Certified Professional for Requirements Engineering

Niveau Fondamentaux

Manuel

Martin Glinz
Hans van Loenhoud
Stefan Staal
Stan Bühne

Remerciements

Le contenu de la version 1.0.0 a été revu par Rainer Grau, Karol Frühauf et Camille Salinesi. Tracey Duffy a procédé à une révision en anglais. Stan Bühne et Stefan Sturm ont réalisé le montage final.

La version 1.0.0 a été approuvée pour publication le 11 novembre 2020 par le Conseil de l'IREB sur recommandation de Xavier Franch et Frank Houdek.

Nous remercions tous les participants pour leur contribution.

Préambule

Ce manuel fournit une introduction à l'ingénierie des exigences basée sur le syllabus version 3.0 pour le Certified Professional for Requirements Engineering (CPRE)–Foundation Level selon la norme IREB. Il complète le syllabus et s'adresse à trois groupes de lecteurs :

- Les *étudiants et les professionnels* qui souhaitent se familiariser avec l'ingénierie des exigences et passer l'examen de certification peuvent utiliser ce manuel pour accompagner les cours de formation proposés par les organismes de formation, ainsi que pour l'auto-apprentissage et la préparation individuelle à l'examen de certification. Ce manuel peut également être utilisé pour rafraîchir les connaissances existantes sur l'ingénierie des exigences, par exemple lors de la préparation d'un cours et d'un examen de niveau avancé du CPRE.
- Les *organismes de formation* qui proposent des formations sur le niveau Fondation du CPRE peuvent utiliser ce manuel en complément du syllabus pour développer leur matériel de formation ou comme texte d'étude pour les participants à leurs formations.
- Les *professionnels* de l'industrie qui souhaitent appliquer des concepts et des connaissances éprouvés en matière d'IE dans leur travail pratique trouveront dans ce manuel une mine d'informations utiles.

Ce manuel fournit également un lien entre le syllabus, qui énumère et explique les objectifs d'apprentissage, et la littérature sur l'ingénierie des exigences. Chaque chapitre est accompagné de références bibliographiques et de conseils de lecture. La structure du manuel correspond à celle du programme d'études.

La terminologie utilisée dans ce manuel est basée sur le glossaire de la terminologie de l'ingénierie des exigences du CPRE (CPRE Glossary of Requirements Engineering Terminology) [Glin2025]. Nous recommandons de télécharger ce glossaire à partir du site web de l'IREB et de l'utiliser comme référence terminologique.

Vous trouverez plus d'informations sur le programme de certification CPRE, y compris les syllabi, le glossaire, les règlements d'examen et les exemples de questions d'examen sur le site Internet de l'IREB à l'adresse suivante : https://www.ireb.org.

Les auteurs et l'IREB ont consacré beaucoup de temps et d'efforts à la préparation, à la révision et à la publication de ce manuel. Nous espérons que vous prendrez plaisir à lire ce manuel. Si vous détectez des erreurs ou si vous avez des suggestions d'amélioration, veuillez nous contacter à l'adresse suivante : info@ireb.org.

Nous tenons à remercier toutes les personnes qui ont contribué à la création et à la publication de ce manuel. Karol Frühauf, Rainer Grau et Camille Salinesi ont relu attentivement le manuscrit et formulé de précieuses suggestions d'amélioration. La traduction française a été réalisée par Eric Riou du Cosquer et Olivier Denoo.

Nous remercions également les responsables du Conseil de l'IREB, Xavier Franch et Frank Houdek, pour leurs commentaires et leur soutien. Stefan Sturm a apporté ses encouragements et son soutien logistique. Nous remercions également nos conjoints et nos familles pour leur patience et leur soutien.

Martin Glinz, Hans van Loenhoud, Stefan Staal et Stan Bühne

Novembre 2020

Formulations sensibles au genre

Nous nous sommes délibérément abstenus d'utiliser des formulations sexospécifiques dans ce document.

Il va sans dire qu'à l'IREB, nous soutenons les formulations sensibles au genre. Cependant, nous voyons aussi la nécessité de formuler des matières complexes d'une manière facilement compréhensible.

Les textes exigeant une forme masculine et féminine seraient, de fait, moins lisibles et donc plus difficiles à comprendre, alors que l'objectif de ce document est de présenter et de communiquer un contenu clair et précis.
Parce que nous voulons aider les lecteurs à rester concentrés sur le contenu, nous avons fait le choix délibéré de n'utiliser que la forme masculine dans ce document.

Ce choix ne doit, en aucun cas, être interprété comme une expression de manque de respect.

Comprendre les encadrés de ce manuel

Le manuel comprend quatre encadrés de couleurs différentes qui complètent le texte explicatif.

Il s'agit de :

Définition (correspondant au glossaire [Glin2025])

Note du traducteur : les définitions figurant dans le glossaire du CPRE sont présentées exclusivement en anglais afin de ne pas dénaturer le sens du texte.

Conseil

Exemple

Expression

Historique des versions

Version	Date	Comment	Authors
1.3.0	1er avril 2026	Première publication	Martin Glinz Hans van Loenhoud Stefan Staal Stan Bühne Wim Decoutre
1.3.1	9 april 2026	Les termes du glossaire apparaissent désormais dans les encadrés verts de définition, à la fois dans la langue nationale et en anglais.	

Table des matières

1 Introduction et vue d'ensemble

Dans ce chapitre, vous apprendrez ce qu'est l'ingénierie des exigences (IE) et la valeur qu'elle apporte.

1.1 Ingénierie des exigences : Quoi ?

Depuis le début de l'évolution humaine, les humains ont mis en place des systèmes techniques et organisationnels pour les *aider* à accomplir des tâches ou à atteindre des objectifs. Avec l'essor de l'ingénierie, les humains ont également commencé à construire des systèmes qui *automatisent* les tâches humaines.

Chaque fois que les humains décident de créer un système pour faciliter ou automatiser certaines tâches, ils doivent déterminer ce qu'ils veulent construire. Cela signifie qu'ils doivent s'informer des désirs et des besoins des personnes ou des organisations qui utiliseront le système, en bénéficieront ou en subiront les conséquences. En d'autres termes, ils doivent connaître les exigences de ce système. Les exigences constituent la base de tout développement ou évolution de systèmes ou de parties de systèmes. Les exigences existent toujours, même si elles ne sont pas explicitement formulées et documentées.

Le terme " *exigence* " fait référence à trois concepts : [Glin2025]:

> **Définition 1.1. Exigence *(Requirement)* :**
>
> 1. A need perceived by a *stakeholder*.
> 2. A capability or property that a *system* shall have.
> 3. A documented representation of a need, capability, or property..

Un ensemble d'exigences représentées de manière systématique – typiquement pour un système – qui satisfait à des critères donnés est appelé *spécification des exigences*.

On distingue trois types d'exigences :

- Les *exigences fonctionnelles* concernent un résultat ou un comportement qui doit être fourni par une fonction d'un système. Cela comprend les exigences relatives aux données ou à l'interaction d'un système avec son environnement.
- Les *exigences* de qualité concernent les préoccupations de qualité qui ne sont pas couvertes par les exigences fonctionnelles – par exemple, la performance, la disponibilité, la sécurité ou la fiabilité.
- Les *contraintes* sont des exigences qui limitent l'espace de la solution au-delà de ce qui est nécessaire pour répondre aux exigences fonctionnelles et aux exigences de qualité données.

Il est à noter que le traitement des exigences relatives aux projets ou aux processus de développement n'entre pas dans le champ d'application du présent manuel.

Il n'est pas toujours facile de faire la distinction entre les exigences fonctionnelles, les exigences de qualité et les contraintes. Une méthode éprouvée pour les différencier consiste à demander quelle est la *préoccupation* à laquelle répond une exigence : si la préoccupation concerne les résultats, le comportement ou les interactions requis, il s'agit d'une exigence fonctionnelle. S'il s'agit d'un problème de qualité qui n'est pas couvert par les exigences fonctionnelles, nous avons une exigence de qualité. Si la préoccupation consiste à restreindre l'espace de la solution mais n'est ni une exigence fonctionnelle ni une exigence de qualité, il s'agit d'une contrainte. La règle populaire "*Ce que* le système doit faire → exigence fonctionnelle vs. *comment* le système doit le faire → exigence de qualité" conduit souvent à des erreurs de classification, en particulier lorsque les exigences sont spécifiées de manière très détaillée ou lorsque les exigences de qualité sont très importantes.

Par exemple, l'exigence "Le formulaire d'entrée du client doit contenir des champs pour le nom et le prénom du client, pouvant contenir jusqu'à 32 caractères par champ, affichant au moins 24 caractères, reliés à gauche, avec une police sans empattement de 12 points" est une exigence fonctionnelle même si elle contient beaucoup d'informations sur *la manière de procéder*. Prenons un autre exemple, celui d'un système qui traite les données de mesure produites par le détecteur d'un accélérateur de particules à haute énergie. Ces détecteurs produisent d'énormes quantités de données en temps réel. Si vous demandez à un physicien : "Que doit faire le système ?"l'une des premières réponses sera probablement que le système doit être capable de faire face au volume de données produites. Toutefois, les exigences relatives au volume de données ou à la vitesse de traitement sont des exigences de qualité [Glin2007] et non des exigences fonctionnelles.

Lorsque des personnes adoptent une approche systématique et disciplinée de la spécification et de la gestion des exigences, on parle d'*ingénierie des exigences (IE)*. La définition suivante de l'ingénierie des exigences reflète également la raison d'être de l'ingénierie des exigences.

> Définition 1.2. Ingénierie des exigences (IE) *(Requirements Engineering (RE))* :
> The systematic and disciplined approach to the specification and management of requirements with the goal of *understanding the stakeholders' desires and needs* and *minimizing the risk* of delivering a system that does not meet these desires and needs.

Le concept de *parties prenantes* [GIWi2007] est un principe fondamental de l'ingénierie des exigences (voir le chapitre 2).

> **Définition 1.3. Partie prenante *(Stakeholder)* :**
> A person or organization who influences a system's requirements or who is impacted by that system.

Notez que l'influence peut également être indirecte. Par exemple, certaines parties prenantes peuvent avoir à suivre des instructions fournies par leurs managers ou leurs organisations.

Conformément à la définition du glossaire du CPRE RE [Glin2025], nous utilisons le terme "*système*" au sens large dans ce manuel :

> **Définition 1.4. Système *(System)* :**
> 1. In general: a principle for ordering and structuring.
> 2. In engineering: a coherent, delimitable set of elements that—by coordinated action—achieve some purpose.

Notez qu'un système peut comprendre d'autres systèmes ou *composants* en tant que sous-systèmes. Les objectifs d'un système peuvent être atteints par :

- *Déployer* le système dans le(s) lieu(x) où il est utilisé
- Vendre/fournir le système à ses utilisateurs en tant que *produit*
- Disposer de fournisseurs qui offrent les capacités du système aux utilisateurs sous forme de *services*

Nous utilisons donc le terme "système" comme un terme générique qui englobe les produits, les services, les applications ou les dispositifs.

1.2 Ingénierie des exigences : Pourquoi ?

Le développement de systèmes (construction de nouveaux systèmes et évolution des systèmes existants) est une entreprise coûteuse et constitue un risque élevé pour tous les participants. Dans le même temps, les systèmes qui ont une importance pratique sont trop vastes pour être appréhendés intellectuellement par une seule personne. C'est pourquoi les ingénieurs ont développé divers principes et pratiques pour gérer le risque lors du développement d'un système et pour maîtriser la complexité intellectuelle. L'ingénierie des exigences fournit les principes et les pratiques de la perspective des exigences.

Une ingénierie des exigences (IE) adéquate ajoute de *la valeur* [Glin2016], [Glin2008] au processus de développement d'un système :

- L'IE réduit au minimum le risque d'échec ou de modifications coûteuses lors des étapes de développement ultérieures. La détection et la correction précoces

d'exigences erronées ou manquantes sont beaucoup moins coûteuses que la correction d'erreurs et le remaniement causés par des exigences manquantes ou erronées à des stades de développement ultérieurs ou même après le déploiement d'un système.
- L'IE facilite la complexité intellectuelle liée à la compréhension du problème qu'un système est censé résoudre et à la réflexion sur les solutions potentielles.
- L'IE fournit une base adéquate pour l'estimation de l'effort et du coût de développement.
- L'IE est un prérequis pour tester correctement le système.

Les symptômes typiques d'une ingénierie des exigences inadéquate sont des exigences manquantes, peu claires ou erronées en raison de :

- La précipitation des équipes de développement dans la mise en œuvre d'un système en raison de la pression exercée par le calendrier
- Problèmes de communication entre les parties concernées – en particulier, entre les parties prenantes et les développeurs de systèmes et entre les parties prenantes elles-mêmes
- L'hypothèse selon laquelle les exigences sont évidentes, ce qui est faux dans la plupart des cas
- Personnes menant des activités d'IE sans avoir la formation et les compétences adéquates

1.3 Ingénierie des exigences : Où ?

L'ingénierie des exigences peut être appliquée aux exigences de tout type de système. Cependant, le cas d'application dominant pour l'IE aujourd'hui concerne les systèmes dans lesquels les logiciels jouent un rôle majeur. Ces systèmes sont constitués de composants logiciels, d'éléments physiques (produits techniques, matériel informatique, appareils, capteurs, etc.) et d'éléments organisationnels (personnes, postes, processus d'entreprise, questions juridiques et de conformité, etc.).

Les systèmes qui contiennent à la fois des logiciels et des composants physiques sont appelés *systèmes cyber-physiques*.

Les systèmes qui englobent les logiciels, le matériel, les personnes et les aspects organisationnels sont appelés *systèmes socio-techniques*.

Selon le point de vue adopté, les exigences se présentent sous différentes formes :

Les *exigences du système* décrivent le fonctionnement et le comportement d'un système – tels qu'ils sont observés à l'interface entre le système et son environnement – afin que le système satisfasse les désirs et les besoins de ses parties prenantes. Dans le cas de systèmes purement logiciels, on parle d'exigences logicielles.

Les exigences des parties prenantes expriment les désirs et les besoins des parties prenantes qui doivent être satisfaits par la construction d'un système, du point de vue des parties prenantes.

Les *exigences des utilisateurs* sont un sous-ensemble des exigences des parties prenantes. Ils couvrent les désirs et les besoins des utilisateurs d'un système.

Les *exigences du domaine* spécifient les propriétés requises d'un système socio-technique ou cyber-physique.

Les *exigences professionnelles* se concentrent sur les buts, les objectifs et les besoins d'une organisation qui doivent être atteints par l'utilisation d'un système (ou d'un ensemble de systèmes).

Les formes d'occurrence présentées ci-dessus correspondent à celles définies dans la norme [ISO29148], à l'exception des exigences relatives au domaine. En raison de leur importance, nous traitons les exigences du domaine comme une forme à part entière. Le rôle et l'importance des exigences du domaine sont examinés à la section 2.2, principe 4.

1.4 Ingénierie des exigences : Comment ?

Les principales tâches en matière d'IE sont l'élucidation (Chapitre 4), la documentation (Chapitre 3), la validation (Section 4.4) et la gestion (Chapitre 6) des exigences. Le support d'outils (Chapitre 7) peut aider à accomplir ces tâches. L'analyse des exigences et la résolution des conflits d'exigences sont considérées comme faisant partie de l'élucidation.

Cependant, il n'existe pas de processus universel décrivant quand et comment l'IE doit être réalisée lors du développement d'un système. Pour chaque développement de système nécessitant des activités d'IE, un processus d'IE approprié doit être adapté à partir d'un large éventail de possibilités. Les facteurs qui influencent cette adaptation sont, par exemple, les suivants :

- Le processus global de développement du système – en particulier, linéaire et axé sur la planification par opposition à itératif et agile
- Le contexte de développement – en particulier, la relation entre le fournisseur, le(s) client(s) et les utilisateurs d'un système
- La disponibilité et la capacité et des parties prenantes

Il existe également une dépendance mutuelle entre les produits d'activité à produire (voir section 3.1) et le processus d'IE à choisir. Pour plus de détails, voir le chapitre 5.

1.5 Le rôle et les tâches d'un ingénieur des exigences

Dans la pratique, très peu de personnes portent le titre d'*ingénieur en exigences*. Nous considérons qu'une personne joue le *rôle* d'un ingénieur en exigences lorsqu'elle :

- Elucide, documente, valide et/ou gére les exigences dans le cadre de leurs fonctions
- A une connaissance approfondie de l'IE, qui lui permet de définir les processus d'IE, de sélectionner les pratiques d'IE appropriées et d'appliquer ces pratiques correctement
- Est capable de combler le fossé entre le problème et les solutions potentielles

Le rôle de l'ingénieur des exigences fait partie de plusieurs fonctions définies par les organisations. Par exemple, les analystes métier, les spécialistes des applications, les Product Owners, les ingénieurs systèmes et même les développeurs peuvent jouer le rôle d'un ingénieur des exigences. Les connaissances et les compétences en matière d'IE sont également utiles à de nombreux autres professionnels, tels que les concepteurs, les testeurs, les architectes de systèmes ou les directeurs techniques.

1.6 Que faut-il savoir sur l'ingénierie des exigences ?

L'ensemble des compétences qu'un ingénieur en exigences doit acquérir se compose de plusieurs éléments. Les éléments fondamentaux sont abordés dans les chapitres suivants de ce manuel.

Au-delà des compétences techniques et analytiques, l'ingénieur en exigences doit également posséder ce que l'on appelle des "soft skills" : la capacité d'écouter, de modérer, de négocier et de servir de médiateur, ainsi que l'empathie pour les parties prenantes et l'ouverture aux besoins et aux idées d'autrui.

L'IE est régie par un ensemble de principes fondamentaux qui s'appliquent à toutes les tâches, activités et pratiques de l'IE. Ces principes sont présentés au chapitre 2.

Les exigences peuvent être documentées sous différentes formes. Divers produits d'activités peuvent être créés à différents niveaux de maturité et de détail, depuis des produits plutôt informels et temporaires jusqu'à des produits d'activités très détaillés et structurés qui adhèrent à des règles de représentation strictes. Il est important de sélectionner des produits d'activités et des formes de documentation adaptés à la situation en question et de créer correctement les produits d'activités choisis. Les produits d'activités et les pratiques de documentation sont présentés au chapitre 3.

Les exigences peuvent être élaborées (c'est-à-dire élucidées et validées) à l'aide de diverses pratiques. L'ingénieur des exigences doit être capable de sélectionner les pratiques les mieux adaptées à une situation donnée et de les appliquer correctement. Les pratiques d'élaboration sont présentées au chapitre 4.

La compréhension des processus et des structures de travail possibles permet aux ingénieurs des exigences de définir une méthode de travail qui correspond aux besoins spécifiques de la situation de développement du système en question. Les processus et les structures de travail sont présentés au chapitre 5.

Les exigences existantes peuvent être gérées à l'aide de diverses pratiques. Les ingénieurs des exigences devraient être en mesure de comprendre quelles pratiques de gestion des exigences les soutiennent pour quelles tâches. Les pratiques de gestion sont présentées au chapitre 6.

Les outils rendent l'IE plus efficace. Les ingénieurs des exigences doivent savoir comment les outils d'IE peuvent les aider et comment choisir un outil adapté à leur situation. Le support aux outils est brièvement abordée au chapitre 7.

1.7 Pour en savoir plus

La terminologie de l'IE utilisée dans ce manuel est définie dans le glossaire CPRE de la terminologie de l'ingénierie des exigences [Glin2025]. Glinz et Wieringa [GlWi2007] expliquent la notion de parties prenantes. Lawrence, Wiegers et Ebert [LaWE2001] examinent brièvement les risques et les pièges de l'IE.

Gause et Weinberg [GaWe1989] ont écrit l'un des premiers manuels sur l'IE qui vaut toujours la peine d'être consulté. Pohl [Pohl2010], Robertson et Robertson [RoRo2012] et Wiegers et Beatty [WiBe2013] sont des manuels populaires sur l'IE. Les notes de cours de Glinz [Glin2019] fournissent une introduction à l'IE sous forme de diapositives. Le manuel de van Lamsweerde [vLam2009] présente une approche de l'IE axée sur les objectifs. Jackson [Jack1995] présente une collection d'essais perspicaces sur les exigences logicielles.

Il existe également des manuels dans d'autres langues que l'anglais. Par exemple, Badreau et Boulanger [BaBo2014] ont rédigé un manuel d'IE en français. Les livres d'Ebert [Eber2022], de Pohl et Rupp [PoRu2021], et de Rupp [Rupp2020] sont des manuels d'ER populaires écrits en allemand. [CLSZ2021] est un manuel en néerlandais.

[PoRu2021] et [CLSZ2021] sont alignés sur la version 3.2 du CPRE Foundation Level Syllabus. Notez que [PoRu2015], qui était le manuel officiel pour l'IREB CPRE Foundation Level version 2.2, n'est plus aligné avec la version actuelle du CPRE Foundation Level Syllabus.

2 Principes fondamentaux de l'ingénierie des exigences

Dans ce chapitre, vous découvrirez neuf principes de base de l'ingénierie des exigences (IE).

2.1 Aperçu des principes

L'IE est régie par un ensemble de principes fondamentaux qui s'appliquent à toutes les tâches, activités et pratiques de l'IE. Une *tâche* est un ensemble cohérent de travaux à effectuer (par exemple, la définition des besoins). Une *activité* est une action ou un ensemble d'actions qu'une personne ou un groupe réalise pour accomplir une tâche (par exemple, l'identification des parties prenantes lors de l'élaboration des exigences). Une *pratique* est une manière éprouvée de réaliser certains types de tâches ou d'activités (par exemple, l'utilisation d'entretiens pour obtenir des exigences de la part des parties prenantes).

Les principes énumérés dans Table 2.1 constituent la base des pratiques présentées dans les chapitres suivants de ce manuel.

Table 2.1 Les neuf principes fondamentaux de l'ingénierie des exigences

1.	*Orientation vers la valeur* : Les exigences sont un moyen d'atteindre une fin, et non une fin en soi
2.	*Les parties prenantes* : L'IE vise à satisfaire les désirs et les besoins des parties prenantes
3.	*Compréhension partagée* : Le développement réussi de systèmes est impossible sans une base commune
4.	*Contexte* : Les systèmes ne peuvent être compris isolément
5.	*Problème – Exigence – Solution* : Un triple entrelacement inévitable
6.	*Validation* : Les exigences non validées sont inutiles
7.	*Evolution* : L'évolution des exigences n'est pas un accident, mais le cas normal
8.	*Innovation* : Il ne suffit pas d'en faire plus
9.	*Travail systématique et discipliné* : On ne peut pas s'en passer en IE

2.2 Les principes expliqués

2.2.1 Principe 1 - Orientation vers la valeur : Les exigences sont un moyen d'atteindre une fin, et non une fin en soi

La rédaction des exigences n'est pas un objectif en soi. Les exigences sont utiles – et les efforts investis dans l'ingénierie des exigences sont justifiés – uniquement si elles apportent une *valeur* ajoutée [Glin2016], [Glin2008], cf. section 1.2. Nous définissons la valeur d'une exigence comme étant son *bénéfice* moins son *coût*. L'avantage d'une exigence est la mesure dans laquelle elle contribue à la construction de systèmes performants (c'est-à-dire des systèmes qui satisfont les désirs et les besoins de leurs parties prenantes) et à la réduction du risque d'échec et de remaniement coûteux lors du développement du système. Le coût d'une exigence correspond au coût nécessaire pour l'élucider, la valider, la documenter et la gérer.

La réduction du risque de remaniement au cours du développement est un élément constitutif de l'avantage d'une exigence bien rédigée. Détecter et corriger une exigence manquée ou erronée au cours de la mise en œuvre ou lorsque le système est déjà opérationnel peut facilement coûter une ou deux fois plus que de spécifier correctement cette exigence dès le départ. Par conséquent, une grande partie de l'avantage des exigences provient des coûts économisés lors de la mise en œuvre et de l'exploitation d'un système.

