l'on veut faire

☐ *Définition* : Selon la norme IEEE 830-1993 : Systems and software engineering — Life cycle processes — Requirements engineering

ne exigence est définie comme étant :

(1) une condition ou capacité dont un utilisateur a besoin pour résoudre un

problème ou atteindre un objectif;

(2) une condition ou capacité qui doit être satisfaite ou possédée par un système pour satisfaire un contrat, une norme, une spécification, ou tout autre document formellement imposé.

2. Ingénierie des exigences

☐ *Définition* :

L'ingénierie des exigences est la discipline qui consiste à identifier, analyser, documenter, valider et gérer les besoins exprimés par les parties prenantes d'un projet logiciel.

Elle permet de transformer des besoins flous en spécifications claires et mesurables, servant de base à la conception et au développement.

" A capability that the system must deliver or a condition that it must be satisfied in order to address a need of a stakeholder."

[Larman, 2002]

2.1. Les acteurs impliqués dans l'ingénierie des exigences

Les acteurs impliqués dans l'ingénierie des exigences

Type d'acteur	Rôle dans le projet
Client	Exprime le besoin, valide les livrables
Utilisateur final	Utilise le système au quotidien
Analyste des exigences	Interprète les besoins et les formalise
Chef de projet	Planifie et valide les étapes
Développeur / Architecte	Met en œuvre les exigences techniques

2.2. Cycle de vie de l'ingénierie des exigences

Cycle de vie L'ingénierie des exigences

Critère	Type	Description	Exemple
Domaine	***Exigences de sécurité***	Confidentialité, intégrité, disponibilité	« Les données des clients doivent être chiffrées. »
	Exigences de performance	Temps de réponse, charge maximale	« Le système doit supporter 500 connexions simultanées. »
	Exigences d'ergonomie	Interface, accessibilité	« L'interface doit être lisible sur mobile. »
	Exigences de maintenance	Facilité d'évolution, modularité	« Le code doit suivre une architecture en couches. »

2.3. Méthodes d'élicitation des exigences

Méthodes d'élicitation des exigences

Méthode	Description	Exemple
Entretien	Discussion directe avec un acteur	"Comment passez-vous vos commandes aujourd'hui ?"
Observation	Étudier les pratiques réelles	Observer un employé qui utilise Excel pour ses stocks
Brainstorming	Réunion collaborative	Idées de fonctionnalités web B2B
Questionnaire	Collecte rapide d'opinions	Formulaire en ligne pour les entreprises clientes
Analyse documentaire	Étude d'un cahier des charges ou d'un site existant	Analyser Jumia Business, Alibaba, etc.

2.4. Validation des exigences

Méthodes

Relecture croisée avec les parties prenantes.

- Prototypage rapide : maquette d'écran.
- Cas de test d'acceptation (validation fonctionnelle).
- Revues d'exigences (inspection formelle du document).

2.5. Normes et standards officiels

Normes et standards officiels en Ingénierie des exigences

Référence	Contenu
IEEE 830 (remplacé par 29148)	Modèle standard pour les spécifications des exigences (SRS)
ISO/IEC 12207	Processus du cycle de vie logiciel
ISO/IEC/IEEE 15288	Ingénierie système
BABOK v3 (Business Analysis Body of Knowledge)	Référence pour l'analyse métier
CMMI-DEV v2.0	Maturité du processus d'ingénierie

2.6. Caractéristiques d'une bonne exigence

Caractéristiques d'une bonne exigence

Une exigence doit être:

Correcte : correspond à un besoin réel et nécessaire

Atomique : n'exprime qu'un seul fait

non ambigüe: une seule interprétation possible

Complète : énoncée entièrement en un seul endroit

Cohérente : sans contradiction avec d'autres exigences

Evaluée : négociée, priorisée, pertinente - stabilité du besoin

Traçable : identifiant unique + trace de toute modification

Vérifiable : que l'on peut contrôler, qualifier

Critère de qualité	Définition / Explication	Exemple correct	Exemple incorrect
Correcte	L'exigence répond à un besoin réel et nécessaire.	« Le système doit permettre aux utilisateurs de réinitialiser leur mot de passe oublié. »	« Le système doit afficher un arc-en-ciel à chaque connexion. » *(inutile / non lié au besoin)*
Atomique	L'exigence exprime **une seule idée** (non composée).	« Le système doit permettre à l'utilisateur d'ajouter un produit au panier. »	« Le système doit permettre à l'utilisateur d'ajouter un produit au panier et d'imprimer la facture. » *(2 exigences en une)*
Non ambiguë	L'exigence ne doit avoir **qu'une seule interprétation possible.**	« Le système doit répondre à chaque requête en moins de 2 secondes. »	« Le système doit répondre rapidement aux requêtes. » *("rapidement" est vague)*
Complète	L'exigence contient **toutes les informations nécessaires** (conditions, acteurs, contexte).	« Le système doit envoyer un e-mail de confirmation à l'adresse fournie par l'utilisateur après chaque commande validée. »	« Le système doit envoyer un e-mail. » *(incomplet : quand, à qui, pourquoi ?)*
Cohérente	L'exigence **ne contredit pas** d'autres exigences.	E1 : « Le système doit autoriser la création d'un compte après validation du courriel. » E2 : « La création de compte n'est possible qu'après validation du courriel. » *(cohérentes entre elles)*	E1 : « Le système doit autoriser la création de compte sans validation du courriel. » E2 : « Le compte ne peut être créé qu'après validation. » *(contradiction)*
Évaluée (pertinente, priorisée)	L'exigence a été discutée, acceptée, priorisée selon la méthode (MoSCoW, par ex.).	**Priorité : Must** – « Le système doit permettre la connexion sécurisée des utilisateurs enregistrés. »	Aucune évaluation : « Le système doit afficher un logo animé. » *(non prioritaire, gadget)*
Traçable	Chaque exigence a un **identifiant unique** et un **lien de traçabilité** (vers tests, code, version).	**REQ-001 :** Connexion utilisateur → Test T-001, Cas d'utilisation UC-01	« L'utilisateur doit pouvoir se connecter. » *(aucun identifiant ni référence)*
Vérifiable (testable)	On peut **mesurer ou tester** si l'exigence est respectée.	« Le système doit traiter 100 requêtes simultanées sans perte de performance. »	« Le système doit être performant. » *(impossible à tester sans critère)*

2.7. Partie prenante (Stakeholder)

Un **stakeholder** est toute personne, groupe ou organisation qui influence, est concernée par, ou affectée par le système ou le projet.

Autrement dit, ce sont tous les acteurs humains (ou organisations) ayant un intérêt, un rôle ou une responsabilité vis-à-vis du produit ou du système développé.

Définition officielle (ISO/IEC/IEEE 29148:2018):

« A stakeholder is an individual or organization having a right, share, claim, or interest in a system or in its possession of characteristics that meet their needs and expectations. »

Traduction :

« Un stakeholder est un individu ou une organisation ayant un droit, un intérêt ou une attente concernant un système ou ses caractéristiques. »

2.8. Rôles en ingénierie des exigences

Les stakeholders sont la source principale des exigences.

Ce sont eux qui expriment :

- Les besoins,
- Les contraintes,
- Les priorités,
- Les critères de satisfaction.

Sans identification claire des stakeholders, on risque :

- Des besoins oubliés,

- Des exigences contradictoires,
- Un produit non accepté par les utilisateurs finaux.

2.9. Exemples des stakholders

- Clients / investisseur
- Acheteurs
- Utilisateur final
- Experts du domaine
- Les fournisseurs de contenu
- Développeurs, Ingénieurs logiciel, gestionnaires de projets, …
- Inspecteurs (reviewer)
- Experts d'un autre système en liaison avec le projet
- Tout autre personne qui apporte une valeur ajoutée au futur système

2.10. Acteurs (stakholders)en Ingénierie des exigences

Type d'acteur	Rôle dans le projet
Client	Exprime le besoin, valide les livrables
Utilisateur final	Utilise le système au quotidien
Analyste des exigences	Interprète les besoins et les formalise
Chef de projet	Planifie et valide les étapes
Développeur / Architecte	Met en œuvre les exigences techniques

3. Types des exigences

Les exigences peuvent être classées selon plusieurs critères (ou aspects de vu).

3.1. 1. Classification principale selon le contenu ou la nature

Classification principale selon le contenu ou la nature

C'est la classification la plus connue et utilisée (IEEE, ISO 29148).

Exigences fonctionnelles : Elles répondent à la question : comment le système .

Exigences non-fonctionnelles : garantir la qualité et les contraintes du système doit être ?