En d'autres termes, les avantages de l'IE sont souvent à long terme, alors que les coûts sont immédiats. Il faut en tenir compte lors de la mise en place d'un nouveau projet. Réduire les coûts à court terme en dépensant moins pour l'IE a un prix : cela augmente considérablement le risque de retouches coûteuses à des stades ultérieurs du projet.

La *valeur de l'ingénierie des exigences* peut être considérée comme la valeur cumulative des exigences spécifiées. Comme les clients paient généralement pour la mise en œuvre des systèmes, mais pas pour les exigences nécessaires à cette mise en œuvre, la valeur économique de l'IE est essentiellement indirecte. Cet effet est renforcé par le fait que l'avantage des exigences découlant de la réduction des coûts de mise à jour est indirect : il permet d'économiser des coûts lors de la mise en œuvre et de l'exploitation.

> Les effets économiques de l'ingénierie des exigences sont essentiellement indirects ; l'ingénierie des exigences n'est donc qu'un coût.

Pour optimiser la valeur d'une exigence, les ingénieurs des exigences doivent trouver un juste équilibre entre le bénéfice et le coût d'une exigence. Par exemple, le fait d'obtenir et de documenter le besoin d'une partie prenante en tant qu'exigence facilite la communication de ce besoin entre toutes les parties concernées. Cela augmente la probabilité que le système à construire finisse par satisfaire ce besoin, ce qui constitue un avantage. Moins l'exigence est

ambiguë et plus elle est précise, plus elle est bénéfique, car cela réduit le risque d'une mise à jour coûteuse dû à une mauvaise interprétation des exigences par les architectes du système et les équipes de développement. D'autre part, l'augmentation du degré de non ambiguïté et de précision d'une exigence augmente également le coût de l'obtention et de la documentation de l'exigence.

En fait, la quantité d'IE nécessaire pour obtenir des exigences ayant une valeur optimale dépend de nombreux facteurs liés à la situation spécifique dans laquelle les exigences sont créées et utilisées. De toute évidence, le risque de construire un système qui ne satisferait pas les désirs et les besoins des parties prenantes, ce qui pourrait entraîner un échec ou des mises à jour coûteuses, est la *force motrice* qui détermine l'ampleur de l'IE nécessaire. Avant tout, la criticité de chaque exigence doit être évaluée en fonction de l'importance de la ou des parties prenantes qui énoncent l'exigence (voir principe 2) et de l'impact d'un manquement à l'exigence (Figure 2.1).

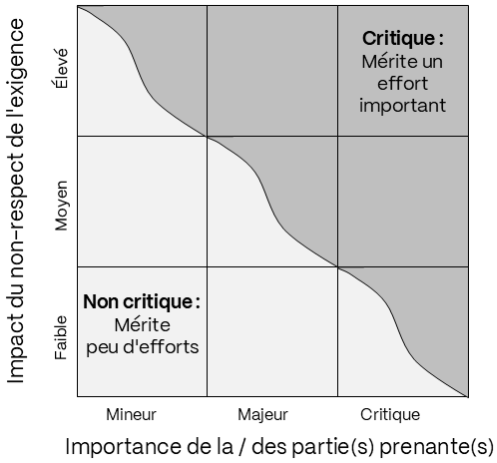


Figure 2.1 Évaluation de la criticité d'une exigence [Glin2008]

En outre, les facteurs d'influence suivants doivent être pris en considération :

- Effort nécessaire pour spécifier l'exigence
- Caractère distinctif de l'exigence (dans quelle mesure elle contribue à la réussite du système global)
- Degré de compréhension commune entre les parties prenantes et les développeurs et entre les parties prenantes
- Existence de systèmes de référence (pouvant servir de spécification par l'exemple)
- Longueur du cycle de retour d'information (temps écoulé entre le moment où une exigence est erronée et celui où l'erreur est détectée)
- Type de relation client-fournisseur
- Conformité réglementaire requise

Nous résumons cette question en deux règles empiriques :

- Le montant optimal d'IE à investir dépend de la situation spécifique et est déterminé par de nombreux facteurs d'influence.
- L'effort investi dans l'IE doit être inversement proportionnel au risque que vous êtes prêt à prendre.

2.2.2 Principe 2 - Les parties prenantes : L'IE vise à satisfaire les désirs et les besoins des parties prenantes

L'objectif final de la construction d'un système est que le système, lorsqu'il est utilisé, résolve les problèmes que ses utilisateurs doivent résoudre et satisfasse les attentes d'autres personnes – par exemple, celles qui ont commandé et payé le système, ou celles qui sont responsables de la sécurité dans l'organisation qui utilise le système. Par conséquent, nous devons déterminer les besoins et les attentes des personnes qui ont un intérêt dans le système, les *parties prenantes* du système [GlWi2007]. Les objectifs fondamentaux de l'IE sont de *comprendre les désirs et les besoins des parties prenantes* et de *minimiser le risque* de fournir un système qui ne réponde pas à ces désirs et à ces besoins ; voir la définition 1.2 à la section 1.2.

Chaque partie prenante a un *rôle* à jouer dans le contexte du système à construire – par exemple, l'utilisateur, le client (acheteur), le client (utilisateur), l'opérateur ou le régulateur. Selon le processus d'IE utilisé, les développeurs d'un système peuvent également être des parties prenantes. C'est souvent le cas dans le cadre d'un développement agile et orienté vers le marché. Une partie prenante peut également avoir plusieurs rôles en même temps. Pour chaque rôle de partie prenante, des personnes appropriées agissant dans ce rôle doivent être sélectionnées en tant que représentants.

Pour les rôles de parties prenantes comportant trop d'individus ou lorsque les individus sont inconnus, des *personas* (personnages fictifs représentant un groupe d'utilisateurs présentant des caractéristiques similaires) peuvent être définis en remplacement. Pour les systèmes déjà en service, les utilisateurs qui fournissent un retour d'information sur le système ou demandent de nouvelles fonctionnalités doivent également être considérés comme des parties prenantes.

Il est logique de classer les parties prenantes en trois catégories en fonction du degré d'influence qu'elles exercent sur la réussite du système :

- *Critique :* ne pas prendre en compte ces parties prenantes entraînera de graves problèmes et fera probablement échouer le système ou le rendra inutile.
- *Important :* le fait de ne pas prendre en compte ces parties prenantes aura un impact négatif sur la réussite du système, mais ne le fera pas échouer.
- *Mineur :* le fait de ne pas prendre en compte ces parties prenantes n'aura aucune influence ou une influence mineure sur la réussite du système.

Cette classification est utile pour évaluer la criticité d'une exigence (voir Figure 2.1) et pour négocier les conflits entre les parties prenantes (voir ci-dessous).

Il ne suffit pas de prendre en compte uniquement les exigences des utilisateurs finaux et des clients (utilisateurs). Cela signifierait que nous risquerions de passer à côté d'exigences critiques d'autres parties prenantes, ce qui peut facilement conduire à l'échec de projets de développement ou au dépassement de leur budget et de leurs délais.

> Il est essentiel d'impliquer les bonnes personnes dans les rôles des parties prenantes concernées pour assurer le succès de l'IE.

Les pratiques d'identification, de priorisation et de travail avec les parties prenantes sont discutées au chapitre 4.

Les parties prenantes ayant des rôles différents ont naturellement des points de vue différents [NuKF2003] sur le système à développer. Par exemple, les utilisateurs veulent généralement un système qui les aide à accomplir leurs tâches de manière optimale, les responsables qui commandent le système veulent l'obtenir à un coût raisonnable, et le responsable de la sécurité de l'organisation se préoccupe avant tout de la sécurité du système. Même les parties prenantes ayant le même rôle peuvent avoir des besoins différents. Par exemple, dans le groupe des utilisateurs finaux, les utilisateurs occasionnels ont des exigences en matière d'interface utilisateur qui peuvent être très différentes de celles des utilisateurs professionnels.

Par conséquent, il ne suffit pas de recueillir les exigences des parties prenantes. Il est essentiel d'identifier les incohérences et les conflits entre les exigences des différentes parties prenantes et de les résoudre, que ce soit en trouvant un consensus, en passant outre ou en spécifiant des variantes du système pour les parties prenantes qui ont en fait des besoins différents ; voir la section 4.3.

2.2.3 Principe 3 - Compréhension partagée : Le développement réussi de systèmes est impossible sans une base commune

Le développement de systèmes, y compris l'IE, est une entreprise à laquelle participent plusieurs personnes. Pour qu'une telle entreprise soit couronnée de succès, les personnes impliquées doivent avoir une *compréhension commune* du problème et des exigences qui en découlent [GlFr2015].

L'IE crée, encourage et garantit une compréhension commune entre les parties concernées : parties prenantes, ingénieurs des exigences et développeurs. Nous distinguons deux formes de compréhension commune :

- Une compréhension *commune explicite* est obtenue grâce à des exigences soigneusement élucidées, documentées et acceptées. C'est l'objectif premier de l'IE dans les processus planifiés.
- Une *compréhension commune implicite* est basée sur une connaissance partagée des besoins, des visions, du contexte, etc. Lorsque les exigences ne sont pas entièrement spécifiées par écrit, il est essentiel de s'appuyer sur une compréhension commune implicite. C'est particulièrement le cas dans toute forme d'ER agile.

La compréhension commune implicite et explicite peut être *fausse*, c'est-à-dire que les gens pensent avoir une compréhension commune d'une question alors qu'ils l'interprètent en fait de différentes manières. Par conséquent, nous ne pouvons jamais nous fier aveuglément à une compréhension commune. Au contraire, la tâche de l'IE est de créer et d'encourager une compréhension commune et de la garantir, c'est-à-dire d'évaluer s'il existe une véritable compréhension commune. Pour limiter l'effort de recherche, il est essentiel de se concentrer sur la compréhension commune des éléments *pertinents*, c'est-à-dire les aspects qui se situent dans les limites du contexte d'un système (cf. principe 4).

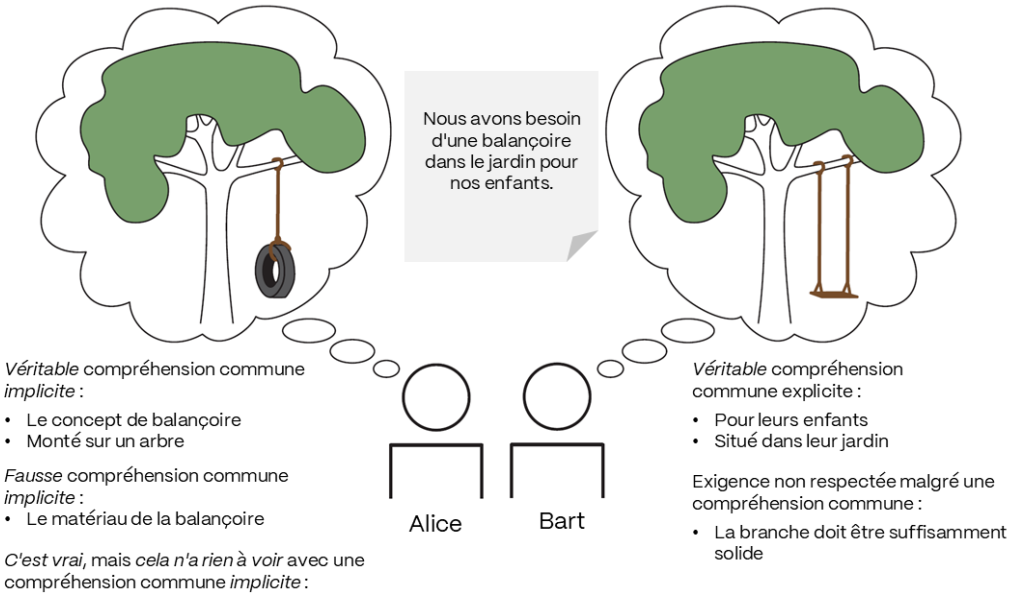


Figure 2.2 Différentes situations de compréhension partagée - illustrées par l'exemple d'un couple qui souhaite installer une balançoire pour ses enfants

Même avec une parfaite compréhension commune, des exigences importantes peuvent encore être ignorées parce que personne ne les a prises en compte. Figure 2.2 illustre différentes situations de compréhension partagée avec l'exemple simple d'un couple qui souhaite installer une balançoire dans son jardin pour ses enfants [Glin2019]. Le post-it au milieu symbolise une spécification écrite.

Les pratiques éprouvées pour *parvenir* à une compréhension commune comprennent l'utilisation de glossaires (section 0), la création de prototypes (section 3.7) ou l'utilisation d'un système existant comme point de référence.

Le principal moyen d'*évaluer une* véritable compréhension explicite et partagée dans l'IE consiste à valider minutieusement toutes les exigences spécifiées (cf. principe 6 et section 4.4). Les pratiques d'évaluation de la compréhension implicite commune comprennent la fourniture d'exemples de résultats attendus, la création de prototypes ou l'estimation du coût de mise en œuvre d'une exigence. La pratique la plus importante pour réduire l'impact des malentendus consiste à utiliser un processus avec de courtes boucles de rétroaction (Chapitre 5).

Certains facteurs favorisent ou entravent la compréhension commune. Par exemple, les facilitateurs sont :

- Connaissance du domaine
- Normes spécifiques à un domaine
- Précédente collaboration réussie
- Existence de systèmes de référence connus de toutes les personnes impliquées
- Culture et valeurs partagées
- Confiance mutuelle éclairée (et non aveugle !)

Les obstacles sont les suivants :

- Distance géographique
- Une relation fournisseur–client (utilisateur) guidée par la méfiance mutuelle
- Externalisation
- Contraintes réglementaires
- Des équipes nombreuses et diversifiées
- Forte rotation des personnes impliquées

Plus la probabilité et l'impact d'une compréhension commune erronée sont faibles et plus le rapport entre les éléments facilitateurs et les obstacles est bon, plus l'IE peut s'appuyer sur une compréhension commune implicite. Inversement, moins il y a de facilitateurs et plus il y a d'obstacles à la compréhension commune, et plus le risque et l'impact d'une fausse compréhension commune d'une exigence sont élevés, plus cette exigence doit être spécifiée et validée explicitement.

2.2.4 Principe 4 - Contexte : Les systèmes ne peuvent être compris isolément

Les exigences ne sont jamais isolées. Elles font référence à des *systèmes* qui sont intégrés dans un contexte. Si le terme de *contexte* désigne en général le réseau de pensées et de significations nécessaire à la compréhension des phénomènes ou des énoncés, il revêt une signification particulière dans le cadre de l'IE.

> Définition 2.1. Contexte (dans l'IE) *(Context (in RE))* :
> The part of a system's environment being relevant for understanding the system and its requirements.

Pour déterminer le contexte d'un système, nous limitons nos considérations au domaine d'application dans lequel le système est intégré. Deux frontières délimitent le contexte d'un système : la frontière du système et la frontière du contexte [Pohl2010] (voir Figure 2.3).

> Définition 2.2. Limite du contexte *(Context boundary)* :
> The boundary between the context of a system and those parts of the application domain that are irrelevant for the system and its requirements.

La *frontière contextuelle* sépare la partie pertinente de l'environnement d'un système à développer de la partie non pertinente, c'est-à-dire la partie qui n'a pas d'influence sur le système à développer et qui ne doit donc pas être prise en compte lors de l'ingénierie des exigences.

> Definition 2.3. Limite du système *(System boundary)* :
> The boundary between a system and its surrounding context.

Le *périmètre du système* délimite le système tel qu'il sera après son implémentation et son déploiement. La limite du système n'est souvent pas claire au départ et peut changer avec le temps. Clarifier les limites du système et définir les interfaces externes entre le système et les éléments de contexte avec lesquels il interagit sont de véritables tâches d'IE.

Les limites du système coïncident souvent avec le *champ d'application* du développement d'un système.

> Définition 2.4. Périmètre *(Scope)* :
> The range of things that can be shaped and designed when developing a system.

Il arrive cependant que les limites du système et son champ d'application ne correspondent pas (voir Figure 2.3). Il peut y avoir des composants à l'intérieur du périmètre du système qui

doivent être réutilisés tels quels (c'est-à-dire qu'ils ne peuvent pas être modelés ou conçus), ce qui signifie qu'ils sont hors du champ d'application. D'autre part, il peut y avoir des éléments dans le contexte du système qui peuvent être redéfinis lorsque le système est développé, ce qui signifie qu'ils sont dans le champ d'application.

Comme les interfaces externes se situent à la frontière du système, l'IE doit déterminer lesquelles de ces interfaces sont dans le champ d'application (c'est-à-dire qu'elles peuvent être façonnées et conçues au cours du processus de développement) et lesquelles sont données et hors du champ d'application.

Il ne suffit pas de prendre en compte uniquement les exigences à l'intérieur de la limite du système.

Premièrement, lorsque le champ d'application comprend des parties du contexte du système, comme le montre Figure 2.3, les changements de contexte dans le champ d'application peuvent avoir une incidence sur les exigences du système. Par exemple, lorsqu'un processus métier doit être partiellement automatisé par un système, il peut être utile d'adapter le processus afin de simplifier son automatisation. Il est évident qu'une telle adaptation a une incidence sur les exigences du système.

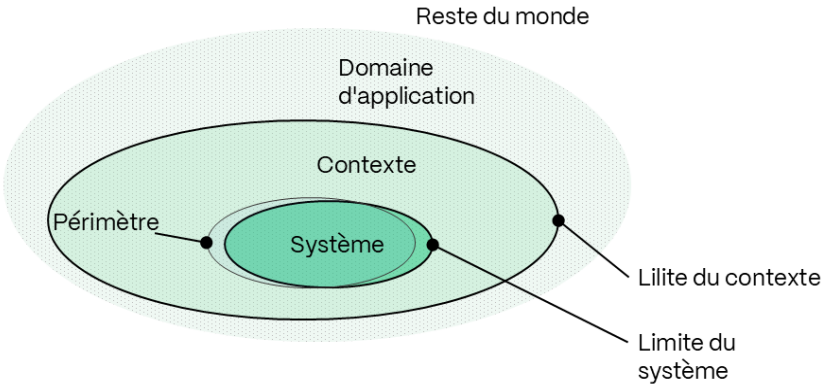


Figure 2.3 Système, contexte, et champ d'application

Deuxièmement, il peut y avoir des phénomènes réels dans le contexte du système qu'un système doit surveiller ou contrôler. Les exigences relatives à ces phénomènes doivent être énoncées en tant qu'exigences de domaine et doivent être correctement mises en correspondance avec les exigences du système. Prenons par exemple le système logiciel qui contrôle la boîte de vitesses automatique d'une voiture. Il est évident que le contexte de ce système comprend la voiture qui est contrôlée par le système. Dans une voiture équipée d'une boîte de vitesses automatique, la position de stationnement ne peut être enclenchée que lorsque la voiture est immobile. Si l'on considère le système de contrôle de la boîte de vitesses, cette exigence se situe dans le contexte du système, c'est-à-dire qu'il s'agit d'une *exigence de domaine*. Pour satisfaire à cette exigence, le système de contrôle doit savoir si la voiture se déplace ou non. Cependant, le système de contrôle ne peut pas détecter directement ce phénomène. Par conséquent, le phénomène réel "la voiture ne bouge pas"

doit être mis en correspondance avec un phénomène que le système de contrôle peut détecter – par exemple, l'entrée d'un capteur qui crée des impulsions lorsqu'une roue de la voiture tourne. L'exigence du domaine concernant l'engagement de la position de stationnement est alors mise en correspondance avec une exigence du système telle que "Le système de commande de la boîte de vitesses ne doit permettre l'engagement de la position de stationnement que si aucune impulsion n'est reçue des capteurs de rotation des roues".

Troisièmement, il peut y avoir des exigences qui ne peuvent être satisfaites par aucune mise en œuvre du système à moins que certaines *exigences* et *hypothèses* du *domaine* dans le contexte du système ne soient respectées. Les hypothèses de domaine sont des hypothèses sur les phénomènes du monde réel dans le contexte d'un système. Prenons l'exemple d'un système de contrôle du trafic aérien (ATS). L'exigence "R1 : L'ATS doit maintenir des positions précises pour tous les aéronefs contrôlés par le système" est une exigence importante du système. Toutefois, cette exigence ne peut être satisfaite que si le radar, dans le contexte de l'ATS, répond aux exigences d'identification correcte de tous les aéronefs dans l'espace aérien contrôlé par le radar et de détermination correcte de leur position. Ces exigences ne peuvent être satisfaites que si tous les aéronefs repérés par le radar répondent correctement aux signaux d'interrogation envoyés par le radar.

En outre, l'exigence R1 ne peut être satisfaite que si certaines hypothèses du domaine dans le contexte de l'ATS se vérifient – par exemple, que le radar n'est pas brouillé par un attaquant malveillant et qu'aucun aéronef ne vole à une altitude inférieure à celle que le radar peut détecter.

L'IE va au-delà de la prise en compte des exigences à l'intérieur des limites du système et de la définition des interfaces externes à l'intérieur des limites du système. L'IE doit également traiter les phénomènes dans le contexte du système.

Par conséquent, l'IE doit également prendre en compte les questions liées au contexte du système :

- Si des changements dans le contexte peuvent se produire, quel est leur impact sur les exigences du système ?
- Quelles sont les exigences du contexte réel qui sont pertinentes pour le système à développer ?
- Comment ces exigences du monde réel peuvent-elles être mises en correspondance de manière adéquate avec les exigences du système ?
- Quelles hypothèses sur le contexte doivent être maintenues pour que le système fonctionne correctement et que les exigences du monde réel soient satisfaites ?

2.2.5 Principe 5 - Problème, exigence, solution : Un triple entrelacement inévitable

Les problèmes, leurs solutions et les exigences sont étroitement et inévitablement liés [SwBa1982]. Toute situation dans laquelle des personnes ne sont pas satisfaites de la manière dont elles font les choses peut être considérée comme l'apparition d'un *problème*. Afin d'éliminer ce problème, un système socio-technique peut être développé et déployé. Les *exigences* relatives à ce système doivent être définies afin de faire du système une *solution* efficace au problème. Spécifier des exigences n'a pas de sens s'il n'y a pas de problème à résoudre ou si aucune solution ne sera développée. Il n'est pas non plus judicieux de développer une solution à la recherche d'un problème à résoudre ou d'exigences à satisfaire.

Il est important de noter que les problèmes, les exigences et les solutions ne se présentent pas nécessairement dans cet ordre. Par exemple, lors de la conception d'un système innovant, les idées de solutions créent des besoins utilisateurs qui doivent être élaborés en exigences et implémentés dans une solution réelle.

Les problèmes, les exigences et les solutions peuvent être liés de nombreuses façons :

- L'entrelacement hiérarchique : lors du développement de grands systèmes avec une hiérarchie à plusieurs niveaux de sous-systèmes et de composants, les exigences de haut niveau conduisent à des décisions de conception de haut niveau, qui à leur tour informent les exigences de niveau inférieur qui conduisent à des décisions de conception de niveau inférieur, etc.
- Faisabilité technique : spécifier des exigences non réalisables est un gaspillage d'efforts ; cependant, il se peut que la faisabilité d'une exigence ne puisse être évaluée que lors de l'exploration de solutions techniques.
- Validation : les prototypes, qui sont un moyen puissant de valider les exigences, constituent des solutions partielles du problème.
- Biais de solution : les différentes parties prenantes peuvent envisager des solutions différentes pour un problème donné, ce qui a pour conséquence qu'elles spécifient des exigences différentes et contradictoires pour ce problème.

L'entrelacement des problèmes, des exigences et des solutions a également des conséquences sur le processus de développement d'un système :

- Il est rarement possible de séparer strictement l'IE des activités de conception et de mise en œuvre du système. C'est pourquoi les processus de développement en cascade ne fonctionnent pas bien.
- Néanmoins, les ingénieurs des exigences visent à séparer autant que possible les problèmes, les exigences et les solutions les uns des autres lors de la réflexion, de la communication et de la documentation. Cette *séparation des préoccupations* rend les tâches de l'IE plus faciles à gérer.