Critère	Type	Description	Exemple
Nature / Contenu	***Exigences fonctionnelles***	Décrivent **les fonctions, services, comportements** que le système doit offrir	« Le client peut réserver un taxi. »
	Exigences non fonctionnelles	Décrivent **les propriétés de qualité, contraintes** ou **performances** du système	« Le temps de réponse doit être < 2 secondes. »

3.2. 2. Classification selon le niveau d'abstraction ou le point de vue

Classification selon le niveau d'abstraction ou le point de vue

Critère	Type	Description	Exemple
Niveau	***Exigences métier***	Objectifs organisationnels ou besoins de haut niveau	« Réduire les délais de transport en ville. »
	Exigences utilisateur	Ce que les utilisateurs veulent faire avec le système	« Le client doit pouvoir suivre son chauffeur en temps réel. »
	Exigences système	Ce que le système doit accomplir techniquement	« Le GPS doit mettre à jour la position toutes les 5 secondes. »

3.3. 3. Classification selon l'origine ou la source

Critère	Type	Description	Exemple
Source	*Exigences internes*	Issues de l'organisation ou de l'équipe projet	« Le système doit s'intégrer avec notre base client existante. »
	Exigences externes	Venant de l'extérieur : clients, utilisateurs, lois, partenaires	« Le système doit être conforme au RGPD. »

3.4. 4. Classification selon la stabilité ou l'évolution

Classification selon la stabilité ou l'évolution

Critère	Type	Description	Exemple
Stabilité	*Stables*	Peu susceptibles de changer pendant le projet	« L'application doit fonctionner sur Android et iOS. »
	Volatiles	Risquent de changer selon les conditions ou retours utilisateurs	« L'interface doit adopter les couleurs de la marque (en cours de refonte). »

Utile pour la gestion des versions et la priorisation.

3.5. 5. Classification selon la priorité ou la criticité

Catégorie	Signification	Exemple
M (Must have)	Indispensable pour le fonctionnement du système	Authentification utilisateur
S (Should have)	Important, mais non vital	Notification en temps réel
C (Could have)	Optionnel si le temps le permet	Mode sombre
W (Won't have)	Hors périmètre de la version actuelle	Chat vocal

Type d'exigence	Impact sur la satisfaction	Exemple
Basique	Attendue, son absence cause de l'insatisfaction	Interface claire
Performance	Plus elle est réalisée, plus le client est satisfait	Vitesse de livraison
Attractive	Surprend positivement, crée un effet "wow"	Recommandation automatique

Méthodes de priorisation comme MoSCoW ou Kano ou 100 points

3.6. 6. Classification selon le domaine concerné

Classification selon le domaine concerné

Critère	Type	Description	Exemple
Domaine	*Exigences de sécurité*	Confidentialité, intégrité, disponibilité	« Les données des clients doivent être chiffrées. »
	Exigences de performance	Temps de réponse, charge maximale	« Le système doit supporter 500 connexions simultanées. »
	Exigences d'ergonomie	Interface, accessibilité	« L'interface doit être lisible sur mobile. »
	Exigences de maintenance	Facilité d'évolution, modularité	« Le code doit suivre une architecture en couches. »

Souvent listées dans les normes ISO 9126 et ISO/IEC 25010 (qualité logicielle)

3.7. 7. Classification selon le contexte ou l'environnement}

Classification selon le contexte ou l'environnement

Critère	Type	Description	Exemple
Contexte	*Exigences d'interface*	Interaction entre systèmes	« Le système doit communiquer avec la base de données de facturation. »
	Exigences d'environnement	Contraintes matérielles ou logicielles	« Le système doit être compatible avec PostgreSQL. »

Important pour les systèmes interconnectés (réseaux, IoT, web, etc.).

3.8. 8. Classification selon le mode d’expression

Classification selon le mode d’expression

Critère	Type	Description	Exemple
Expression	***Exigences déclaratives***	Énoncées comme des contraintes ("Le système doit...")	« Le système doit envoyer un mail de confirmation. »
	Exigences opérationnelles	Exprimées en scénarios ou cas d’usage	« L’utilisateur sélectionne un taxi, saisit sa destination, puis confirme la course. »

Influence la modélisation et la traçabilité (ex. UML, BPMN, SysML).

3.9. 9. Classification selon la vérifiabilité

Classification selon la vérifiabilité

Critère	Type	Description	Exemple
Vérifiabilité	***Exigences vérifiables***	Peut être testée, mesurable	« Temps de réponse < 2 s. »
	Non vérifiables	Trop vagues ou subjectives	« L’interface doit être agréable. »

Les exigences doivent toujours être formulées pour être vérifiables.

3.10. Autres

Autres

Sous-type	Description	Exemple
Objectifs stratégiques	Vision à long terme	Devenir la plateforme de transport n°1 en Algérie.
Objectifs réglementaires	Contraintes légales ou normes	Respecter la réglementation sur les données personnelles.
Objectifs économiques	Rentabilité, réduction des coûts	Réduire de 20 % les coûts de gestion de flotte.
Objectifs de qualité de service	Satisfaction client	Améliorer la note moyenne à 4,5 / 5.