Malgré l'entrelacement inévitable des problèmes, des exigences et des solutions, les ingénieurs des exigences s'efforcent de séparer les préoccupations liées aux exigences de celles liées aux solutions lorsqu'ils réfléchissent, communiquent et documentent.

2.2.6 Principe 6 - Validation : Les exigences non validées sont inutiles

Lorsqu'un système est développé, le système final déployé doit satisfaire les désirs et les besoins des parties prenantes. Cependant, il est très risqué d'effectuer ce contrôle à la toute fin du développement. Afin de contrôler le risque d'insatisfaction des parties prenantes dès le début, la validation des exigences doit commencer dès l'IE (voir Figure 2.4).

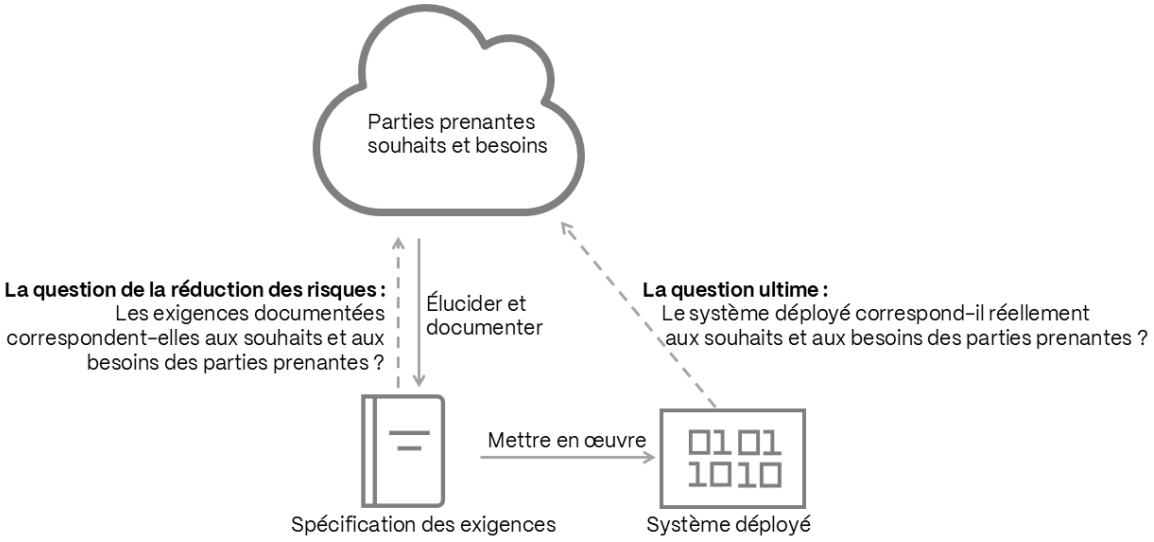


Figure 2.4 Validation [Glin2019]

Définition 2.5. Validation :

The process of confirming that an item (a system, a work product, or a part thereof) matches its stakeholders' needs.

Dans l'IE, la *validation* est le processus qui consiste à confirmer que les exigences documentées correspondent aux besoins des parties prenantes ; en d'autres termes, il s'agit de confirmer que les bonnes exigences ont été spécifiées.

La validation est une activité centrale de l'IE : il n'y a pas de spécification sans validation.

Lors de la validation des exigences, nous devons vérifier si :

- Les parties prenantes sont parvenues à un accord sur les exigences (les conflits ont été résolus, les priorités ont été fixées)
- Les souhaits et les besoins des parties prenantes sont couverts de manière adéquate par les exigences
- Les hypothèses relatives au domaine (voir le principe 4 ci-dessus) sont raisonnables, c'est-à-dire que l'on peut s'attendre à ce qu'elles soient respectées lorsque le système sera déployé et opérationnel

Les pratiques de validation des exigences sont examinées dans la section 4.4.

2.2.7 Principe 7 - Évolution : Modifier des exigences n'est pas un accident, mais le cas normal

Tout système technique est susceptible d'évoluer. Les besoins, les entreprises et les capacités évoluent en permanence. En conséquence, les exigences relatives aux systèmes censés répondre aux besoins, soutenir les entreprises et utiliser les capacités techniques évolueront également. Dans le cas contraire, ces systèmes et leurs exigences perdent progressivement de leur valeur et finissent par devenir inutiles.

Une exigence peut changer pendant que les ingénieurs des exigences sont encore en train d'élucider d'autres exigences, lorsque le système est en cours de mise en œuvre, ou lorsqu'il est déployé et utilisé.

Il existe de nombreuses raisons qui conduisent à demander le changement d'une exigence ou d'un ensemble d'exigences pour un système, par exemple :

- Des modifications de processus métier
- Des concurrents qui lancent de nouveaux produits ou services
- Des clients qui changent leurs priorités ou leurs opinions
- Des changements de technologie
- Des retours d'informations d'utilisateurs du système demandant de nouvelles fonctionnalités ou des modifications
- Des erreurs détectées dans les exigences ou des hypothèses erronées dans le domaine détectées

Les exigences peuvent également changer en raison du retour d'information des parties prenantes lors de la validation des exigences, de la détection de défauts dans les exigences précédemment définies ou de l'évolution des besoins.

En conséquence, les ingénieurs des exigences doivent poursuivre deux objectifs apparemment contradictoires :

- Permettre aux exigences de changer, car il serait vain d'essayer d'ignorer l'évolution des exigences.
- Maintenir les exigences stables, car sans une certaine stabilité dans les exigences, le coût du changement peut devenir prohibitif. En outre, les équipes de développement

ne peuvent pas développer de manière systématique si les exigences changent quotidiennement.

Les ingénieurs en exigences doivent gérer l'évolution des exigences. Sinon, c'est l'évolution qui les gérera.

Les processus de modification des exigences qui répondent à ces deux objectifs sont examinés à la section 6.7.

2.2.8 Principe 8 - Innovation : Il ne suffit pas d'en faire plus

Alors que l'IE vise à satisfaire les désirs et les besoins des parties prenantes, les ingénieurs des exigences qui se contentent de jouer le rôle d'enregistreurs de la voix des parties prenantes, en spécifiant exactement ce que les parties prenantes leur disent, ne font pas le bon travail. Donner aux parties prenantes exactement ce qu'elles veulent, c'est se priver de la possibilité de faire mieux qu'avant.

Imaginez par exemple le scénario suivant. Une compagnie d'assurance souhaite renouveler le système de reporting pour ses agents. Le rapport le plus fréquemment utilisé est un tableau de 18 colonnes, qui est environ deux fois plus large que l'écran lorsqu'il est affiché sur les ordinateurs portables des agents. La visualisation de ce rapport nécessite donc beaucoup de défilement. Les parties prenantes veulent donc pouvoir zoomer dans le rapport, en utilisant les boutons plus et moins sur l'écran. Dans cette situation, un bon ingénieur en exigences ne se contentera pas d'enregistrer cette exigence. Au contraire, ils commenceront à poser des questions. Il s'avère que l'entreprise va remplacer les ordinateurs portables des agents par des tablettes. Par conséquent, l'utilisation de gestes à deux doigts au lieu des boutons requis facilitera grandement le zoom. En outre, il s'avère que trois colonnes du rapport peuvent être supprimées moyennant une légère modification des règles de déclaration, ce que l'entreprise accepte de faire. En outre, seules six colonnes du rapport sont toujours nécessaires ; les autres colonnes ne sont utilisées que dans des cas particuliers.

Compte tenu de ces éléments, les ingénieurs des exigences suggèrent que les parties prenantes exigent que (1) le rapport affiche les mêmes informations que dans le système actuel, à l'exception du contenu des trois colonnes supprimées ; (2) lorsque le rapport est ouvert, seules les six colonnes importantes sont affichées en pleine largeur, tandis que les autres colonnes sont réduites à une largeur minimale ; et (3) que les agents puissent développer une colonne réduite en tapant sur son en-tête (et la réduire à nouveau en tapant une nouvelle fois sur son en-tête).

De cette manière, les agents disposeront d'un système qui ne se contente pas d'ajouter une solution de contournement pour afficher un rapport trop volumineux. Au lieu de cela, le système résoudra le problème des agents grâce à une fonction innovante de filtrage de l'information et à un moyen intuitif de zoomer.

C'est ainsi que naît l'innovation. Les bons ingénieurs des exigences sont sensibles à l'innovation : ils s'efforcent non seulement de satisfaire les parties prenantes, mais aussi de les rendre heureuses, enthousiastes ou de leur donner un sentiment de sécurité [KSTT1984]. En même temps, ils évitent le piège de croire qu'ils savent tout mieux que les parties prenantes.

> Les bons ingénieurs en exigences vont au-delà de ce que leur disent les parties prenantes.

À petite échelle, l'IE façonne des systèmes innovants en s'efforçant d'offrir de nouvelles fonctionnalités et une facilité d'utilisation. En outre, les ingénieurs des exigences doivent également avoir une vue d'ensemble, en explorant avec les parties prenantes s'il existe des façons perturbatrices de faire les choses, ce qui conduit à une innovation à grande échelle [MaGR2004].

La section 4.2 examine plusieurs techniques pour favoriser l'innovation dans le domaine de l'IE.

2.2.9 Principe 9 - Travail systématique et discipliné : On ne peut pas s'en passer en IE

L'IE n'est pas un art mais une discipline, ce qui implique que l'IE soit pratiquée de manière systématique et disciplinée. Quel que soit le ou les processus utilisés pour développer un système, Il est nécessaire de s'appuyer sur des processus et des pratiques d'IE appropriés pour élucider, documenter, valider et gérer systématiquement les exigences. Même lorsqu'un système est développé de manière ad hoc, une approche systématique et disciplinée de l'IE (par exemple, en favorisant systématiquement une compréhension commune, voir le principe 3) améliorera la qualité du système qui en résultera.

> L'agilité et la flexibilité ne sont pas des excuses valables pour un style de travail non systématique et ad hoc dans l'IE.

Cependant, il n'existe pas de processus universel d'IE ni d'ensemble universel de pratiques d'IE qui fonctionnent bien dans toutes les situations données ou du moins dans la plupart des situations : il n'y a pas de "taille unique" en matière d'IE.

Un travail systématique et discipliné signifie que les ingénieurs des exigences :

- Configurent un processus d'IE qui soit bien adapté au problème posé et qui corresponde bien au processus utilisé pour développer le système (voir le chapitre 5).
- Parmi l'ensemble des pratiques et des produits d'activités disponibles en matière d'IE, sélectionnent ceux qui conviennent le mieux au problème, au contexte et à l'environnement de travail donnés (voir les chapitres 3, 4 et 6).
- N'utilisent pas toujours les mêmes processus, pratiques et produits d'activités.

- Ne réutilisent pas sans réflexion les processus et les pratiques issus de précédents projets d'IE ayant réussi.

2.3 Pour en savoir plus

Glinz [Glin2008] discute de la valeur des exigences de qualité et des exigences en général [Glin2016].

Glinz et Wieringa [GlWi2007] expliquent la notion et l'importance des parties prenantes.

Glinz et Fricker [GlFr2015] discutent du rôle et de l'importance d'une compréhension commune.

Les articles de Jackson [Jack1995b] et de Gunter et al. [GGJZ2000] sont fondamentales pour le problème des exigences dans le contexte. Le rôle du contexte dans l'IE est également examiné par Pohl [Pohl2010].

Gause et Weinberg [GaWe1989] discutent de l'interdépendance des problèmes et des solutions. Swartout et Balzer [SwBa1982] ont été les premiers à souligner qu'il est rarement possible de créer une spécification complète avant de commencer la mise en œuvre.

La validation est abordée dans tous les manuels d'IE. Grünbacher et Seyff [GrSe2005] expliquent comment parvenir à un accord en négociant sur les exigences.

Kano et al. [KSTT1984] ont été parmi les premiers à souligner le rôle de l'innovation. Maalej, Nayebi, Johann et Ruhe [MNJR2016] discutent de l'utilisation du retour d'information explicite et implicite de l'utilisateur pour l'IE. Maiden, Gitzikis et Robertson [MaGR2004] examinent comment la créativité peut favoriser l'innovation dans le domaine de l'ingénierie des exigences. Gorschek et al. [GFPK2010] définissent un processus d'innovation systématique.

3 Produits d'activités et pratiques de documentation

L'ingénierie des exigences (IE) traditionnelle prévoit la rédaction d'un cahier des charges complet, exhaustif et sans ambiguïté [IEEE830], [Glin2016]. Bien qu'il soit encore approprié de créer des spécifications complètes dans de nombreux cas, il y a aussi de nombreux autres cas où le coût de la rédaction de ces spécifications est supérieur à leur avantage. Par exemple, des cahiers des charges complets sont utiles, voire nécessaires, en cas d'appel d'offres ou d'externalisation de la conception et de la mise en œuvre d'un système, ou lorsqu'un système est essentiel pour la sécurité et qu'il faut se conformer à la réglementation. En revanche, lorsque les parties prenantes et les développeurs unissent leurs forces pour définir et développer un système de manière itérative, la rédaction d'un cahier des charges complet n'a pas de sens. Il est donc essentiel dans l'IE d'adapter la documentation au contexte du projet et de sélectionner des produits d'activités pour documenter les exigences et les informations liées aux exigences qui apportent une valeur optimale au projet.

Dans ce chapitre, vous découvrirez les produits d'activités typiques de l'IE et vous apprendrez à les créer.

3.1 Produits d'activités dans l'ingénierie des exigences

Il existe une variété de produits d'activités qui sont utilisés dans l'IE.

> Définition 3.1. Produit d'activités *(Work product)* :
> A recorded intermediate or final result generated in a work process.

Nous considérons le terme *artefact* comme un synonyme de produit d'activités. Nous préférons le terme de produit d'activités à celui d'artefact pour exprimer la connotation selon laquelle un produit d'activités est le résultat d'un travail effectué dans le cadre d'un processus de travail.

Selon cette définition, un produit de travail d'ER peut être tout ce qui exprime des exigences, d'une simple phrase ou d'un diagramme à une spécification des exigences du système qui couvre des centaines de pages, ou des informations liées aux exigences telles qu'un glossaire. Il est également important de noter qu'un produit d'activités peut contenir d'autres produits d'activités.

3.1.1 Caractéristiques des produits d'activités

Les produits d'activités peuvent être caractérisés par les aspects suivants : *objectif*, *taille*, *représentation*, *durée de vie* et *stockage*.

Table 3.1 donne une vue d'ensemble des produits d'activités typiques utilisés dans l'IE, avec leur objectif respectif (c'est-à-dire ce que le produit d'activités spécifie ou fournit) et leur taille typique. Le tableau est structuré en quatre groupes : produits d'activités pour des exigences uniques, ensembles cohérents d'exigences, documents ou structures de documentation, et autres produits d'activités.

Il existe de nombreuses façons de représenter un produit d'activités. Dans l'IE, les représentations basées sur le *langage naturel*, les *templates* et les *modèles* sont particulièrement importants. Ces questions sont abordées dans les sections 3.2, 3.3, et 3.4, respectivement. Il existe d'autres représentations, telles que les dessins ou les prototypes, qui sont traitées à la section3.7.

Chaque produit a une *durée de vie*. Il s'agit de la période qui s'écoule entre la création du produit d'activités et le moment où le produit d'activités est jeté ou devient sans objet. En ce qui concerne la durée de vie, nous distinguons trois catégories de produits d'activités : les produits d'activités *temporaires*, les produits d'activités *évolutifs* et les produits d'activités *durables*.

Des *produits d'activités temporaires* sont créés pour soutenir la communication et créer une compréhension commune (par exemple, une esquisse d'une interaction utilisateur-système créée dans un atelier). Les produits d'activités temporaires sont jetés après utilisation ; aucune métadonnée n'est conservée sur ces produits d'activités.

Les *produits d'activités évolutifs* émergent en plusieurs itérations au fil du temps (par exemple, une collection de récits d'utilisateurs dont le nombre et le contenu augmentent). Certaines métadonnées (au moins le propriétaire, le statut et l'historique des révisions) doivent être conservées pour chaque produit d'activités évolutif. En fonction de l'importance et du statut d'un produit d'activités, des procédures de contrôle des changements doivent être appliquées lors de la modification d'un produit d'activités en évolution.

Des *produits d'activités durables* ont été définis ou publiés (par exemple, une spécification d'exigences qui fait partie d'un contrat ou un carnet de commandes qui est implémentée dans une itération donnée). Un ensemble complet de métadonnées doit être conservé pour gérer correctement le produit d'activités et un processus de modification élaboré doit être suivi pour modifier un produit d'activités durable (chapitre 6).

Un produit d'activités temporaire peut devenir un produit évolutif lorsque les ingénieurs des exigences décident de conserver un produit d'activités et de le développer davantage. Dans ce cas, il convient d'ajouter des métadonnées afin de contrôler l'évolution du produit. Lorsqu'un produit d'activités évolutif fait l'objet d'une définition de base ou d'une publication, sa durée de vie passe d'évolutive à durable.

Table 3.1 Vue d'ensemble des produits d'activités de l'IE

Produit d'activités	Objectif : le produit d'activités spécifie / fournit	Taille*
Exigences uniques		
Exigence individuelle	Une seule exigence, généralement sous forme de texte	S
User Story	Une fonction ou un comportement du point de vue d'une partie prenante	S
Des ensembles cohérents d'exigences		
Cas d'utilisation	Fonction d'un système du point de vue d'un acteur ou d'un utilisateur	S–M
Modèle graphique	Divers aspects, par exemple le contexte, la fonction, le comportement (voir section 3.4)	M
Description de la tâche	Tâche qu'un système doit accomplir	S–M
Description de l'interface externe	Les informations échangées entre un système et un acteur dans le contexte du système	M
Epic	Une vue d'ensemble des besoins des parties prenantes	M
Feature	Caractéristique distinctive d'un système	S–M
Documents et structures de documentation		
Spécification des exigences système**	Un document d'exigences complet	L–XL
Backlog de produit et backlog de sprint	Une liste des éléments de travail, y compris les exigences	M–L
Story map	Un arrangement visuel des user stories	M
Vision	L'imagination conceptuelle d'un futur système	M
Autres produits d'activités		
Glossaire	Terminologie commune non ambiguë et convenue	M
Note textuelle ou croquis graphique	Un mémo pour la communication et la compréhension	S
Prototype	Une spécification par l'exemple, en particulier pour comprendre, valider et négocier les exigences	S–L

* : S : Petit, M : Moyen, L : Grand, XL : Très grand
** : D'autres exemples sont : la spécification des besoins métiers, la spécification des besoins du domaine, la spécification des besoins des parties prenantes/utilisateurs ou la spécification des besoins du logiciel

De nos jours, la plupart des produits d'activités sont stockés électroniquement sous forme de fichiers, dans des bases de données ou dans des outils d'IE. Des produits d'activités informels et temporaires peuvent également être stockés sur d'autres supports – par exemple, du papier ou des post-it sur un tableau Kanban.

3.1.2 Niveaux d'abstraction

Les exigences et les produits d'activités correspondants se situent à différents niveaux d'abstraction – allant, par exemple, des exigences de haut niveau pour un nouveau processus métier, jusqu'à des exigences à un niveau très détaillé, comme la réaction d'un composant logiciel spécifique à un événement exceptionnel.

Les exigences métier, les exigences du domaine et les exigences des parties prenantes/utilisateurs se situent généralement à un niveau d'abstraction plus élevé que les exigences du système. Lorsqu'un système est constitué d'une hiérarchie de sous-systèmes et de composants, nous avons des exigences de système aux niveaux d'abstraction correspondants pour les sous-systèmes et les composants.

Lorsque les exigences métier et des parties prenantes sont exprimées dans des produits d'activités durables, tels que les spécifications des exigences métier, les spécifications des exigences des parties prenantes ou les documents de vision, elles précèdent la spécification des exigences du système. Par exemple, dans les situations contractuelles, lorsqu'un client commande le développement d'un système à un fournisseur, le client crée et publie fréquemment une spécification des exigences des parties prenantes. Le fournisseur s'en sert ensuite comme base pour établir une spécification des exigences système. Dans d'autres projets, les exigences métier, les exigences des parties prenantes et les exigences du système peuvent évoluer conjointement.

Certains produits d'activités, tels que les exigences individuelles, les esquisses ou les modèles de processus, se retrouvent à tous ces niveaux. D'autres produits d'activités sont spécifiquement associés à certains niveaux. Par exemple, une spécification des exigences système est associée au niveau système. Notez qu'une exigence individuelle à un niveau d'abstraction élevé peut être raffinée en plusieurs exigences détaillées à des niveaux plus concrets.

Le choix du niveau d'abstraction approprié dépend notamment du sujet à spécifier et de l'objectif de la spécification. Par exemple, si le sujet à spécifier est une partie de bas niveau du problème à résoudre, il sera spécifié à un niveau d'abstraction assez bas. Il est toutefois important de ne pas mélanger des exigences qui se situent à des niveaux d'abstraction différents. Par exemple, dans la spécification d'un système d'information sur les soins de

santé, lors de la rédaction d'une exigence détaillée concernant les photos sur les cartes d'identité des clients, le paragraphe suivant ne doit pas énoncer un objectif général du système tel que la réduction des coûts des soins de santé tout en maintenant le niveau de service actuel pour les clients. Dans les produits d'activités de petite et de moyenne taille (par exemple, les user stories ou les cas d'utilisation), les exigences doivent se situer à peu près au même niveau d'abstraction . Dans les grands produits d'activités tels que la spécification des exigences du système, les exigences des différents niveaux d'abstraction doivent être maintenues séparées en structurant la spécification en conséquence (Section 3.6).

Les exigences se situent naturellement à différents niveaux d'abstraction. Il est utile de sélectionner des produits d'activités adaptés à un niveau d'abstraction donné et de structurer correctement les produits d'activités qui contiennent des exigences à plusieurs niveaux d'abstraction.

3.1.3 Niveau de détail

Lors de la spécification des exigences, les ingénieurs des exigences doivent décider du niveau de détail avec lequel les exigences doivent être spécifiées. Cependant, décider du niveau de détail approprié ou même optimal pour un besoin donné est une tâche difficile.

Par exemple, dans une situation où le client (utilisateur) et le fournisseur d'un système collaborent étroitement, il peut être suffisant d'énoncer une exigence concernant un formulaire de saisie des données comme suit : "Le système doit fournir un formulaire pour la saisie des données personnelles du client (utilisateur)". En revanche, lorsque la conception et la mise en œuvre du système sont confiées à un fournisseur ayant peu ou pas de connaissances dans le domaine, une spécification détaillée du formulaire d'entrée du client sera nécessaire.

Le niveau de détail auquel les exigences doivent être spécifiées dépend de plusieurs facteurs, notamment :

- Le problème et le contexte du projet : plus le problème est difficile et moins les ingénieurs chargés des exigences et les développeurs sont familiarisés avec le contexte du projet, plus il est nécessaire de fournir des détails.
- Le degré de compréhension commune du problème : lorsque la compréhension commune implicite est faible (voir le principe 3 au chapitre 2), des spécifications explicites et détaillées sont nécessaires pour créer le degré nécessaire de compréhension commune.
- Le degré de liberté laissé aux concepteurs et aux programmeurs : des exigences moins détaillées donnent plus de liberté aux développeurs.
- La disponibilité d'un retour d'information rapide de la part des parties prenantes pendant la conception et la mise en œuvre : lorsqu'un retour d'information rapide est disponible, des spécifications moins détaillées suffisent à contrôler le risque de développer un mauvais système.

- Le coût par rapport à la valeur d'une spécification détaillée : plus l'avantage d'une exigence est élevé, plus nous pouvons nous permettre de la spécifier en détail.
- Les normes et réglementations : les normes imposées et les contraintes réglementaires peuvent signifier que les exigences doivent être spécifiées de manière plus détaillée qu'elles ne le seraient autrement.

Il n'existe pas de niveau de détail universellement "correct" pour les exigences. Pour chaque exigence, le niveau de détail adéquat dépend de nombreux facteurs. Plus le niveau de détail des exigences spécifiées est élevé, plus le risque d'obtenir un produit présentant des fonctionnalités ou des propriétés inattendues ou manquantes est faible. Cependant, le coût de la spécification augmente à mesure que le niveau de détail augmente.

3.1.4 Aspects à prendre en compte

Quels que soient les produits d'activités IE utilisés, plusieurs aspects doivent être pris en compte lors de la spécification des exigences [Glin2019].

Tout d'abord, comme il existe des exigences fonctionnelles, des exigences de qualité et des contraintes (voir la section 1.1), les ingénieurs des exigences doivent s'assurer qu'ils couvrent les trois types d'exigences lorsqu'ils documentent les exigences. Dans la pratique, les parties prenantes ont tendance à omettre les exigences de qualité parce qu'elles les considèrent comme allant de soi.

Ils ont également tendance à spécifier les contraintes en tant qu'exigences fonctionnelles. Il est donc important que les ingénieurs en charge des exigences s'y retrouvent.

Lorsqu'on examine les exigences fonctionnelles, on constate qu'elles portent sur différents aspects, comme, par exemple, une structure de données requise, un ordre d'actions requis ou la réaction requise à un événement extérieur. Nous distinguons trois aspects principaux : la *structure et les données*, la *fonction et le flux*, et l *'état et le comportement*.

L'aspect *structure et données* se concentre sur les exigences concernant la structure statique d'un système et les données (persistantes) qu'un système doit connaître afin d'exécuter les fonctions requises et de fournir les résultats nécessaires.

L'aspect *fonction et flux* traite des fonctions qu'un système doit fournir et du flux de contrôle et de données au sein des fonctions et entre elles pour créer les résultats requis à partir d'entrées données.

L'aspect *état et comportement* se concentre sur la spécification du comportement d'un système en fonction de l'état – en particulier, comment un système doit réagir à tel ou tel événement externe en fonction de l'état actuel du système.

Lorsqu'il s'agit d'*exigences de qualité*, telles que la facilité d'utilisation, la fiabilité ou la disponibilité, un modèle de qualité – par exemple, le modèle fourni par la norme ISO/IEC 25010 [ISO25010]– peut être utilisé comme checklist.

Parmi les exigences de qualité, les *exigences de performance* revêtent une importance particulière. Les exigences de performance portent sur :

- Le temps (par exemple, pour effectuer une tâche ou réagir à des événements extérieurs)
- Le volume (par exemple, taille de la base de données requise)
- La fréquence (par exemple, calcul d'une fonction ou réception de stimuli provenant de capteurs)
- Le débit (par exemple, transmission de données ou taux de transaction)
- Consommation de ressources (par exemple, CPU, stockage, bande passante, batterie)

Certains considèrent également la précision requise d'un calcul comme une exigence de performance.

Dans la mesure du possible, des valeurs mesurables doivent être spécifiées. Lorsque les valeurs suivent une distribution de probabilité, il ne suffit pas de spécifier la moyenne. Si la fonction de distribution et ses paramètres ne peuvent être spécifiés, les ingénieurs des exigences doivent s'efforcer de spécifier des valeurs minimales et maximales ou des valeurs à 95 % en plus des moyennes.

Il est notoirement difficile de documenter les exigences de qualité au-delà des exigences de performance.

Les *représentations qualitatives,* telles que "Le système doit être sûr et facile à utiliser", sont ambiguës et donc difficiles à réaliser et à valider.

Les *représentations quantitatives* sont mesurables, ce qui est un atout important pour atteindre et valider systématiquement une exigence de qualité. Cependant, ils soulèvent des difficultés principales (par exemple, comment peut-on définir la sécurité en termes quantitatifs ?) et peuvent être assez coûteux à spécifier.

Les *représentations opérationnalisées* énoncent une exigence de qualité en termes d'exigences fonctionnelles pour atteindre la qualité souhaitée. Par exemple, une exigence de sécurité des données peut être exprimée en termes de fonction de connexion qui limite l'accès aux données et de fonction qui crypte les données stockées. Les représentations opérationnalisées rendent les exigences de qualité testables, mais peuvent également impliquer des décisions de conception prématurées.

La règle souvent entendue "Seule une exigence de qualité quantifiée est une bonne exigence de qualité" est dépassée et peut conduire à des exigences de qualité ayant une valeur faible, voire négative, en raison de l'effort important que représente la quantification. Au lieu de cela, il convient d'adopter une approche fondée sur les risques [Glin2008].

Des représentations qualitatives des exigences de qualité *suffisent* dans les situations suivantes :

- Il y a suffisamment de compréhension implicite partagée entre les parties prenantes, les ingénieurs des exigences et les développeurs.

- Les parties prenantes, les ingénieurs des exigences et les développeurs se mettent d'accord sur une solution connue qui répond aux exigences.
- Les parties prenantes ne veulent donner que des indications générales sur la qualité et font confiance aux développeurs pour régler les détails.
- De courtes boucles de rétroaction sont en place afin que les problèmes puissent être détectés rapidement.

Lorsque les développeurs sont *capables de généraliser à partir d'exemples*, la spécification des exigences de qualité en termes d'exemples quantifiés ou de comparaisons avec un système existant est un moyen peu coûteux et efficace de documenter les exigences de qualité.

Ce n'est que dans les cas où il existe un *risque élevé* de ne pas répondre aux besoins des parties prenantes, en particulier lorsque les exigences de qualité sont *critiques pour la sécurité*, qu'une représentation entièrement quantifiée ou une opérationnalisation en termes d'exigences fonctionnelles doit être envisagée.

Lors de la définition des *contraintes*, il convient de tenir compte des catégories de contraintes suivantes :

- *Technique*: interfaces ou protocoles donnés, composants ou cadres à utiliser, etc.
- *Juridique*: restrictions imposées par des lois, des contrats, des normes ou des règlements
- *Organisationnel*: il peut y avoir des contraintes en termes de structures organisationnelles, de processus ou de politiques qui ne doivent pas être modifiées par le système.
- *Culturel*: les habitudes et les attentes des utilisateurs sont, dans une certaine mesure, façonnées par la culture dans laquelle ils vivent. Il s'agit d'un aspect particulièrement important à prendre en compte lorsque les utilisateurs d'un système sont issus de cultures différentes ou lorsque les ingénieurs des exigences et les développeurs sont enracinés dans une culture différente de celle des utilisateurs du système.
- *Environnement*: lors de la spécification des systèmes cyber-physiques, les conditions environnementales telles que la température, l'humidité, les radiations ou les vibrations peuvent être considérées comme des contraintes ; la consommation d'énergie et la dissipation de chaleur peuvent constituer d'autres contraintes.
- *Physique*: lorsqu'un système comprend des composants physiques ou interagit avec eux, il est soumis aux lois de la physique et aux propriétés des matériaux utilisés pour les composants physiques.
- En outre, les *solutions particulières ou les restrictions exigées par des parties prenantes importantes* constituent également des contraintes.

Enfin, les exigences ne peuvent être comprises que dans leur *contexte* (voir Principe 4 du chapitre 2). Par conséquent, un autre aspect doit être pris en considération, que nous appelons le *contexte et les limites*.

L'aspect *contexte et limites* couvre les exigences et les hypothèses du domaine dans le contexte du système, ainsi que les acteurs externes avec lesquels le système interagit et les interfaces externes entre le système et son environnement au niveau des limites du système.

Il existe de nombreuses interrelations et dépendances entre les aspects mentionnés ci-dessus. Par exemple, une requête émise par un utilisateur (contexte) peut être reçue par le système via une interface externe (limite), déclencher une transition d'état du système (état et comportement), qui déclenche une action (fonction) suivie d'une autre action (flux) qui nécessite des données avec une structure donnée (structure et données) pour fournir un résultat à l'utilisateur (contexte) dans un intervalle de temps donné (qualité).

Certains produits d'activités se concentrent sur un aspect spécifique et font abstraction des autres aspects. C'est notamment le cas des modèles d'exigences (Section 3.4). D'autres produits d'activités, tels qu'une spécification des exigences du système, couvrent tous ces aspects. Lorsque différents aspects sont documentés dans des produits d'activités distincts ou dans des chapitres distincts du même produit d'activités, ces produits ou chapitres doivent être cohérents entre eux.

De nombreux aspects différents doivent être pris en compte lors de la documentation des exigences, en particulier la fonctionnalité (structure et données, fonction et flux, état et comportement), la qualité, les contraintes et le contexte environnant (contexte et limites).

3.1.5 Lignes directrices pour la documentation générale

Indépendamment des techniques utilisées, certaines lignes directrices générales doivent être suivies lors de la création de produits d'IE :

- Sélectionnez un type de produit d'activités correspondant à l'objectif visé.
- Évitez la redondance en référençant le contenu au lieu de répéter le même contenu.
- Évitez les incohérences entre les produits d'activités, en particulier lorsqu'ils couvrent différents aspects.
- Utilisez les termes de manière cohérente, comme définis dans le glossaire.
- Structurez correctement les produits d'activités, par exemple en utilisant des structures standard.

3.1.6 Planification des produits d'activités

Chaque configuration de projet et chaque domaine étant différents, l'ensemble des produits d'activités qui en résultent doit être défini pour chaque situation. Les parties concernées, en particulier les ingénieurs des exigences, les parties prenantes et les propriétaires ou gestionnaires du projet/produit doivent se mettre d'accord sur les points suivants :

- Dans quels produits d'activités les exigences doivent-elles être enregistrées et à quelles fins (voir Table 3.1)?
- Quels sont les niveaux d'abstraction à prendre en compte (Section 3.1.2)?

- Jusqu'à quel niveau de détail les exigences doivent-elles être documentées pour chaque niveau d'abstraction (Section 03.1.3)?
- Comment les exigences doivent-elles être représentées dans ces produits d'activités (par exemple, en langage naturel ou sur la base de modèles, voir ci-dessous) et quelle(s) notation(s) doit-on utiliser ?

Les ingénieurs des exigences doivent définir les produits d'activités d'IE à utiliser dès les premières étapes d'un projet. Une définition si précoce :

- Aide à la planification des efforts et des ressources
- Veiller à ce que les notations appropriées soient utilisées
- Veiller à ce que tous les résultats soient enregistrés dans les bons produits d'activités
- Veiller à ce qu'aucun remaniement majeur de l'information ou "mise au point finale" ne soit nécessaire
- Contribuer à éviter les redondances, ce qui se traduit par moins de travail et une maintenabilité plus facile

3.2 Produits d'activités basés sur le langage naturel

Le langage naturel, tant à l'oral qu'à l'écrit, a toujours été un moyen essentiel pour communiquer les exigences des systèmes. L'utilisation du langage naturel pour rédiger les produits d'activités de l'IE présente de nombreux avantages. En particulier, le langage naturel est extrêmement expressif et flexible, ce qui signifie que presque toutes les exigences concevables, sous quelque aspect que ce soit, peuvent être exprimées en langage naturel. De plus, le langage naturel est utilisé dans la vie quotidienne et enseigné à l'école, de sorte qu'aucune formation spécifique n'est nécessaire pour lire et comprendre les exigences rédigées dans ce langage.

L'évolution humaine a façonné le langage naturel comme un moyen de *communication orale entre des personnes en interaction directe*, où les malentendus et les informations manquantes peuvent être détectés et corrigés rapidement. Le langage naturel *n'* est donc *pas* optimisé pour une communication précise, non ambiguë et complète au moyen de documents écrits. Cela constitue un problème majeur lorsqu'il s'agit de rédiger des documents techniques (tels que des exigences) en langage naturel. Contrairement à la communication en langue naturelle *parlée*, où la communication est contextualisée et interactive avec un retour d'information immédiat, il n'existe aucun moyen naturel de détecter et de corriger rapidement les ambiguïtés, les omissions et les incohérences dans les textes *écrits* en langue naturelle. Au contraire, la recherche d'ambiguïtés, d'omissions et d'incohérences dans les textes écrits est difficile et coûteuse, en particulier pour les produits d'activités qui contiennent une grande quantité de texte en langage naturel.

Le problème peut être atténué dans une certaine mesure en rédigeant la documentation technique de manière consciente, en suivant des règles éprouvées et en évitant les pièges connus.

Lorsqu'ils rédigent des exigences en langage naturel, les ingénieurs des exigences peuvent éviter de nombreux malentendus potentiels en appliquant quelques règles simples :

- Rédiger des phrases courtes et bien structurées. La règle empirique consiste à exprimer un besoin unique en une phrase en langage naturel. Pour obtenir une bonne structure, les ingénieurs des exigences doivent utiliser des gabarits de phrases (section 3.3).
- Créer des produits d'activités bien structurés. Outre la rédaction de phrases bien structurées (voir ci-dessus), les produits d'activités rédigés en langue naturelle doivent également être bien structurés dans leur ensemble. Une façon éprouvée d'y parvenir consiste à utiliser une structure hiérarchique composée de parties, de chapitres, de sections et de sous-sections, comme c'est généralement le cas dans les ouvrages techniques. Les modèles de documents (section 3.3) vous aident à obtenir une bonne structure.
- Définir et utiliser de manière cohérente une terminologie uniforme. La création et l'utilisation d'un glossaire (section 0) est le principal moyen d'éviter les malentendus et les incohérences terminologiques.
- Éviter d'utiliser des termes et des expressions vagues ou ambigus.
- Connaître et éviter les pièges de la rédaction technique (voir ci-dessous).

Lors de la rédaction de documents techniques en langage naturel, il existe des pièges bien connus qu'il convient d'éviter ou des éléments à utiliser avec précaution (voir, par exemple, [GoRu2003]).

Les ingénieurs des exigences doivent *éviter de* rédiger des exigences qui contiennent les éléments suivants :

- *Descriptions incomplètes*. Dans le langage naturel, les verbes sont généralement accompagnés d'un ensemble d'espaces réservés pour les noms ou les pronoms. Par exemple, le verbe "donner" comporte trois espaces réservés pour indiquer *qui* donne *quoi* à *qui*. Lors de la rédaction d'une exigence en langage naturel, tous les espaces réservés du verbe utilisé doivent être remplis.
- *Noms non spécifiques*. L'utilisation de noms tels que "les données" ou "l'utilisateur" laisse trop de place à des interprétations différentes par les différentes parties prenantes ou les développeurs. Ils doivent être remplacés par des noms plus spécifiques ou être rendus plus spécifiques par l'ajout d'adjectifs ou l'attribution d'un type bien défini.
- *Conditions incomplètes*. Lorsqu'elles décrivent ce qui doit être fait, de nombreuses personnes se concentrent sur le cas normal, en omettant les cas exceptionnels. Dans la rédaction technique, il s'agit d'un piège à éviter : lorsque quelque chose ne se produit que si certaines conditions sont remplies, ces conditions doivent être énoncées, en prévoyant des clauses *"alors*" et "sinon*"*.
- *Comparaisons incomplètes*. Dans la communication orale, les gens ont tendance à utiliser des comparatifs (par exemple, "la nouvelle application vidéo est bien meilleure") sans dire à quoi ils se comparent, en supposant généralement que le

contexte est clair. Dans la rédaction technique, les comparaisons doivent inclure un objet de référence, par exemple "plus rapide *que 0,1 ms*".

Il existe d'autres éléments que les ingénieurs des exigences doivent utiliser avec précaution, car ils constituent des pièges potentiels :

- *Voix passive*. Les phrases à la voix passive n'ont pas de sujet agissant. Si une exigence est formulée à la voix passive, cela peut cacher qui est responsable de l'action décrite dans l'exigence, ce qui conduit à une description incomplète.
- *Quantificateurs universels*. Les quantificateurs universels sont des mots tels que *"tout"*, " *toujours*" ou " *jamais"*, qui sont utilisés pour formuler des affirmations universellement vraies. Dans les systèmes techniques, cependant, de telles propriétés universelles sont rares. Lorsque les ingénieurs des exigences utilisent un quantificateur universel, ils doivent se demander s'ils énoncent une propriété véritablement universelle ou s'ils spécifient plutôt une règle générale qui comporte des exceptions (qu'ils doivent également spécifier). Ils doivent faire preuve de la même prudence lorsqu'ils utilisent des clauses de type "ou bien, ou bien" qui, de par leur sémantique, excluent tout autre cas d'exception.
- *Nominalisations*. Lorsqu'un nom est dérivé d'un verbe (par exemple, "authentification" à partir de "authentifier"), les linguistes parlent de nominalisation. Lorsqu'ils spécifient des exigences, les ingénieurs en exigences doivent manipuler les nominalisations avec précaution, car une nominalisation peut cacher des exigences non spécifiées. Par exemple, l'exigence "Le système ne doit permettre à l'utilisateur d'accéder à (...) qu'après une authentification réussie" implique l'existence d'une procédure d'authentification des utilisateurs. Lors de la rédaction d'une telle exigence, l'ingénieur des exigences doit donc vérifier s'il existe également des exigences relatives à la procédure d'authentification des utilisateurs légitimes.

Le langage naturel est un moyen très puissant pour rédiger des exigences. Pour atténuer les inconvénients inhérents à l'utilisation du langage naturel dans la documentation technique, les ingénieurs des exigences doivent suivre des règles de rédaction éprouvées et éviter les pièges bien connus.

3.3 Produits d'activités basés sur des templates

Comme indiqué à la section 3.2, l'utilisation de templates est un moyen éprouvé de rédiger des produits d'activités de qualité et bien structurés en langage naturel et d'atténuer ainsi certaines des faiblesses du langage naturel pour la rédaction technique. Un template est une sorte de plan prêt à l'emploi pour la structure syntaxique d'un produit d'activités. Lors de l'utilisation du langage naturel dans l'IE, nous distinguons trois catégories de templates : les gabarits de phrases, les modèles de formulaires et les modèles de documents.

3.3.1 Gabarits de phrases

> **Définition 3.2. Gabarit de phrase *(Phrase template)* :**
> A template for the syntactic structure of a phrase that expresses an individual requirement or a user story in natural language.

Un gabarit de phrase fournit un squelette de structure avec des espaces réservés, dans lequel les ingénieurs des exigences remplissent les espaces réservés afin d'obtenir des phrases bien structurées et uniformes qui expriment les exigences.

L'utilisation de gabarits de phrases est une bonne pratique pour la rédaction d'exigences individuelles en langage naturel et pour la rédaction de user stories.

3.3.1.1 Gabarits de phrases pour les exigences individuelles

Différents gabarits de phrases pour la rédaction d'exigences individuelles ont été définis, par exemple dans [ISO29148], [MWHN2009], et [Rupp2020]. La norme ISO/IEC/IEEE 29148 [ISO29148] fournit un gabarit unique et uniforme pour les exigences individuelles comme suit :

> [<Condition>] <Sujet> <Action> <Objets> [<Restriction>].

Exemple : Lorsqu'une carte valide est détectée, le système doit afficher le message "Entrer votre PIN" sur l'écran de dialogue dans un délai de 200 ms.

Lors de la formulation d'une action à l'aide de ce modèle, les conventions suivantes concernant l'utilisation des verbes auxiliaires sont fréquemment utilisées dans la pratique :

- Le terme "*doit* " indique une exigence obligatoire.
- Le terme "*devrait* " indique une exigence qui n'est pas obligatoire mais fortement souhaitée.
- Le mot "*peut*" indique une suggestion.

Le *mot "devra"* (ou l'utilisation d'un *verbe au présent* sans l'un des verbes auxiliaires mentionnés ci-dessus) désigne une déclaration factuelle qui n'est pas considérée comme une exigence.

Lorsqu'il n'y a pas de signification convenue pour les verbes auxiliaires dans un projet, ou en cas de doute, des définitions telles que celles données ci-dessus doivent faire partie d'une spécification des besoins.

EARS (Easy Approach to Requirements Syntax – Approche simplifiée de la syntaxe des exigences) [MWHN2009] fournit un ensemble de gabarits de phrases adaptés à différentes situations, comme décrit ci-dessous.

Exigences omniprésentes (doivent toujours être respectées) :

> Le <nom du système> doit <réponse du système>.

Exigences événementielles (déclenchées par un événement extérieur) :

> QUAND <conditions préalables facultatives> <déclencheur> le <nom du système> doit <réponse du système>.

Comportement indésirable (description des situations à éviter) :

> SI <conditions préalables facultatives> <déclencheur>, ALORS le <nom du système> doit <réponse du système>.

Note : Bien que le modèle de comportement indésirable soit similaire au modèle de comportement événementiel, Mavin et al. fournissent un gabarit distinct pour ce dernier, arguant que le comportement indésirable (principalement dû à des événements inattendus dans le contexte, tels que des échecs, des attaques ou des choses auxquelles personne n'a pensé), est une source majeure d'omissions dans l'IE.

Exigences imposées par l'État (ne s'appliquent que dans certains États) :

> PENDANT QUE <dans un état spécifique> le <nom du système> doit <réponse du système>.

Caractéristiques optionnelles (applicables uniquement si certaines caractéristiques sont incluses dans le système) :

> LORSQUE la <caractéristique est incluse> le <nom du système> doit <réponse du système>.

Dans la pratique, des phrases combinant les mots-clés QUAND, PENDANT QUE et LORSQUE peuvent être nécessaires pour exprimer des exigences complexes.

EARS a été conçu principalement pour la spécification de systèmes cyber-physiques. Cependant, il peut également être adapté à d'autres types de systèmes.

3.3.1.2 Gabarits de phrases pour les user stories

Le gabarit de phrase classique pour la rédaction de user stories a été introduit par Cohn [Cohn2004]:

> En tant que <rôle>, je veux <exigence> afin de <bénéfice>.

Exemple : "En tant que responsable hiérarchique, je souhaite effectuer des requêtes ad hoc dans le système comptable afin de pouvoir effectuer la planification financière de mon service."

Bien que Cohn ait désigné la partie <bénéfice> du modèle comme étant facultative, il est aujourd'hui d'usage de spécifier un bénéfice pour chaque user story.

Chaque user story doit être accompagnée d'un ensemble de *critères d'acceptation, c'est-à-dire* de critères que la mise en œuvre de la user story doit satisfaire pour être acceptée par les parties prenantes. Les critères d'acceptation rendent une user story plus concrète et moins ambiguë. Cela permet d'éviter les erreurs de mise en œuvre dues à des malentendus.

3.3.2 Gabarit de formulaires

Définition 3.3. Gabarit de formulaire *(Form template)* :

A template providing a form with predefined fields to be filled in.

Les gabarits de formulaires sont utilisés pour structurer les produits d'activités de taille moyenne tels que les cas d'utilisation. Cockburn [Cock2001] a présenté un gabarit de formulaire populaire pour les cas d'utilisation. [Laue2002] a proposé un gabarit de description de tâches.

Table 3.2 présente un gabarit de formulaire simple pour les cas d'utilisation. Chaque étape du flux peut être subdivisée en une action d'un acteur et une réponse du système.

Table 3.2 Un gabarit de formulaire simple pour rédiger des cas d'utilisation

Nom	< Une courte phrase verbale active >
Condition préalable	<Condition(s) qui doit(vent) être remplie(s) lorsque l'exécution du cas d'utilisation est déclenchée>
Condition de fin de succès	<Etat à la fin du cas d'utilisation>
Échec de la condition finale	<État en cas d'échec de l'exécution du cas d'utilisation>
Acteur principal	<Nom de l'acteur>
Autres acteurs	<Liste des autres acteurs impliqués, le cas échéant>
Déclencheur	<Evénement qui initie l'exécution du cas d'utilisation>
Débit normal	<Description du principal scénario de réussite dans une séquence d'étapes : <étape 1> <action 1> <étape 2> <action 2> ... <étape n> <action n> ... >
Flux alternatifs	<Description des étapes alternatives ou exceptionnelles, avec des références aux étapes correspondantes dans le flux normal
Extensions	<Extensions au flux normal (s'il y en a), avec des références aux étapes étendues du flux normal>
Informations connexes	<Champ facultatif pour des informations supplémentaires, telles que la performance, la fréquence, la relation avec d'autres cas d'utilisation, etc.