Autres types d'exigences

3.11. Résumé

Résumé des types des exigences

Critère	Question clé	Exemples de types
Nature / contenu	Que fait le système ?	Fonctionnelles / Non fonctionnelles
Niveau / abstraction	À quel niveau de détail ?	Métier / Utilisateur / Système
Source	D’où vient l’exigence ?	Interne / Externe
Stabilité	Va-t-elle évoluer ?	Stable / Volatile
Priorité	Quelle importance ?	Must / Should / Could / Won’t
Domaine	Quelle qualité ?	Sécurité / Performance / Ergonomie...
Contexte	Avec quoi interagit-elle ?	Interface / Environnement
Mode d’expression	Comment elle est formulée ?	Déclarative / Opérationnelle
Vérifiabilité	Est-elle mesurable ?	Oui / Non

4. Langages/outils de spécification des exigences

Langage naturel (informel)

Langue courante utilisée pour exprimer les besoins sous forme de texte.

Exemple **:**

« Le système doit permettre à l'utilisateur de se connecter à l'aide d'un identifiant et d'un mot de passe. »

4.1. Problèmes

- Ambiguïté lexicale : que signifie « se connecter » exactement ?
- Omission de cas d'erreurs.
- Impossibilité de test automatique.

4.2. Bonnes pratiques

Employer un gabarit standardisé comme :

- Le <type d'acteur> doit pouvoir <verbe d'action> <complément>.
- Éviter les mots vagues : rapide, facile, sécurisé → les remplacer par des critères mesurables.

4.3. Langages structurés (ou pseudo-formels)

Objectif

Réduire l'ambiguïté du langage naturel en imposant une grammaire contrôlée.

Exemple **:**

EARS (Easy Approach to Requirements Syntax)}

Syntaxe simple basée sur des modèles de phrases :

- --> When <condition>, the <system> shall <response>.

Exemple **:**

Quand l'utilisateur clique sur le bouton "login", le système doit vérifier le mot de passe.

Exemple **: Gabarits IEEE 830**

Le système doit [verbe] [complément] [condition ou critère de performance].

Exemple **: Langage contrôlé (CNL – Controlled Natural Language)**

Utilise un sous-ensemble restreint du langage naturel :

- Vocabulaire limité
- Règles syntaxiques simples

Exemple **:**

Attempto Controlled English (ACE).

4.4. Langages semi-formels

Ils combinent texte structuré + diagrammes normalisés.

Objectif : clarifier la structure du système tout en restant lisible

4.5. Langages formels

☐ *Définition* :

Langages basés sur des notations mathématiques permettant :

- La spécification non ambiguë,
- La vérification automatique (preuve, model checking),
- La validation de cohérence.

☐ *Exemple* : Langages formels de spécification des exigences

Langage	Domaine d'application	Principe de base
Z	Spécification de systèmes logiciels	Ensemble, logique des prédicats
VDM (Vienna Development Method)	Développement logiciel sûr	Théorie des types + logique
B / Event-B	Systèmes critiques, ferroviaires, avioniques	Raffinement formel, preuve mathématique
Alloy	Modélisation structurelle	Logique relationnelle et analyse automatique
OCL (Object Constraint Language)	Contraintes UML	Expressions logiques attachées à UML

4.6. Langages hybrides et modernes

Les outils récents intègrent plusieurs niveaux de langage :

Outil	Langage intégré	Fonction
SysML + OCL	Semi-formel + Formel	Vérification automatique des modèles SysML
Alloy Analyzer	Formel graphique	Exploration de modèles
DOORS / ReqView / Polarion	Langage structuré	Gestion et traçabilité des exigences
UML + Natural Language Template	Semi-formel	Lien entre texte et modèle UML

4.7. Synthèse

Synthèse des types de langages de spécification des exigences

Type de langage	Niveau de formalisation	Avantages	Inconvénients	Exemple
Langage naturel	Informel	Facile à comprendre, adapté aux non-techniciens	Ambigu, difficile à vérifier	Français, anglais
Langage semi-formel	Structuré	Plus précis, adapté aux diagrammes	Peut manquer de rigueur logique	UML, SysML, BPMN
Langage formel	Formel (mathématique)	Non ambigu, vérifiable automatiquement	Difficile pour les non-experts	OCL, Z, Alloy, VDM