Les gabarits de formulaires sont également utiles pour rédiger des exigences de qualité sous une forme mesurable [Gilb1988].Table 3.3 propose un modèle de formulaire simple pour les exigences de qualité mesurables, ainsi qu'un exemple.

Table 3.3 Un gabarit de formulaire pour spécifier des exigences de qualité mesurables

Modèle		Exemple
ID	(Numéro de l'exigence)	R137.2
But	<Objectif qualitativement énoncé>	Confirmer immédiatement les réservations de chambres
Échelle	<Échelle de mesure de l'exigence>	Temps écoulé en secondes (échelle de rapport)
Compteur	<Procédure de mesure de l'exigence>	Horodatage des moments où l'utilisateur appuie sur le bouton "Réserver" et où l'application affiche la confirmation. Mesure du temps écoulé.
Minimum	<Qualité minimale acceptable à atteindre>	Moins de 5s dans au moins 95% des cas
Intervalle OK	<Intervalle de valeurs qui est OK et qui est visée>	Entre 0,5 et 3 s dans plus de 98% des cas
Souhaitée	<Qualité atteinte dans les meilleures conditions possibles>	Moins de 0,5s dans 100% des cas

3.3.3 Gabarits de documents

> **Définition 3.4. Gabarit de document *(Document template)* :**
> A template providing a predefined skeleton structure for a document.

Les gabarits de documents permettent de structurer systématiquement les documents d'exigences – par exemple, une spécification des exigences du système. Les gabarits de documents d'IE peuvent être trouvés dans des normes, par exemple dans [ISO29148]. Le gabarit Volere de Robertson et Robertson [RoRo2012], [Vole2026] est également populaire dans la pratique. Lorsqu'une spécification des exigences est incluse dans l'ensemble des produits d'activités qu'un client a commandés et qu'il paiera, ce client peut prescrire l'utilisation de gabarits de documents qu'il a lui-même fournis. Sur le site Figure 3.1, nous présentons un exemple de gabarit de document simple pour la spécification des exigences d'un système.

3.3.4 Avantages et inconvénients

L'utilisation de gabarits lors de la rédaction de produits d'IE en langage naturel présente des avantages majeurs. Les gabarits fournissent une structure claire et réutilisable pour les produits d'activités, leur donnent une apparence uniforme et améliorent ainsi la lisibilité des produits d'activités. Les gabarits vous aident également à saisir les informations les plus pertinentes et à réduire les erreurs d'omission. D'autre part, il existe un piège potentiel lorsque les ingénieurs des exigences utilisent les gabarits de manière mécanique, en se concentrant sur la structure syntaxique plutôt que sur le contenu, et en négligeant tout ce qui ne correspond pas au gabarit.

Partie	Sections
Partie I: Introduction	
	Objectif du système Périmètre du développement d'un système Parties prenantes
Partie II : Vue d'ensemble du système	
	Vision et Objectifs du système Limite et contexte du système Structure générale du système Caractéristiques de l'utilisateur
Partie III: Exigences du système	
	Organisées de manière hiérarchique selon la structure du système, en utilisant un système de numérotation hiérarchique pour les exigences. Par sous-système/composant : ▪ Exigences fonctionnelles (structure et données, fonction et flux, état et comportement) ▪ Exigences qualité ▪ Contraintes ▪ Interfaces
Références	
	Glossaire (s'il n'est pas géré comme un produit d'activités à part entière)
Annexes	
	Hypothèses et dépendances

Figure 3.1 Un gabarit simple de spécification des exigences système

L'utilisation de gabarit lors de la rédaction de produits d'IE en langage naturel améliore la qualité des produits, à condition que les gabarits ne soient pas utilisés à mauvais escient comme un simple exercice syntaxique.

3.4 Produits d'activités basés sur des modèles

Les exigences formulées en langage naturel peuvent être facilement lues par les personnes qui parlent la langue. Le langage naturel souffre d'ambiguïté en raison de l'imprécision de la sémantique des mots, des expressions et des phrases [Davi1993]. Cette imprécision peut entraîner des confusions et des omissions dans les exigences. Lorsque vous lisez des exigences textuelles, vous essayez de les interpréter à votre manière. Nous essayons souvent d'imaginer ces exigences dans notre esprit. Lorsque le nombre d'exigences est gérable, il est possible de conserver une vision et une vue d'ensemble des exigences textuelles. Lorsque le nombre d'exigences textuelles devient "trop important", nous perdons la vue d'ensemble. Cette limite est différente pour chaque personne. Le nombre d'exigences textuelles n'est pas la seule raison pour laquelle on perd la vision et la vue d'ensemble. La complexité des exigences, la relation entre les exigences et l'abstraction des exigences y contribuent également. Il se peut que vous deviez lire plusieurs fois les exigences formulées en langage naturel avant d'obtenir une image correcte et complète de ce à quoi le système doit se conformer. Notre capacité à traiter les exigences en langage naturel est limitée.

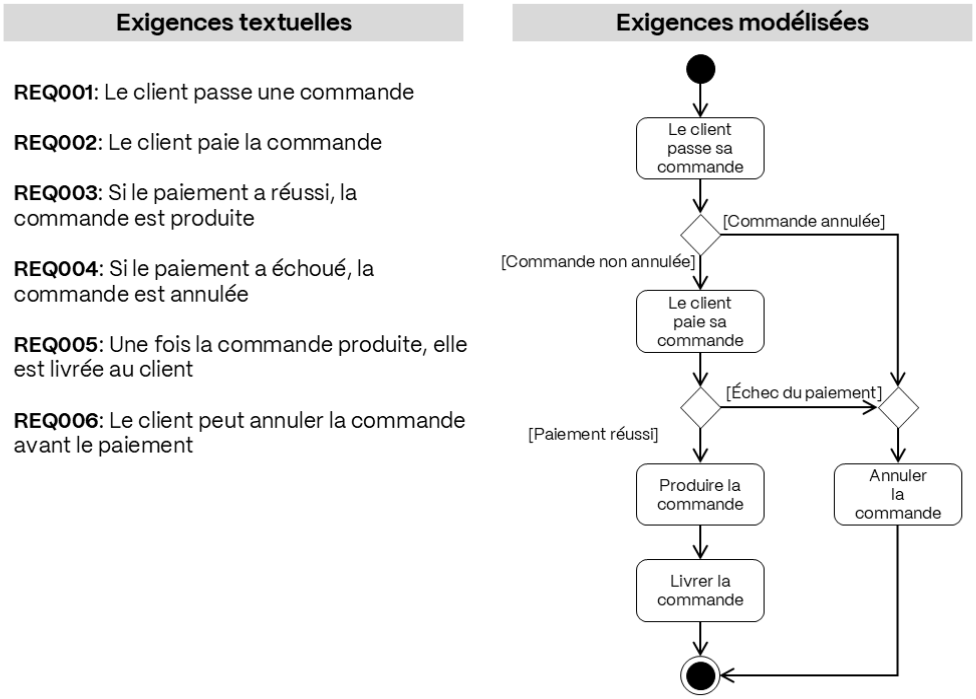


Figure 3.2 Exigences textuelles et exigences modélisées

Un modèle est une représentation abstraite d'une partie de la réalité existante ou d'une partie de la réalité à créer. Le fait de présenter les exigences (également) à l'aide d'un modèle (ou d'une image) contribuera à ce que les lecteurs comprennent les exigences. Cette représentation schématique d'un modèle est appelée diagramme.

Le diagramme de Figure 3.2 montre d'un coup d'œil ce que le système doit fournir, mais seulement si vous maîtrisez le langage de modélisation. Il est évident que si vous ne comprenez pas le diagramme, dans ce cas un diagramme d'activité UML, l'image ne contribuera pas à une meilleure compréhension des exigences.

Dans la section suivante (3.4.1), le concept de modèle d'exigences est expliqué. La modélisation des exigences et des objectifs de l'entreprise est expliquée à la section 3.4.6. Le modèle contextuel est une méthode importante pour décrire la délimitation d'un système. Des exemples de ce contexte sont présentés à la section 3.4.2. Les sections 3.4.3 à 3.4.5 donnent un certain nombre d'exemples de langages de modélisation qui sont souvent utilisés dans la pratique de l'ingénierie des systèmes.

3.4.1 Le rôle des modèles dans l'ingénierie des exigences

Comme tout langage, un langage de modélisation se compose de règles grammaticales et d'une description de la signification des constructions du langage, voir la section 3.4.1.1. Bien qu'un modèle soit une représentation visuelle de la réalité, les règles linguistiques sont importantes pour comprendre le modèle et ses nuances.

Il n'est pas toujours efficace de résumer les exigences dans un modèle. En comprenant les propriétés d'un modèle, nous pouvons mieux déterminer quand nous pouvons appliquer tel ou tel modèle, voir la section 3.4.1.2.

Tout comme le langage naturel présente des avantages et des inconvénients pour l'expression des exigences, il en va de même pour les modèles. Si nous observons ces faits lors de l'application d'un modèle, nous pouvons mieux déterminer la valeur ajoutée de l'application du "bon" modèle. Ce point est abordé à la section 3.4.1.3.

De nombreux modèles ont déjà été normalisés et sont utilisés dans divers domaines d'application, voir la section 3.4.1.4. Prenons l'exemple de la construction d'une maison, où un architecte utilise un modèle standardisé pour décrire la maison. Un exemple de modèles utilisés par les architectes est celui des modèles d'information du bâtiment (BIM) [ISO19650], qui modélisent les éléments nécessaires pour planifier, construire et gérer les bâtiments et autres éléments de construction.

Un autre exemple est celui de l'électronique, où le dessin des schémas électroniques est normalisé afin que les professionnels puissent comprendre, calculer et réaliser l'électronique.

Pour déterminer si un diagramme est appliqué correctement, nous pouvons valider les critères de qualité d'un diagramme. Ces critères sont décrits à la section 3.4.1.5.

3.4.1.1 Syntaxe et sémantique

Si vous pensez à une langue naturelle, par exemple votre langue maternelle, elle est définie par sa grammaire et sa sémantique.

La grammaire décrit les éléments (mots et phrases) et les règles auxquelles la langue doit obéir. Dans un langage de modélisation, on appelle cela la syntaxe, voir Figure 3.3. La syntaxe décrit les éléments de notation (symboles) utilisés dans le langage. Il décrit également comment ces éléments de notation peuvent être utilisés en combinaison.

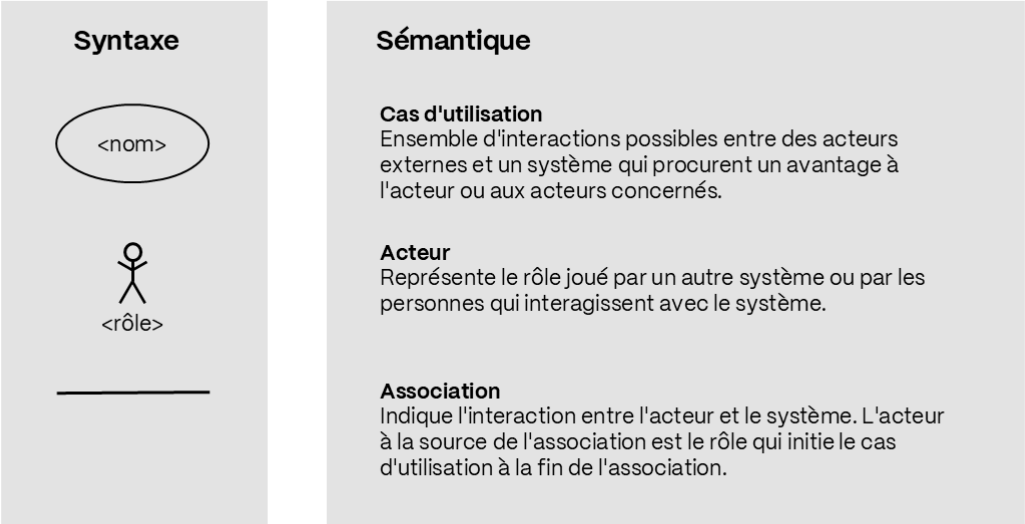


Figure 3.3 Modélisation de la syntaxe et de la sémantique des langues

La sémantique définit la signification des éléments de notation et la signification de la combinaison d'éléments. Il est essentiel de comprendre la signification des éléments de notation pour éviter le risque d'une mauvaise interprétation du modèle.

3.4.1.2 Propriétés d'un modèle

Un modèle d'exigences est un modèle conceptuel qui décrit les exigences du système à développer. Un modèle est également utilisé pour représenter la situation actuelle afin de comprendre, d'analyser et d'explorer les problèmes actuels. Dans ce contexte, conceptuel signifie que la réalité est réduite à son essence. Un modèle a un niveau d'abstraction élevé et réduit la réalité à ce qui est pertinent à ce niveau générique.

Un langage de modélisation conceptuel peut être normalisé (au niveau international) et est alors appelé langage de modélisation formel. Le langage de modélisation UML (Unified Modeling Language), très répandu et fréquemment utilisé, en est un exemple.

Un modèle possède un certain nombre de propriétés qui sont examinées plus en détail dans les sections suivantes :

- Un modèle est réalisé dans un but précis.
- Un modèle donne une représentation de la réalité.
- Un modèle est utilisé pour réduire l'information afin de mieux comprendre la réalité ou de se concentrer sur une partie de la réalité.

Un modèle est une représentation abstraite d'une partie de la réalité existante ou d'une partie de la réalité à créer. La notion de réalité comprend tout ensemble d'éléments, de phénomènes ou de concepts concevables, y compris d'autres modèles. La partie modélisée de la réalité est appelée l'original. Le processus de description de l'original peut être descriptif ou normatif.

La modélisation de l'original existant est appelée modélisation descriptive. Il montre la réalité actuelle et reflète les exigences qui sont satisfaites. Si aucun modèle de l'original n'est encore disponible, un tel modèle est le résultat de l'analyse de la situation actuelle.

La modélisation d'un original à créer est appelée modélisation prescriptive. Il indique quelle réalité future est attendue ou exigée. S'il existe un modèle aux propriétés descriptives pour une situation donnée, un modèle aux propriétés prescriptives peut être dérivé de l'original en indiquant les exigences qui seront nouvelles, modifiées ou qui ne sont plus nécessaires. Le modèle prescriptif décrit la situation future souhaitée.

La réalité peut être complexe. Si nous appliquons "trop" de détails, un modèle peut être difficile à comprendre. Cette réalité complexe peut être simplifiée en réduisant la quantité d'informations dans le modèle. Dans un modèle, nous pouvons omettre les informations non pertinentes. La réduction de la quantité d'informations peut nous permettre de mieux comprendre la réalité et d'en saisir plus facilement l'essence. En fonction de l'usage prévu (première propriété) pour lequel le modèle est appliqué, seules les informations pertinentes sont affichées dans le modèle.

Attention, si l'on réduit "trop" d'informations, il peut en résulter une image trouble ou incorrecte de la réalité. Il convient donc d'examiner attentivement dans quelle mesure les informations peuvent être réduites sans déformer la réalité.

Il existe plusieurs façons de réduire les informations :

- **Par compression ou agrégation**

  L'agrégation d'informations est un moyen de rendre l'information plus abstraite. L'information est dépouillée des détails non pertinents et est donc plus compacte. L'information est en quelque sorte condensée.

- **Par sélection**

  En ne sélectionnant que les informations pertinentes, et non pas tout, il est possible d'indiquer quel est le sujet traité. L'accent est mis sur une partie ou un nombre spécifique de parties du total.

Les deux méthodes de réduction de l'information peuvent également être appliquées conjointement.

Un modèle est une représentation de la réalité et chaque modèle représente certains aspects de la réalité. Par exemple, un plan de construction montre la répartition de l'espace dans un bâtiment et un schéma électrique montre le câblage du circuit électrique.

Les deux modèles représentent le bâtiment dans un but spécifique. Un modèle est conçu dans un but précis et dans un contexte précis. Dans l'exemple ci-dessus, le contexte est la conception et/ou la réalisation d'un bâtiment. Les différents plans de construction représentent des informations sur un aspect spécifique du bâtiment. Il est ainsi immédiatement clair qu'un modèle spécifique ne peut être utilisé que s'il correspond à l'usage pour lequel il a été conçu.

3.4.1.3 Avantages et inconvénients de la modélisation des exigences

Par rapport aux langues naturelles, les modèles présentent notamment les avantages suivants :

- Les éléments et leurs liens sont plus faciles à comprendre et à mémoriser.

  Une image en dit plus long que mille mots. Une image, ainsi qu'un modèle, peuvent être plus faciles à saisir et à mémoriser. Il convient de noter qu'un modèle n'est pas explicite et qu'il nécessite des informations supplémentaires, telles qu'une légende, des exemples, des scénarios, etc.

- L'accent mis sur un seul aspect réduit la charge cognitive nécessaire pour comprendre les exigences modélisées.

  Étant donné qu'un modèle a un objectif spécifique et une quantité réduite d'informations, la compréhension de la réalité modélisée peut nécessiter moins d'efforts.

- Les langages de modélisation des exigences ont une syntaxe restreinte qui réduit les ambiguïtés et les omissions possibles.

  Le langage de modélisation (syntaxe et sémantique) étant plus simple (nombre limité d'éléments de notation et règles linguistiques plus strictes par rapport au langage naturel), le risque de confusion et d'omission est moindre.

- Potentiel plus élevé pour l'analyse et le traitement automatisés des exigences.

  Parce qu'un langage de modélisation est plus formel (nombre limité d'éléments de notation et règles de langage plus strictes) qu'un langage naturel, il se prête mieux à l'automatisation de l'analyse ou du traitement des exigences.

Malgré les avantages considérables que présente la visualisation des exigences à l'aide de modèles, ces derniers ont également leurs limites.

- Il est difficile de maintenir la cohérence entre des modèles qui se concentrent sur différents aspects.

  Si plusieurs modèles sont utilisés pour décrire les exigences, il est important que ces modèles soient cohérents entre eux. Cela demande beaucoup de discipline et de coordination entre les modèles.

- Les informations provenant de différents modèles doivent être intégrées pour une compréhension causale.

  Si plusieurs modèles sont utilisés, tous les modèles doivent être compris pour permettre une bonne compréhension des exigences.

- Les modèles se concentrent principalement sur les exigences fonctionnelles.

  Les modèles permettant de décrire les exigences et les contraintes en matière de qualité sont limités, voire inexistants dans un contexte spécifique. Ces types d'exigences doivent ensuite être fournis en langage naturel avec les modèles, par exemple sous la forme d'un produit d'activités distinct.

- La syntaxe restreinte d'un langage de modélisation graphique implique que toutes les informations pertinentes ne peuvent pas être exprimées dans un modèle.

  Étant donné qu'un modèle est conçu dans un but et un contexte spécifiques, il n'est pas toujours possible d'enregistrer toutes les exigences dans le modèle ou dans plusieurs modèles. Les exigences qui ne peuvent pas être exprimées dans des modèles sont ajoutées au modèle en tant qu'exigences en langage naturel ou en tant que produit d'activité séparé.

Par conséquent, les modèles d'exigences doivent toujours être accompagnés d'un langage naturel [Davi1995].

3.4.1.4 Application d'un modèle pour les exigences

Comme indiqué dans les sections précédentes, il existe des modèles communs pour différents contextes. Par exemple, en architecture, il existe des plans de construction, des schémas de tuyauterie, des schémas électriques, etc. pour exprimer les spécifications d'un bâtiment. Dans d'autres contextes – par exemple, le développement de logiciels – il existe des langages de modélisation qui sont utiles dans ce type de contexte. Un aspect important de l'application des modèles est l'utilisation de modèles qui sont courants dans le contexte ou qui ont été spécialement développés pour un contexte spécifique.

De nombreux langages de modélisation, par exemple UML [OMG2017] ou BPMN [OMG2013], ont été normalisés. Lorsque les exigences sont spécifiées dans un langage de modélisation non normalisé, la syntaxe et la sémantique du langage doivent être expliquées au lecteur, par exemple à l'aide d'une légende.

Les modèles sont utilisés pour décrire les exigences d'un certain point de vue. Dans le cadre du développement d'un système, les exigences fonctionnelles sont classées selon les perspectives suivantes (voir également la section 3.1.4) :

- Structure et données

  Modèles axés sur les propriétés structurelles statiques d'un système ou d'un domaine

- Fonction et flux

  Les modèles qui se concentrent sur la séquence d'actions requises pour produire les résultats requis à partir d'intrants donnés ou sur les actions nécessaires pour exécuter un processus (métier), y compris le flux de contrôle et de données entre les actions et qui est responsable de quelle action

- État et comportement

  Les modèles qui se concentrent sur le comportement d'un système ou le cycle de vie des objets métier en termes de réactions, dépendantes de l'état, à des événements ou à la dynamique de l'interaction des composants

La nature du système en cours de modification ou de construction oriente les modèles à utiliser. Par exemple, si la nature du système est de traiter des informations et des relations, on s'attend à ce qu'il y ait un grand nombre d'exigences fonctionnelles qui décrivent ces informations et ces relations. Par conséquent, nous utilisons un langage de modélisation adapté qui se prête à la modélisation des données et de leur structure.

Naturellement, un système sera constitué d'une combinaison des perspectives susmentionnées. Il s'ensuit qu'un système doit être modélisé sous plusieurs angles. Les sections 3.4.3 à 3.4.5 décrivent plus en détail les différents modèles pour chaque perspective.

Avant que les exigences ne soient formulées et documentées – par exemple à l'aide de modèles – un inventaire des objectifs et du contexte est dressé. Ceux-ci peuvent également être modélisés, voir les sections 3.4.6 et 3.4.2.

L'application de modèles nous aide principalement de la manière suivante :

- *Spécifier* les exigences (principalement fonctionnelles) en partie ou même complètement, comme moyen de remplacer les exigences représentées textuellement
- *Décomposer* une réalité complexe en aspects bien définis et complémentaires, chaque aspect étant représenté par un modèle spécifique, nous aidant à saisir la complexité de la réalité
- *Paraphraser* des exigences représentées textuellement afin de les rendre plus faciles à comprendre, notamment en ce qui concerne les relations entre elles

- *Valider* des exigences représentées textuellement dans le but de découvrir des omissions, ambiguïtés et incohérences

La modélisation des besoins permet également de structurer et d'analyser les connaissances. Vous pouvez utiliser des diagrammes pour structurer vos propres réflexions afin de mieux comprendre le système et son contexte.

3.4.1.5 Facettes de la qualité d'un modèle d'exigences

Il s'agit d'une section supplémentaire pour laquelle il n'y aura pas de questions dans l'examen de niveau Fondation du CPRE.

Une grande partie des modèles d'exigences sont des diagrammes ou des représentations graphiques. La qualité du modèle d'exigences est déterminée par la qualité des différents diagrammes et de leurs relations mutuelles. À son tour, la qualité des diagrammes individuels est déterminée par la qualité des éléments de modèle dans les diagrammes.

La qualité des modèles d'exigences et des éléments de modèle peut être évaluée sur la base de trois critères : [LiSS1994]:

- Qualité syntaxique
- Qualité sémantique
- Qualité pragmatique

La qualité syntaxique exprime la mesure dans laquelle un élément de modèle unique (graphique ou textuel), un diagramme d'exigences ou un modèle d'exigences est conforme aux spécifications syntaxiques. Si, par exemple, un modèle qui décrit les exigences sous la forme d'un modèle de classe contient des éléments de modélisation qui ne font pas partie de la syntaxe, ou si des éléments de modélisation sont mal utilisés, la qualité syntaxique du modèle s'en trouvera diminuée. Une partie prenante de ce modèle – par exemple, un testeur – peut mal interpréter les informations représentées par le modèle. Cela pourrait éventuellement conduire à des cas de test inappropriés.

Les outils de modélisation des exigences permettent de vérifier la qualité syntaxique des modèles.

La qualité sémantique exprime la mesure dans laquelle un élément de modèle unique (graphique ou textuel), le diagramme des exigences ou le modèle des exigences représente correctement et complètement les faits.

Comme dans le langage naturel, la sémantique donne un sens aux mots. Si un terme peut avoir différentes significations ou si plusieurs termes ont la même signification, cela peut conduire à une mauvaise communication. Il en va de même pour la sémantique des éléments de modélisation. Si les éléments de modélisation sont mal interprétés ou mal appliqués, le modèle peut être mal interprété.

La qualité pragmatique exprime la mesure dans laquelle un élément de modèle unique (graphique ou textuel), le diagramme des exigences ou le modèle des exigences est adapté à l'utilisation prévue, c'est-à-dire si le degré de détail et le niveau d'abstraction sont

appropriés à l'utilisation prévue et si le modèle approprié est sélectionné en fonction du domaine ou du contexte. Cela peut être évalué si l'objectif et les parties prenantes du diagramme sont connus. Des versions intermédiaires du modèle peuvent être soumises aux parties prenantes intéressées afin de valider l'adéquation des diagrammes à leur objectif.

Lors de la validation des exigences, la qualité des diagrammes de modélisation utilisés est évaluée afin de s'assurer que ces diagrammes correspondent à l'objectif et à l'utilité prévus.

3.4.1.6 Le meilleur des deux mondes

Comme expliqué dans la section précédente, les exigences exprimées sous forme textuelle ou visuelle/graphique (c'est-à-dire via des modèles d'exigences) présentent des avantages et des inconvénients. En utilisant à la fois des représentations textuelles et graphiques des exigences, nous pouvons exploiter la puissance et les avantages des deux formes de représentation.

La modification d'un modèle par des exigences textuelles donne plus de sens au modèle. Une autre combinaison utile est la possibilité de lier les exigences et les contraintes de qualité à un modèle ou à un élément de modélisation spécifique. Cela permet d'avoir une vision plus complète des besoins spécifiques.

L'utilisation de modèles peut également soutenir les exigences textuelles. L'ajout de modèles et d'images aux exigences textuelles soutient ces modèles pour une meilleure compréhension et une meilleure vue d'ensemble.

3.4.2 Modélisation du contexte du système

Le principe 4 du chapitre 2 introduit la notion que les exigences ne sont jamais isolées et que le contexte du système, tel que les systèmes existants, les processus et les utilisateurs, doit être pris en compte lors de la définition des exigences pour le système nouveau ou modifié. Le contexte du système est décrit à l'aide d'un modèle, ce qui permet d'avoir une meilleure vue d'ensemble et de mieux comprendre le contexte du système. Un simple contexte de système peut également être saisi, par exemple en langage naturel ou sous forme de tableau, mais ce n'est pas préférable.

Les modèles de contexte spécifient l'intégration structurelle d'un système dans son environnement, avec ses interactions avec les utilisateurs du système ainsi qu'avec d'autres systèmes nouveaux ou existants dans le contexte concerné. Un modèle de contexte n'est pas une description graphique des exigences, mais sert à révéler certaines des sources des exigences. Figure 3.4 fournit un exemple abstrait d'un système et de son environnement, avec ses interfaces avec les utilisateurs du système et ses interfaces avec d'autres systèmes. Les diagrammes de contexte permettent donc d'identifier les interfaces utilisateur et les interfaces système. Si le système interagit avec les utilisateurs, les interfaces utilisateurs doivent être spécifiées lors d'une étape ultérieure de l'IE.

Si le système interagit avec d'autres systèmes, les interfaces avec ces systèmes doivent être définies plus en détail dans une étape ultérieure. Les interfaces avec d'autres systèmes peuvent déjà exister ou doivent être développées ou modifiées.

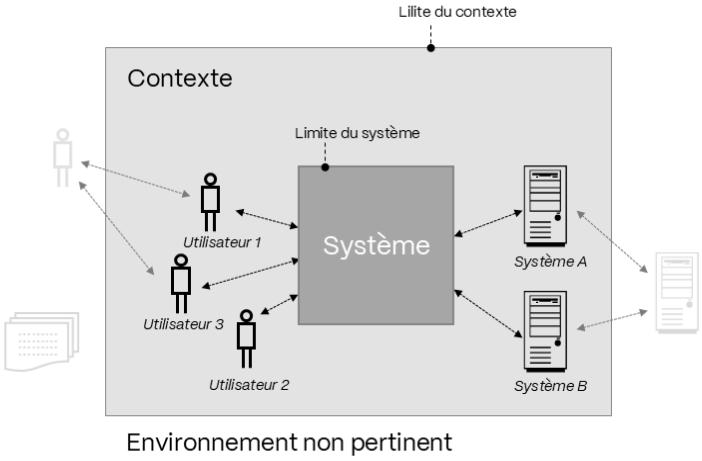


Figure 3.4 Un système dans son contexte

Même s'il n'existe pas de langage de modélisation normalisé pour les modèles de contexte, ces derniers sont souvent représentés par :

- Diagrammes de flux de données à partir d'une analyse structurée [DeMa1978]
- Diagrammes de cas d'utilisation UML Remarque : le modèle de cas d'utilisation UML se compose de deux éléments : le diagramme de cas d'utilisation UML (voir) et la spécification du cas d'utilisation (section [OMG2017]).
  Diagrammes de cas d'utilisation UML Remarque : le modèle de cas d'utilisation UML se compose de deux éléments : le diagramme de cas d'utilisation UML (voir Figure 3.6) et la spécification du cas d'utilisation (section 3.4.2.2). Ce chapitre se concentre sur la modélisation à l'aide des diagrammes de cas d'utilisation UML.
- Diagrammes en boîte et en ligne sur mesure [Glin2019]

Dans le domaine de l'ingénierie des systèmes, les diagrammes de définition de blocs SysML [OMG2018] peuvent être adaptés pour exprimer des modèles de contexte en utilisant des blocs stéréotypés pour le système et les acteurs.

Dans les deux sous-sections suivantes, nous présentons la notation des diagrammes de flux de données (DFD) et des diagrammes de cas d'utilisation UML pour modéliser le contexte d'un système. Ces deux exemples ne décrivent pas l'ensemble du contexte, mais le mettent en évidence d'un point de vue spécifique.

3.4.2.1 Diagramme de flux de données

Le contexte du système peut être envisagé sous différents angles. L'analyse structurée des systèmes [DeMa1978] parle du diagramme de contexte. Ce diagramme est un diagramme de flux de données (DFD) spécial dans lequel le système est représenté par un processus (le système). Figure 3.5 montre un exemple de diagramme de contexte.

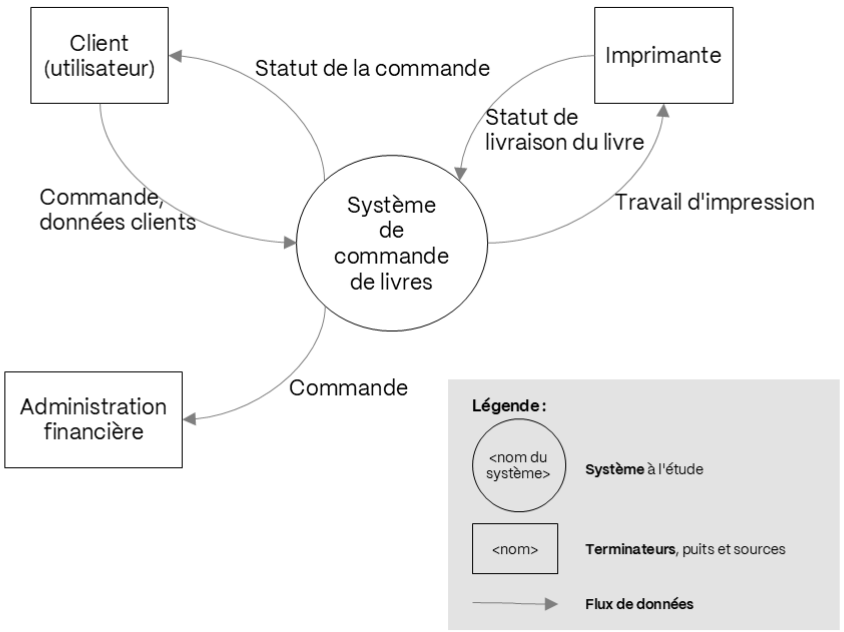


Figure 3.5 Exemple de diagramme de contexte utilisant un DFD

Le système est placé au centre du modèle. Il porte un nom clair afin que les lecteurs sachent de quel système il s'agit.

Les rectangles qui entourent le système sont des terminateurs : client, imprimeur et administration financière. Un terminateur qui fournit des informations ou des services au système est appelé une *source*. Un terminateur qui prend des informations ou des services du système est appelé un *puits*. Un terminateur peut jouer l'un ou l'autre rôle en fonction des données fournies ou extraites, comme le client dans l'exemple ci-dessus.

Les flèches de l'exemple montrent comment les informations provenant des terminateurs entrent dans le système (source) et comment elles sortent du système vers les terminateurs (puits). Les flèches reçoivent un nom logique qui décrit l'information transférée. Les détails non pertinents sont omis au niveau du diagramme de contexte. Le flux d'informations entre le client et le système contient, par exemple, des *données sur le client*. Les informations (nom, date de naissance, adresse électronique, numéro de téléphone, adresse de livraison, adresse de facturation, etc.) qui composent les *données du client* ne doivent pas encore être pertinentes à ce niveau d'abstraction.

Le flux d'informations peut être constitué d'objets matériels (matériels) et immatériels (informations). En outre, à ce niveau conceptuel, il n'y a pas (encore) de référence à la *manière dont* l'information est fournie *(courriel*, site web, formulaire, etc.).

L'ajout de détails supplémentaires au diagramme de contexte peut le rendre plus clair pour les parties prenantes concernées et contribuer à améliorer la compréhension commune. Ces détails doivent être réglés pour chaque situation individuelle.

L'utilisation d'un diagramme de flux de données pour modéliser le contexte d'un système permet de mieux comprendre les interactions du système avec son environnement, par exemple :

- Les interfaces avec les personnes, les services, les organisations et les autres systèmes de l'environnement
- Les objets (matériels et immatériels) que le système reçoit de l'environnement
- Les objets (matériels et immatériels) produits par le système et livrés à l'environnement

Un diagramme de flux de données indique une frontière claire entre le système et son environnement. Les utilisateurs et les systèmes pertinents de l'environnement sont identifiés lors de l'élucidation des besoins (section 4.1). Les diagrammes de contexte DFD peuvent aider à structurer le contexte afin de parvenir à une compréhension commune du contexte et des limites du système.

3.4.2.2 Diagramme de cas d'utilisation UML

Une autre vision du contexte d'un système peut être obtenue à partir d'une perspective fonctionnelle. Le diagramme de cas d'utilisation UML est une approche commune pour modéliser les aspects fonctionnels d'un système et les limites du système, ainsi que les interactions du système avec les utilisateurs et les autres systèmes. Les cas d'utilisation constituent un moyen simple de décrire systématiquement les différentes fonctions du champ d'application défini, du point de vue de l'utilisateur. Cela diffère des diagrammes de contexte DFD, dans lesquels le système est représenté comme une grande boîte noire.

Les cas d'utilisation ont été proposés pour la première fois comme méthode de documentation des fonctions d'un système dans [Jaco1992]. Les cas d'utilisation UML consistent en des diagrammes de cas d'utilisation auxquels sont associées des spécifications textuelles de cas d'utilisation (voir section 3.3.2). Une spécification de cas d'utilisation précise chaque cas d'utilisation en détail en décrivant, par exemple, les activités possibles du cas d'utilisation, sa logique de traitement, ainsi que les préconditions et les postconditions à l'exécution du cas d'utilisation. La spécification des cas d'utilisation est essentiellement textuelle, par exemple, par le biais de modèles de cas d'utilisation, comme le recommande le site [Cock2001].

Comme indiqué, un diagramme de cas d'utilisation UML présente les fonctions (cas d'utilisation) du point de vue des utilisateurs directs et des autres systèmes qui interagissent avec le système considéré. Le nom du cas d'utilisation est souvent composé d'un verbe et

d'un nom. Il s'agit d'une brève description de la fonction offerte par le système, comme le montre l'exemple de Figure 3.6.

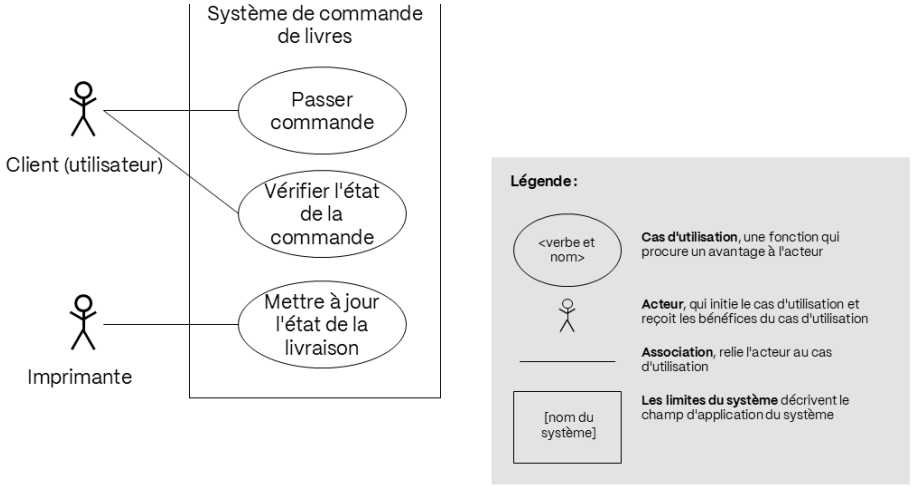


Figure 3.6 Exemple de diagramme de contexte utilisant un diagramme de cas d'utilisation UML

Les acteurs sont les utilisateurs directs ou les systèmes qui interagissent avec le système considéré. L'acteur (utilisateur ou système) qui lance le cas d'utilisation reçoit le bénéfice que le cas d'utilisation apporte (par exemple, montrer le statut d'une commande au client). L'association relie l'acteur au cas d'utilisation concerné, mais elle ne documente aucune direction ni aucun flux de données (comme c'est le cas dans les DFD) ; elle indique seulement que l'acteur reçoit le bénéfice du cas d'utilisation.

Un diagramme de cas d'utilisation UML décrit la fonctionnalité que le système offre à son environnement. La séparation entre la fonctionnalité du système et les acteurs du contexte est visualisée par la frontière du système (rectangle autour des cas d'utilisation, par exemple "système de commande de livres"). Les diagrammes de cas d'utilisation permettent de préciser les limites du système et de vérifier si le champ fonctionnel du système est couvert à un niveau élevé.

Chaque cas d'utilisation comprend également une spécification détaillée du cas d'utilisation, documentant les conditions préalables, le déclencheur, les actions, les conditions ultérieures, les acteurs, etc. Les cas d'utilisation sont généralement décrits à l'aide d'un modèle (section 3.3). Si les scénarios d'un cas d'utilisation deviennent complexes ou volumineux, il est recommandé de les visualiser à l'aide de diagrammes d'activité UML, voir la section 3.4.4.1. La spécification détaillée des cas d'utilisation ne fait pas partie de la modélisation du contexte et peut être élaborée ultérieurement, lorsque ces informations deviennent pertinentes.

3.4.3 Modélisation de la structure et des données

Pour les exigences fonctionnelles du point de vue des objets commerciaux (voir section 3.1.4), différents modèles de données sont disponibles. Un objet (métier) peut être un objet matériel ou immatériel, tel qu'un vélo, une pédale, une sonnette de vélo, mais aussi une

demande de formation, un panier d'achat contenant des produits numériques, etc. Un objet (métier) est "quelque chose" dans le monde réel. Certains (ou peut-être tous) de ces objets (métiers) sont utilisés par le système considéré. Le système utilise ces objets en tant qu'intrants pour les traiter, les conserver et/ou produire des résultats. Les modèles de données sont utilisés pour décrire les objets (métier) que le système doit connaître. Ces types de diagrammes modélisent l'objet, les attributs de l'objet et les relations entre les objets. Par souci de simplicité, nous parlons de modélisation de la structure et des données, qui représentent toutefois les structures d'information entre les objets (métier) dans le monde réel.

Il existe un certain nombre de modèles courants pour représenter la structure et les données :

- Diagramme des relations entre entités (ERD) [Chen1976]
- Diagrammes de classes UML [OMG2017]. Voir la section 3.4.3.1
- Diagrammes de définition des blocs SysML [OMG2018]. Voir la section 3.4.6.2

Pour expliquer le concept de modélisation de la structure et des données, ce chapitre utilise le diagramme de classes UML comme exemple. UML, abréviation de Unified Modeling Language (langage de modélisation unifié), consiste en un ensemble intégré de diagrammes. Cet ensemble de diagrammes est un recueil des meilleures pratiques d'ingénierie et a fait ses preuves dans la modélisation de systèmes complexes et de grande taille. UML a été conçu par Grady Booch, James Rumbaugh et Ivar Jacobson dans les années 1990 et constitue un langage de modélisation normalisé depuis 1997. Si vous souhaitez approfondir le sujet ou utiliser un modèle différent, lisez la documentation mentionnée et entraînez-vous avec le langage de modélisation souhaité.

3.4.3.1 Diagrammes de classes UML

UML est un ensemble de modèles différents qui peuvent être utilisés pour décrire un système. L'un de ces modèles est le diagramme de classes. Un diagramme de classes représente un ensemble de classes et les associations entre elles. Nous n'abordons ici que les éléments courants et simples de ce modèle. Si vous souhaitez approfondir la question, nous vous renvoyons à la littérature ou à la modélisation des exigences au niveau avancé du CPRE. Dans l'aperçu ci-dessous, vous trouverez les éléments de notation les plus courants.

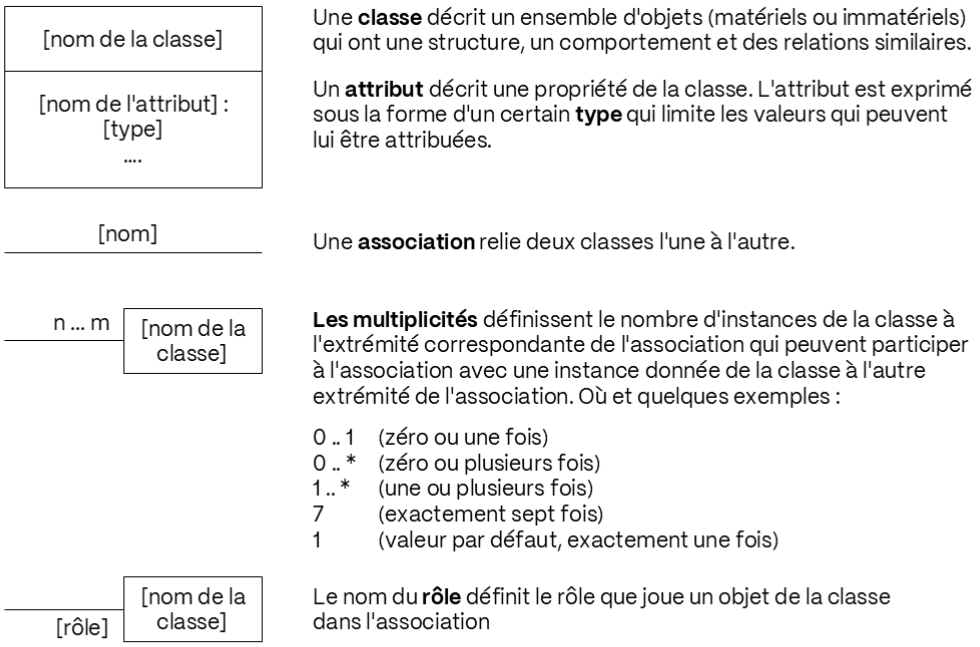


Figure 3.7 Sous-ensemble des éléments de modélisation des diagrammes de classes UML

Dans un modèle de classe, vous trouverez les concepts et les termes qui sont pertinents dans le domaine. Ces concepts font l'objet d'une définition claire qui figure dans le glossaire. Grâce à l'utilisation de modèles de données, le glossaire est complété par des informations sur la structure et la cohérence des termes et des concepts. Une définition claire et cohérente des termes utilisés permet d'éviter toute erreur de communication sur le sujet traité.

Figure 3.8 montre un modèle simplifié du système de commande de livres (voir les exemples du contexte à la section 3.4.2). Les informations statiques dont le système a besoin pour remplir sa fonction – commander un livre – sont modélisées.

Un client commande un livre et, par conséquent, les informations sont conservées pour les classes *Client*, *Commande* et *Livre*. Un client peut passer une commande et il existe donc une relation (association) entre le *client* et la *commande*. Un client peut passer plusieurs commandes au fil du temps et il ne devient client que s'il passe une commande. Cette information détermine la multiplicité : 1 client passe 1 ou plusieurs commandes.

Le fait qu'un client puisse commander un livre signifie qu'il existe également une relation entre les classes *Commande* et *Livre*. Pour que l'exemple reste simple, le client ne peut commander qu'un seul livre à la fois. En outre, une commande doit contenir au moins un livre. Une commande sans livre n'est pas une commande.

Dans la classe *Livre,* l'attribut *en stock* est également géré. Des informations telles que "si le stock n'est pas suffisant pour honorer la commande, un travail d'impression est envoyé à l'imprimante" ne peuvent pas être modélisées. Il s'agit d'un type d'information qui ne peut être modélisé dans un diagramme de classes car il décrit une certaine fonctionnalité du système. Ces informations font partie des exigences et doivent être documentées dans un

autre produit d'activité. Elle peut être ajoutée en tant qu'exigence textuelle accompagnant le diagramme de classes ou être modélisée à l'aide d'un autre diagramme, par exemple un diagramme d'activité UML (voir la section 3.4.4.1).

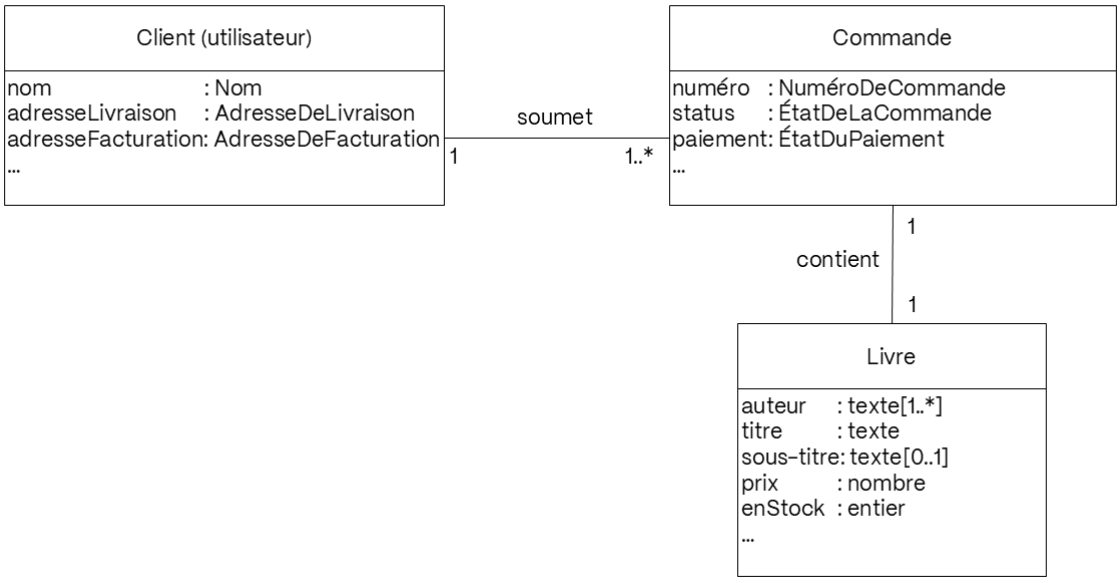


Figure 3.8 Exemple de diagramme de classes UML simple

3.4.4 Modélisation des Fonctions et des Flux

La fonction et le flux décrivent comment le (sous-)système transforme les données d'entrée en données de sortie. Nous pouvons visualiser ce type d'exigences à l'aide de modèles qui décrivent la fonction et le flux.

Contrairement à la modélisation des données, qui ne nécessite qu'un seul type de diagramme, la fonction et le flux peuvent être envisagés sous différents angles. Selon les besoins des parties prenantes pour passer à l'étape suivante du processus de développement, plusieurs modèles peuvent être nécessaires pour documenter les exigences relatives à la fonction et au flux.

Voici quelques modèles courants de représentation de la fonction et du flux :

- Diagramme de cas d'utilisation UML [OMG2017]. Voir la section 3.4.2.2
- Diagramme d'activité UML [OMG2017]. Voir la section 3.4.4.1
- Diagramme de flux de données [DeMa1978]. Voir la section 0
- Modèles de stories de domaine [HoSch2020]. Voir la section 3.4.6.3
- Modèle et notation des processus d'entreprise (BPMN) [OMG2013].

Remarque : Les modèles de processus BPMN sont utilisés pour décrire des processus métier ou des processus techniques. BPMN est fréquemment utilisé pour exprimer les modèles de processus métier.

Pour expliquer le concept de modélisation de la fonction et du flux, nous limitons cette section à quelques exemples de diagrammes UML. Si vous souhaitez approfondir le sujet ou utiliser un modèle différent, lisez la documentation mentionnée et entraînez-vous avec le langage de modélisation approprié.

3.4.4.1 Diagramme d'activité UML

Les modèles d'activité UML sont utilisés pour spécifier les fonctions du système. Ils fournissent des éléments pour la modélisation des actions et le flux de contrôle entre les actions. Les diagrammes d'activité peuvent également exprimer qui est responsable de quelle action. Des éléments de modélisation avancés (non couverts par ce handbook) fournissent les moyens de modéliser le flux de données.

Un diagramme d'activité UML exprime le flux de contrôle des activités d'un (sous-)système. La pensée par flux provient de la visualisation du code du programme à l'aide d'organigrammes (d'après [DIN66001], [ISO5807]). Cela a aidé les programmeurs à concevoir et à comprendre des structures et des flux complexes dans les programmes. Avec l'introduction d'UML [OMG2017], un modèle a été introduit pour visualiser les activités et les actions d'un point de vue fonctionnel.

Dans l'aperçu ci-dessous, vous trouverez les éléments de base de la notation.

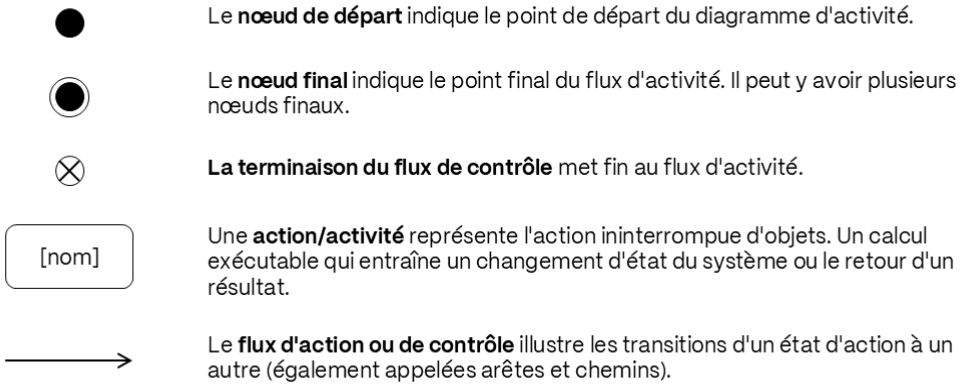


Figure 3.9 Éléments de notation de base du diagramme d'activité UML

Avec cet ensemble d'éléments de notation de base, vous pouvez créer un diagramme d'activités séquentiel simple. Si un contrôle plus poussé est nécessaire, le modèle peut être étendu avec des décisions et des flux parallèles d'activités en utilisant les éléments de notation ci-dessous.

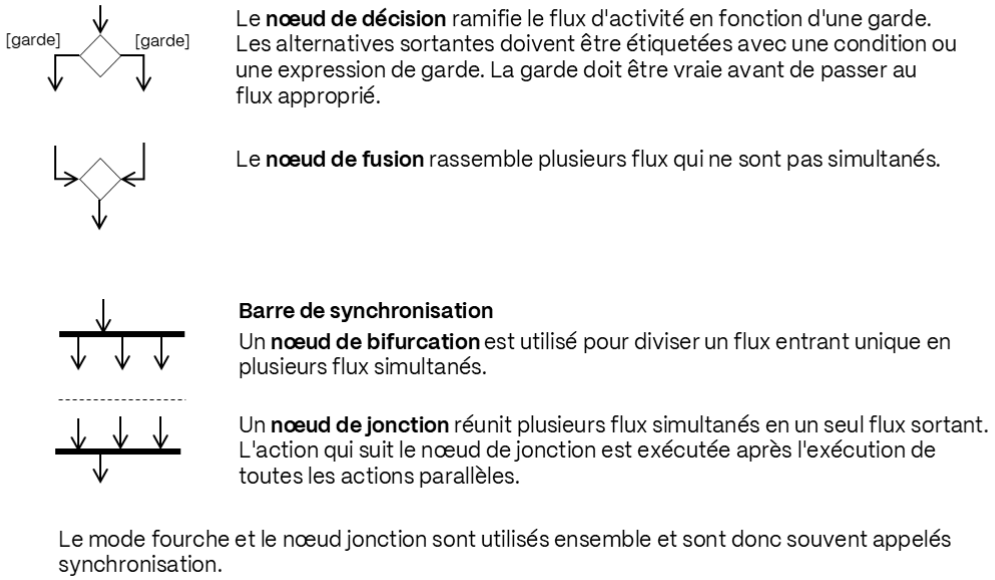


Figure 3.10 Décisions et flux parallèles dans un diagramme d'activité UML

Les diagrammes d'activités peuvent être utilisés pour spécifier en détail la logique de traitement des scénarios de cas d'utilisation (voir section 3.3.2). Des diagrammes d'activités sont créés pour visualiser les scénarios, qui sont des processus avec des activités et une logique de traitement. Tant que le diagramme reste compréhensible, le scénario principal peut être modélisé conjointement avec les scénarios alternatifs et les scénarios d'exception dans le cadre du même diagramme.

Figure 3.11 donne un exemple simple du système de commande de livres. Ce flux d'action simplifié commence lorsque le client envoie sa commande. Tout d'abord, la *commande* et les informations relatives au *client* sont validées afin de déterminer si toutes les informations (nécessaires) sont fournies. Si les informations relatives à la *commande* ou au *client* ne sont pas valides (incorrectes ou insuffisantes), une notification est envoyée au client et le processus de commande est annulé. Le scénario de base est que les informations relatives à la *commande* et au *client* sont valides. Le scénario selon lequel les informations relatives à la *commande* ou au *client* ne sont pas valides est appelé flux exceptionnel et traite une condition fonctionnelle défectueuse dans le processus.

Si les informations relatives à la *commande* et au *client* sont correctes, le stock est vérifié. Si le nombre de produits en stock est suffisant, la *commande* est préparée et envoyée au *client*. Un flux alternatif est lancé si le nombre de produits en stock est insuffisant. Une demande de travail d'impression est envoyée à l'*imprimante* et une notification de re-livraison est envoyée au *client*.

Figure 3.11 Exemple de diagramme d'activité UML

Le système de commande de livres comporte également d'autres flux séparés du processus de commande et de livraison. Par exemple, les processus de paiement, de re-livraison et de facturation ont des flux distincts pour permettre une séparation claire des préoccupations. Si, par exemple, la décision est prise de ne plus garder de produits en stock, le processus de commande et de livraison reste appliqué. Si des modifications sont nécessaires dans ce flux, elles peuvent ne pas affecter les autres flux. Cette décomposition de la fonctionnalité permet de garder les choses simples et claires.

3.4.5 Modélisation de l'Etat et du Comportement

Les exigences fonctionnelles qui décrivent le comportement, les états et les transitions d'un (sous-)système ou d'un objet métier sont des exigences dans la perspective comportementale. Un exemple d'état d'un système est *Marche*, *Attente* ou *Arrêt*. Un objet métier peut avoir un cycle de vie qui passe par un certain nombre d'états prescrits. Par exemple, un objet métier *Commande* peut se trouver dans les états suivants : *Passée*, *Validée*, *Payée*, *Expédiée* et *Terminée*.

Une technique largement utilisée pour décrire le comportement d'un système est celle des diagrammes d'état [Hare1988]. Les diagrammes d'état sont des machines d'état avec des états qui sont décomposés hiérarchiquement et/ou orthogonalement. Les machines d'état, y compris les diagrammes d'état, peuvent être exprimées dans le langage de modélisation UML [OMG2017] avec des diagrammes de machines d'état (également appelés diagrammes d'états).

Les diagrammes d'état décrivent des machines à états finis. Cela signifie que ces systèmes finissent par atteindre un état final. Un diagramme d'état montre les états que peut prendre le système ou un objet. Il indique également comment changer d'état, c'est-à-dire la transition d'état. Un système ne fait pas grand-chose par lui-même. Le changement d'état nécessite un déclencheur provenant du système ou de l'environnement du système.

Les modèles courants de représentation du comportement et des états sont les suivants :

- Diagrammes d'état [Hare1988]
- Diagramme d'états UML[OMG2017]

3.4.5.1 Diagramme d'états UML

Pour expliquer le concept de modélisation du comportement et des états, ce chapitre utilise le diagramme d'état UML comme exemple. Si vous souhaitez approfondir le sujet ou utiliser un modèle différent, lisez la documentation mentionnée et entraînez-vous avec le langage de modélisation approprié.

Dans l'aperçu ci-dessous, vous trouverez les éléments de base de la notation.

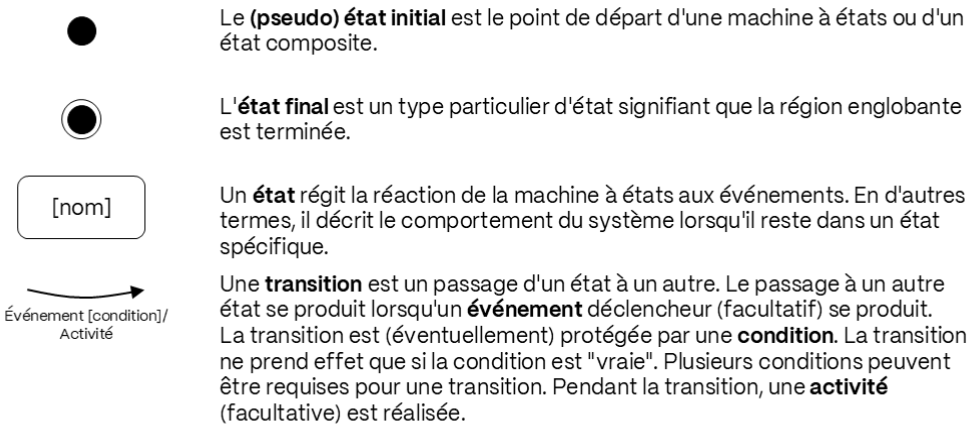


Figure 3.12 Éléments de notation de base du diagramme d'état UML

Comme nous l'avons vu au début de la section, un diagramme d'états peut clarifier les états que peut prendre un objet. Nous voyons ici une opportunité de visualiser des informations supplémentaires (et partiellement redondantes) d'un objet. Imaginez que vous commandiez un livre sur un site web et que vous souhaitiez suivre l'état de votre commande. Une commande est utilisée dans le monde réel et est modélisée comme un objet métier dans un diagramme de classes (voir Figure 3.8) avec, très probablement, un *statut d'*attribut. Le diagramme de classes indique que l'attribut *statut* peut prendre un nombre limité de valeurs, telles que Validé, Payé, Livré, Annulé, etc. Le diagramme de classes ne décrit pas l'ordre des changements d'état possibles. Un diagramme de classes ne décrit pas non plus le comportement du système dans un certain "état". Un diagramme d'état UML permet de

clarifier ce point – par exemple, une commande proposée ne peut pas passer directement à l'état *Livré* sans que le client n'ait payé la commande.

Figure 3.13 donne un exemple de diagramme d'état du système de commande de livres. Dans le diagramme de classes (Figure 3.8) du système de commande de livres, un objet *Commande* est modélisé. Cet objet possède un attribut *statut* qui peut avoir un nombre limité de valeurs. Ces valeurs sont énumérées et expliquées dans le diagramme de classe. Ce qu'un diagramme de classes ne décrit pas, c'est la séquence dans laquelle la commande est traitée. Un diagramme d'état visualise les états et les transitions entre les états, ce qui permet de comprendre la séquence de l'état de la commande. Le diagramme des états montre, par exemple, que la commande ne peut pas être envoyée avant d'avoir été entièrement préparée (transition entre les états *Préparée* et *Envoyée*). En outre, si la commande se trouve dans l'état *Envoyé,* l'état suivant ne peut être que *Payé*. Il n'est pas possible de passer de Envoyée, à *Traitée*. Ce diagramme montre aussi clairement que le paiement a lieu après l'envoi du livre. Vous pouvez demander aux parties prenantes si c'est ce dont elles ont besoin ou ce qu'elles ont demandé.

Une transition peut conduire au même statut. Cette situation est visible dans l'état *Préparée*. Chaque fois que la commande n'est pas prélevée jusqu'à son terme, elle reste dans le même état afin d'éviter l'envoi d'une commande incomplète. Ce n'est que lorsque la commande est entièrement préparée qu'elle est envoyée au client.

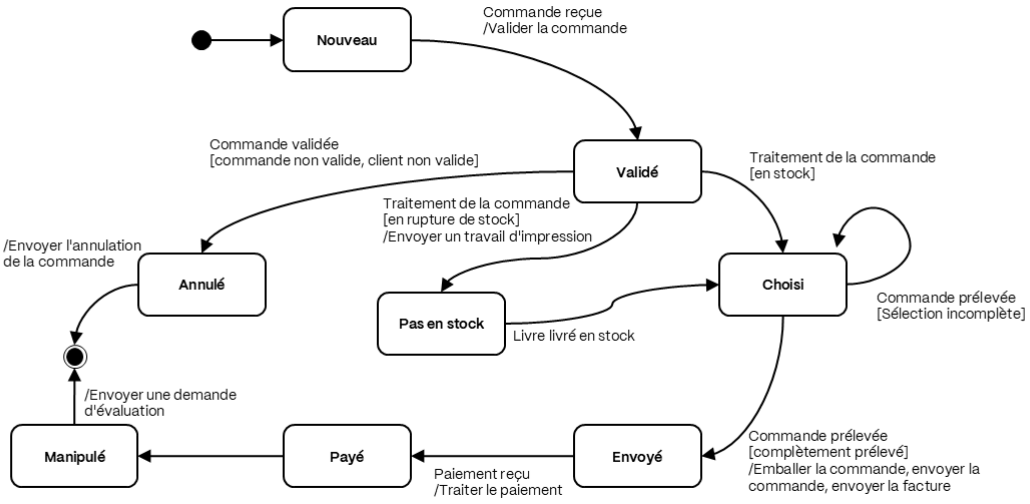


Figure 3.13 Exemple de diagramme d'état UML

Quelques mois après le lancement du système de commande de livres, des clients se sont plaints de ne pas avoir la possibilité d'annuler une commande. Il a été convenu qu'un client pouvait annuler sa commande à chaque étape du processus de commande. La modélisation de cette nouvelle exigence signifie qu'une transition vers *Annulée* est nécessaire pour chaque état. Cela peut rendre le diagramme difficile à lire. L'ajout d'une exigence textuelle pour décrire ce comportement pourrait être un moyen de garder le modèle simple pour le public.

3.4.6 Modèles supplémentaires

Au Niveau Fondation du CPRE, la compréhension et l'application des modèles sont limitées à certains types de modèles importants. D'autres types de modèles sont utilisés dans l'Ingénierie des Exigences. Les sous-sections suivantes fournissent des exemples supplémentaires de modèles qui sont complémentaires et ne seront pas examinés dans le cadre de l'examen de niveau CPRE Foundation.

3.4.6.1 Objectifs de la modélisation

Les exigences métier décrivent un objectif ou un besoin de l'entreprise. Ils décrivent le résultat final que la solution doit atteindre et avec lequel le problème (métier) est résolu, voir le chapitre 2, principe 5. Pour s'assurer que l'accent est mis sur la résolution du problème et que l'effort se concentre sur la valeur ajoutée, les objectifs sont soigneusement décrits. En ingénierie des exigences, il existe plusieurs façons de documenter les objectifs. La plus courante est l'utilisation du langage naturel (section 3.2) ou de modèles (section 3.3). Les formulaires de documentation basés sur des gabarits peuvent être trouvés, par exemple, dans [Pich2010], [Pohl2010], ou [RoRo2012].

Il existe également des notations basées sur des modèles pour documenter les objectifs. La notation la plus simple et la plus courante est un arbre ET/OU [AnPC1994] . Les arbres ET/OU nous permettent de documenter les objectifs à différents niveaux de détail et de relier les sous-objectifs aux objectifs à l'aide de relations ET et OU. Une relation ET signifie que tous les sous-objectifs doivent être remplis pour que l'objectif soit atteint. Une relation OU est utilisée pour exprimer qu'au moins un des sous-objectifs doit être rempli pour que l'objectif soit atteint.

Des approches de modélisation plus élaborées pour les objectifs peuvent être trouvées dans :

- Langage d'exigences orienté vers les objectifs (GRL) [GRL2020]

  Il s'agit d'un langage qui prend en charge la modélisation et le raisonnement orientés vers les objectifs des exigences, en particulier pour traiter les exigences non fonctionnelles.

- Acquisition de connaissances dans la spécification automatisée (KAOS) [vLam2009]

  KAOS est une méthodologie qui comprend la modélisation des objectifs. Cela permet aux analystes de construire des modèles d'exigences et de dériver des documents d'exigences à partir des modèles d'objectifs KAOS.

- Le framework i* est l'une des méthodes de modélisation et de raisonnement orientées vers les objectifs et les agents les plus populaires dans le domaine. i* permet de créer des modèles représentant une organisation ou un système socio-technique. [FLCC2016] fournit une vue d'ensemble du cadre i* et de ses applications.

La documentation des objectifs (sous forme textuelle ou graphique) est un point de départ important pour l'obtention des exigences, le renvoi des exigences à leur raison d'être et l'identification des sources – telles que les parties prenantes – des exigences, etc.

3.4.6.2 Diagrammes de définition des blocs SysML

Langage de modélisation des systèmes (SysML) [OMG2018] est un langage de modélisation à usage général pour les applications d'ingénierie des systèmes. SysML est un dialecte d'UML, qui réutilise et étend certaines parties d'UML.

SysML peut être adopté pour de nombreux objectifs différents. Les *diagrammes de définition de bloc* dans SysML sont une extension du diagramme de classe UML. Ils peuvent, par exemple, être adaptés pour représenter des diagrammes de contexte en utilisant des blocs stéréotypés pour le système et les acteurs. Les diagrammes de définition de blocs peuvent également être utilisés pour modéliser la structure d'un système en termes d'entités conceptuelles du système et de relations entre elles.

3.4.6.3 Modèles de stories de domaine

Les modèles de stories de domaine peuvent être utilisés pour modéliser la fonction et le flux, en spécifiant des stories visuelles sur la façon dont les acteurs interagissent avec des dispositifs, des artéfacts et d'autres éléments dans un domaine, en utilisant généralement des symboles spécifiques au domaine [HoSch2020] . Ils permettent de comprendre le domaine d'application dans lequel un système fonctionnera.

Les techniques sont conçues pour être exécutées très simplement pour raconter une story, un tableau et des notes adhésives peuvent suffire. Rassembler les parties prenantes qui connaissent réellement le fonctionnement de l'entreprise permet de susciter une discussion fructueuse en racontant des stories qui se produisent dans le domaine. Le récit de domaine améliore la compréhension commune d'un processus métier et est utilisé pour analyser et résoudre les problèmes dans le domaine.

3.4.6.4 Diagramme de séquence UML

Le diagramme de séquence UML est utilisé pour décrire l'interaction entre les partenaires de communication et pour modéliser l'aspect dynamique des systèmes. L'aspect dynamique des systèmes qu'un diagramme de séquence décrit peut être la fonction et le flux ainsi que l'état et le comportement. Par conséquent, un diagramme de séquence UML peut être utilisé à différentes fins.

Les partenaires de communication dans un diagramme de séquence UML sont des acteurs, des systèmes, des composants et/ou des objets au sein d'un système. L'interaction affiche la séquence de messages (un scénario) entre ces partenaires de communication.
L'interaction qui a lieu entre les partenaires de communication réalise l'objectif d'un scénario, respectivement (une partie) d'un cas d'utilisation.

Dans l'aperçu ci-dessous, vous trouverez les éléments de base de la notation.

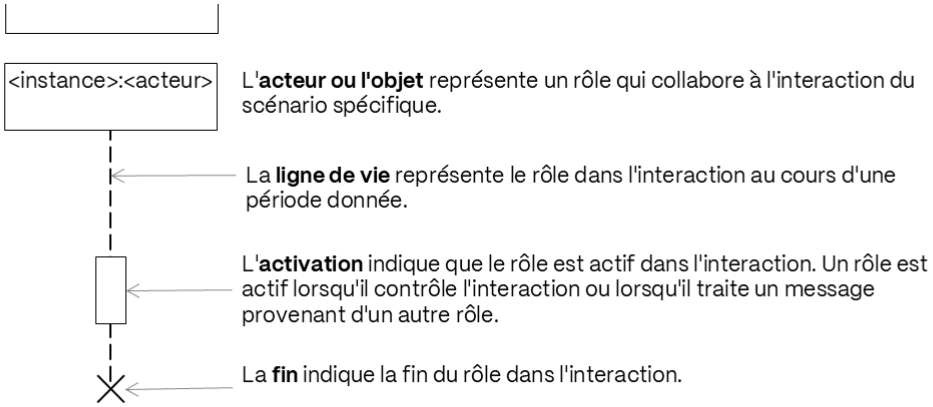


Figure 3.14 Éléments de notation de base du diagramme de séquence UML

Une ligne de vie dans un scénario représente le rôle dans le scénario, c'est-à-dire l'instance d'un acteur. Lors de la modélisation des diagrammes de séquence, le nom de l'instance d'un acteur ou d'un objet est souvent omis. Les rôles qui participent à la communication interagissent en envoyant des messages. Deux types de messages sont utilisés dans l'interaction.

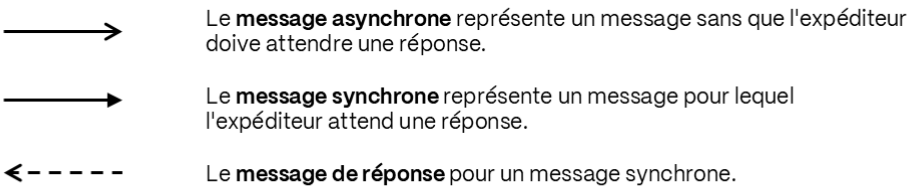


Figure 3.15 Éléments de notation de base de la messagerie dans le diagramme de séquence UML

Un message peut également être envoyé depuis ou vers des objets extérieurs au scénario. Elle est représentée par un cercle rempli. L'expéditeur ou le destinataire de ce type de messages peut être inconnu.

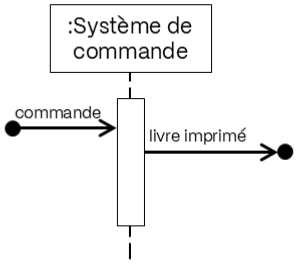


Figure 3.16 Messages en provenance et à destination d'un objet extérieur au scénario

Figure 3.17 montre un modèle du scénario dans lequel un *client* commande un livre et ce livre spécifique est en rupture de stock. Le *client* demande à passer une *commande*. Si la *commande* n'est pas valide, une notification indiquant que la *commande* est annulée est renvoyée. Si la *commande* est valide, le stock est vérifié et si un livre n'est pas en stock, un travail d'impression est envoyé à l'*imprimeur*.

Il s'agit d'un message synchrone car nous attendons de recevoir le livre – même si l'impression du livre peut prendre un certain temps. Une notification est envoyée au *client* pour l'informer que le livre est en rupture de stock et qu'il sera livré à nouveau. La *commande* est désactivée jusqu'à ce que le livre soit livré par l'*imprimeur*.

Lorsque le livre est reçu de l'*imprimeur,* le *système de commande* est à nouveau activé. La commande est préparée et envoyée au *client*. La *commande* est ainsi terminée et une dernière notification de l'état de la *commande* est envoyée au *client*.

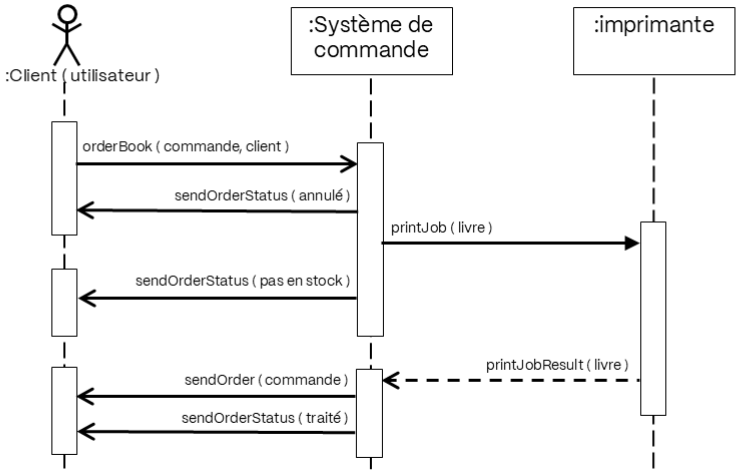


Figure 3.17 Exemple de diagramme de séquence UML

3.5 Glossaires

Les glossaires sont un moyen essentiel d'établir une compréhension commune de la terminologie utilisée lors du développement d'un système : ils permettent d'éviter que les personnes impliquées en tant que parties prenantes ou développeurs n'utilisent et n'interprètent les mêmes termes de différentes manières.

Un bon glossaire contient les définitions de tous les termes pertinents pour le système, qu'il s'agisse de termes spécifiques au contexte ou de termes courants utilisés avec une signification particulière dans le contexte du système à développer. Un glossaire doit également définir toutes les abréviations et tous les acronymes utilisés. S'il existe des synonymes (c'est-à-dire des termes différents désignant la même chose), ils doivent être indiqués comme tels. Les homonymes (c'est-à-dire les termes identiques qui désignent des choses différentes) doivent être évités ou au moins marqués comme tels dans le glossaire.

Quelques règles guident la création, l'utilisation et la maintenance du glossaire dans le cadre d'un projet de développement de système.

- *Création et maintenance*. Pour que la terminologie définie dans le glossaire soit cohérente et toujours à jour, il est essentiel que le glossaire soit géré et mis à jour de manière centralisée tout au long du projet, une personne ou un petit groupe étant responsable du glossaire. Lors de la définition des termes, il est important que les parties prenantes soient impliquées et acceptent la terminologie.
- *Utilisation*. Pour tirer pleinement parti d'un glossaire, son utilisation doit être obligatoire. Les produits d'activités doivent être vérifiés par rapport à une utilisation correcte du glossaire. Cela signifie évidemment que toutes les personnes impliquées dans un projet doivent avoir un accès en lecture au glossaire.

Lorsqu'une organisation développe des systèmes connexes dans le cadre de plusieurs projets, il est logique de créer un glossaire au niveau de l'entreprise afin d'obtenir une terminologie cohérente entre les projets.

La création, la mise à jour et l'utilisation d'un glossaire permettent d'éviter les erreurs et les malentendus concernant la terminologie utilisée. L'utilisation de glossaires est une pratique courante dans l'IE.

3.6 Documents et Structures de Documentation des Exigences

Il ne suffit pas de travailler avec les exigences au niveau des exigences individuelles. Les exigences doivent être rassemblées et regroupées dans des produits d'activités appropriés, qu'il s'agisse de documents d'exigences explicites ou d'autres structures documentaires liées à l'IE (comme un carnet de commandes).

Les modèles de documents (voir section 3.3.3) peuvent être utilisés pour organiser ces documents selon une structure bien définie afin de créer un ensemble d'exigences cohérent et maintenable. Des modèles sont disponibles dans la littérature [Vole2026], [RoRo2012] et

dans les normes [ISO29148]. Les modèles peuvent également être réutilisés à partir de projets antérieurs similaires ou peuvent être imposés par un client. Une organisation peut également décider de créer un modèle de document en tant que norme interne.

Un document d'exigences peut également contenir des informations et des explications supplémentaires, par exemple un glossaire, des critères d'acceptation, des informations sur le projet ou les caractéristiques de la mise en œuvre technique.

Les documents d'exigences fréquemment utilisés sont :

- *Spécification des exigences des parties prenantes*: les désirs et les besoins des parties prenantes qui doivent être satisfaits par la construction d'un système, du point de vue des parties prenantes. Lorsqu'un client rédige une spécification des exigences des parties prenantes, il s'agit d'une *spécification des exigences du client*.
- *Spécification des exigences des utilisateurs* : sous-ensemble d'une spécification des exigences des parties prenantes, couvrant uniquement les exigences des parties prenantes qui sont des utilisateurs potentiels d'un système.
- *Spécification des exigences du système*: les exigences relatives à un système à construire et à son contexte afin qu'il satisfasse les désirs et les besoins de ses parties prenantes.
- *Spécification des exigences métier*: les buts, objectifs et besoins métiers d'une organisation qui doivent être atteints par l'utilisation d'un système (ou d'un ensemble de systèmes).
- *Document de vision*: imagination conceptuelle d'un système futur, décrivant ses principales caractéristiques et la manière dont il créera de la valeur pour ses utilisateurs.

Des structures de documentation alternatives fréquemment utilisées sont :

- *Backlog du produit*: liste des éléments de travail classés par ordre de priorité, couvrant toutes les exigences nécessaires et connues pour le produit
- *Backlog de sprint*: un sous-ensemble sélectionné d'un backlog de produit avec des éléments de travail qui seront réalisés au cours de la prochaine itération
- *Story map*: organisation visuelle en deux dimensions des user stories dans un backlog de produit, en termes de temps et de contenu

Il n'existe pas de document d'exigences ou de structure de documentation standard ou universel. En conséquence, les documents ou les structures documentaires ne doivent pas être réutilisés sans réflexion à partir de projets antérieurs.

Le choix effectif dépend de plusieurs facteurs, par exemple :

- Du processus de développement choisi
- Du type de projet et le domaine (par exemple, solution sur mesure, développement de produits ou personnalisation de produits standard)
- Du contrat (un client peut prescrire l'utilisation d'une structure de documentation donnée)
- De la taille du document (plus le document est volumineux, plus il faut le structurer)l

3.7 Prototypes en ingénierie des exigences

Les prototypes jouent un rôle important dans l'ingénierie et la conception.

> Définition 3.5 Prototype:
> 1. In manufacturing: A piece which is built prior to the start of mass production.
> 2. In software and systems engineering: A preliminary, partial realization of certain characteristics of a system.
> 3. In design: A preliminary, partial instance of a design solution.

Dans le domaine de l'ingénierie des logiciels et des systèmes, les prototypes sont utilisés à trois fins principales : [LiSZ1994]:

Les *prototypes exploratoires* sont utilisés pour apporter une compréhension commune, clarifier des exigences ou valider des exigences à différents niveaux de fidélité. Ces prototypes constituent des produits d'activités temporaires qui sont mis au rebut après utilisation. Les prototypes exploratoires peuvent également être utilisés comme moyen de spécification par l'exemple. Ces prototypes doivent être traités comme des produits évolutifs ou durables.

Les *prototypes expérimentaux* (également appelés "breadboards") sont utilisés pour explorer les concepts de solutions de conception technique, en particulier en ce qui concerne leur faisabilité technique. Ils sont jetés après usage. Les prototypes expérimentaux ne sont pas utilisés dans l'IE.

Les *prototypes évolutifs* sont des systèmes pilotes qui forment le noyau d'un système à développer. Le système final évolue en étendant et en améliorant progressivement le système pilote en plusieurs itérations. Le développement de systèmes agiles utilise fréquemment une approche de prototypage évolutif.

Les ingénieurs des exigences utilisent principalement des prototypes exploratoires comme moyen d'élucidation et de validation des exigences. Dans le cadre de l'élucidation, les prototypes servent de moyen de spécification par l'exemple. En particulier, lorsque les parties prenantes ne peuvent pas exprimer clairement ce qu'elles veulent, un prototype peut leur montrer ce qu'elles obtiendraient, ce qui les aide à formuler leurs exigences. En matière de validation, les prototypes sont un moyen puissant de valider l'*adéquation* (voir section 3.8) des exigences.

Les prototypes exploratoires peuvent être construits et utilisés avec différents degrés de fidélité. Nous distinguons les wireframes, les maquettes et les prototypes natifs.

Les *wireframes* (également appelés prototypes papier) sont des prototypes basse-fidélité construits avec du papier ou des matériaux simples ou des outils d'esquisses qui servent

principalement à discuter et à valider des idées de conception et les concepts d'interface utilisateur. Lors du prototypage de systèmes numériques, les wireframes peuvent également être élaborés à l'aide d'outils d'esquisse numérique ou d'outils de wireframing spécifiques. Toutefois, lorsqu'on utilise un outil numérique pour le wireframing, il est important de conserver les propriétés essentielles d'un wireframe : il peut être construit rapidement, modifié facilement, et n'a pas l'air peaufiné ni ne ressemble à un produit final.

Les *maquettes* sont des prototypes moyenne-fidélité. Lors de la spécification de systèmes numériques ils utilisent des écrans et des flux de clics réels, mais sans réelle fonctionnalité. Elles servent principalement à spécifier et à valider les interfaces utilisateurs. Les maquettes donnent aux utilisateurs une expérience réaliste de la manière d'interagir avec un système par le biais de son interface utilisateur. Elles sont généralement construites avec des outils de prototypage dédiés.

Les *prototypes natifs* sont des prototypes haute-fidélité qui implémentent des parties critiques d'un système dans une mesure telle que les parties prenantes peuvent utiliser le prototype pour voir si la partie prototypée du système fonctionnera et se comportera comme prévu.

Ils servent à la fois à la spécification par l'exemple et à la validation approfondie des exigences critiques. Les prototypes natifs peuvent également être utilisés pour explorer et décider des *variantes d'* exigences pour un aspect donné – par exemple, deux manières différentes de soutenir un processus d'entreprise donné.

Selon le degré de fidélité, les prototypes exploratoires peuvent être un produit d'activités coûteux. Les ingénieurs des exigences doivent trouver un compromis entre le coût de la construction et de l'utilisation des prototypes et la valeur obtenue en termes de facilité d'obtention et de réduction du risque d'exigences inadéquates, voire erronées.

3.8 Critères de qualité pour les produits d'activités et les exigences

Il est évident que les ingénieurs des exigences doivent s'efforcer de rédiger de bonnes exigences qui répondent à des critères de qualité donnés. La littérature et les normes en matière d'IE fournissent un riche ensemble de critères de qualité. Cependant, il n'existe pas de consensus général sur les critères de qualité à appliquer aux exigences. L'ensemble des critères présentés dans cette sous-section vise à fournir une pratique éprouvée au niveau fondation.

L'IE moderne suit une approche des exigences axée sur la valeur (voir le principe 1 au chapitre 2). Par conséquent, le degré de satisfaction d'une exigence par rapport aux critères de qualité donnés correspond à la valeur créée par cette exigence. Cela a deux conséquences importantes :

- Les exigences ne doivent pas nécessairement respecter tous les critères de qualité.
- Certains critères de qualité sont plus importants que d'autres.

Nous faisons la distinction entre les critères de qualité pour les exigences individuelles et les critères de qualité pour les produits d'activités de l'IE, tels que les documents de l'IE ou les structures de documentation.

Pour les exigences individuelles, nous recommandons d'utiliser les critères de qualité suivants :

- *Adéquat* : l'exigence décrit les besoins réels et convenus des parties prenantes.
- *Nécessaire*: l'exigence fait partie du champ d'application du système concerné, ce qui signifie qu'elle contribuera à la réalisation d'au moins un objectif ou d'un besoin de la partie prenante.
- *Sans ambiguïté*: il existe une véritable compréhension commune de l'exigence, ce qui signifie que toutes les personnes concernées l'interprètent de la même manière.
- *Complète*: l'exigence est autonome, c'est-à-dire qu'il ne manque aucun élément nécessaire à sa compréhension.
- *Compréhensible*: l'exigence est compréhensible pour le public cible, ce qui signifie que le public cible peut entièrement comprendre l'exigence.
- *Vérifiable*: le respect de l'exigence par un système implémenté peut être vérifié de manière indiscutable (afin que les parties prenantes ou les clients puissent décider si une exigence est ou non respectée par le système implémenté).

Adéquation et compréhensibilité sont les critères de qualité les plus importants. Sans elles, une exigence est inutile, voire nuisible, indépendamment de la satisfaction de tous les autres critères. La vérifiabilité est importante lorsque le système implémenté doit faire l'objet d'une procédure d'acceptation formelle.

Certaines personnes utilisent le terme " *exactitude"* au lieu de " *adéquation*". Cependant, la notion d'exactitude implique qu'il existe une procédure formelle pour décider si quelque chose est correct ou non. Comme il n'existe pas de procédure formelle pour valider une exigence documentée par rapport aux souhaits et aux besoins que les parties prenantes ont à l'esprit, nous préférons le terme d'adéquation à celui d'exactitude.

Pour les produits d'activités couvrant plusieurs exigences, nous recommandons d'appliquer les critères de qualité suivants :

- *Cohérence*: aucune exigence, enregistrée dans un seul produit d'activités ou dans des produits d'activités différents, ne se contredit.
- *Non redondant* : chaque exigence n'est documentée qu'une seule fois et ne chevauche aucune autre exigence.
- *Complet*: le produit d'activités contient toutes les exigences pertinentes (exigences fonctionnelles, exigences de qualité et contraintes) qui sont connues à ce stade et qui sont liées à ce produit d'activités.
- *Modifiable*: le produit d'activités est conçu de telle sorte qu'il peut être modifié sans que sa qualité en soit affectée.
- *Traçable*: les exigences contenues dans le produit d'activités peuvent être retracées jusqu'à leur origine, jusqu'à leur mise en œuvre (dans la conception, le code et les tests) et jusqu'aux autres exigences dont elles dépendent.

- *Conforme*: s'il existe des instructions obligatoires de structuration ou de formatage, le produit d'activités doit s'y conformer.

3.9 Pour en savoir plus

Mavin et al. [MWHN2009] présentent et décrivent le modèle EARS. Robertson et Robertson [RoRo2012] décrivent les modèles Volere. Goetz et Rupp [GoRu2003], [Rupp2020] examinent les règles et les pièges liés à la rédaction d'exigences en langage naturel. Cockburn [Cock2001] a écrit un livre entier sur la façon de rédiger des cas d'utilisation. Lauesen [Laue2002] aborde la question des descriptions de tâches et donne quelques exemples de produits d'activités de l'IE dans le monde réel.

La norme ISO/IEC/IEEE 29148 [ISO29148] fournit de nombreuses ressources concernant les produits d'activités de l'IE : des gabarits de phrases, des critères de qualité pour les exigences et des descriptions détaillées du contenu des différents produits d'activités de l'IE, y compris un gabarit de document pour chaque produit d'activités. Cohn [Cohn2010] consacre un chapitre à la manière de formuler les exigences dans un carnet de commandes.

Gregory [Greg2016] et Glinz [Glin2016] examinent le problème de la spécification des exigences détaillées et de la possibilité de spécifier des exigences complètes et non ambiguës.

De nombreuses publications traitent de l'utilisation de modèles pour spécifier les exigences. La spécification UML [OMG2017], ainsi que les manuels sur UML, décrivent les modèles disponibles dans UML. Hofer et Schwentner [HoSch2020] introduisent la modélisation de domaine avec la narration de domaine. [OMG2013] et [OMG2018] décrivent respectivement les langages de modélisation BPMN pour la modélisation des processus métier et SysML pour la modélisation des systèmes. Les livres de Booch, Rumbaugh et Jacobson [BoRJ2005], [JaSB2011], [RuJB2004] donnent plus de détails et d'applications (pratiques) d'UML. En outre, les livres et articles suivants sont recommandés pour une connaissance plus approfondie et des modèles de modélisation des exigences : [DaTW2012], [Davi1993], [Fowl1996], [GHJV1994]. [LiSS1994] et [Pohl2010] permettent de mieux comprendre les aspects qualitatifs des modèles.

4 Pratiques pour l'élaboration des exigences

Dans les chapitres précédents, nous avons découvert la nature des exigences en tant que représentation des souhaits et des besoins des personnes et des organisations pour quelque chose de nouveau (par exemple, un système à développer ou à adapter), les principes qui sous-tendent la production des exigences et les moyens de capturer les exigences dans la documentation. Nous établissons des exigences avant de construire ou de modifier un système (ou une partie d'un système) afin de nous assurer que le système est utile et accepté par les personnes ou l'organisation qui l'ont demandé. Ces exigences servent ensuite de base à l'équipe de développement qui construira et mettra en œuvre le système.

C'est l'ingénierie des exigences en un mot ; elle intervient, explicitement ou souvent implicitement, chaque fois que l'on essaie de développer quelque chose. En principe, la qualité des exigences détermine la qualité du résultat du développement ultérieur. Sans exigences appropriées, il est peu probable que le système résultant soit utile. Il est donc important d'élaborer les exigences de manière professionnelle. Cela nécessite une définition explicite du " *comment faire*": les pratiques à utiliser pour une élaboration de qualité.

C'est l'objet de ce chapitre : il donne une vue d'ensemble des tâches, des activités et des pratiques qui sont pertinentes pour toute personne impliquée dans l'ingénierie des exigences. Elle commence par la recherche de sources potentielles d'exigences et se termine par la fourniture d'un ensemble d'exigences unique, cohérent, compréhensible et convenu, qui peut servir d'entrée au développement, à la maintenance et au fonctionnement efficaces d'un système performant.

La première tâche de l'ingénierie des exigences consiste à identifier et à analyser les *sources* potentielles *d'exigences*. Cette tâche peut sembler simple et évidente, mais comme nous le verrons dans la section 4.1, il y a plusieurs aspects qui doivent être pris en compte et analysés. Le fait de négliger une source conduira inévitablement à des exigences médiocres, voire manquantes, et dégradera donc la qualité du système qui en résultera.

L'étape suivante consiste à obtenir les exigences à partir de ces sources. C'est comme tirer de l'eau d'un puits : on ne sait jamais ce qu'il y a dans le seau tant qu'on ne l'a pas remonté à la surface. En ingénierie des exigences, cette tâche s'appelle l'*élucidation*; elle est expliquée à la section 4.2. Lors de l'élucidation, nous transformons les désirs, souhaits, besoins, demandes, attentes et autres éléments implicites en exigences explicites qui peuvent être reconnues et comprises par toutes les parties concernées.

Cependant, si vous demandez à deux personnes quels sont leurs besoins pour un certain système, vous obtiendrez rarement les mêmes réponses. Dans une série complète d'exigences élucidées issues de différentes sources, il est presque certain que certaines d'entre elles seront contradictoires. Comme il est impossible de mettre en œuvre des exigences contradictoires dans un seul et même système, la *résolution des conflits* sera toujours une tâche importante de l'ingénierie des exigences, comme décrit dans la section 4.3.

La section 4.4 est consacrée à la dernière tâche de l'ingénierie des exigences : la *validation des exigences*. L'objectif de cette étape est de s'assurer que la qualité de l'ensemble des exigences élucidées et des exigences individuelles au sein de cet ensemble est suffisante pour permettre le développement ultérieur du système.

La description ci-dessus des tâches d'ingénierie des exigences donne l'impression qu'elles sont exécutées comme un processus linéaire avec une séquence stricte d'étapes. Toutefois, ce n'est certainement pas l'intention de cette description et c'est rarement le cas dans la pratique.

Figure 4.1 montre quelques étapes du processus qui sont communes dans l'ingénierie des exigences. Ils peuvent être exécutés en parallèle, en boucle ou de manière séquentielle, selon ce qui convient le mieux à la situation donnée.

Le point de départ est souvent un ensemble limité de sources évidentes. Au cours de l'élucidation à partir de ces sources, de nouvelles sources sont identifiées, ce qui déclenche des tâches d'élucidation supplémentaires. En cas de conflit, il peut être nécessaire d'obtenir des informations plus détaillées pour trouver une solution. Lors de la validation, il peut apparaître qu'une source a été négligée, qu'une exigence est erronée ou qu'un conflit n'a pas été découvert, ce qui entraîne une nouvelle série d'activités d'analyse des sources, d'élucidation et/ou de résolution des conflits et de validation. Même au cours du développement ultérieur du système, les circonstances peuvent nécessiter une ingénierie des exigences supplémentaire.

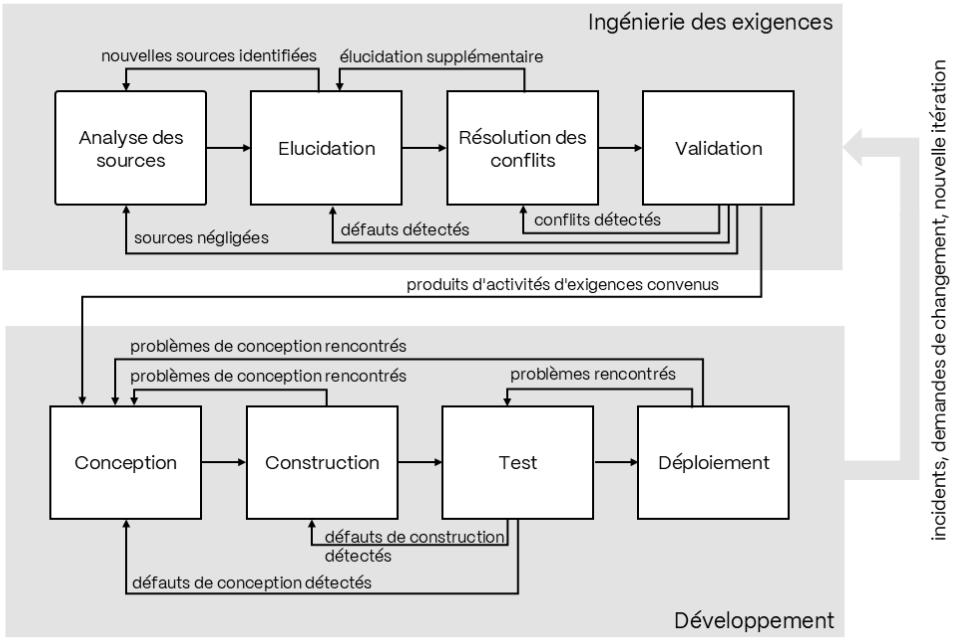


Figure 4.1 L'ingénierie des exigences n'est pas un processus linéaire

Dans les projets agiles, l'ingénierie des exigences itérative et incrémentale et le développement du système vont de pair, les exigences étant élaborées juste avant le développement d'un nouvel incrément du système. Dans ce type de projet, on constate souvent que le projet démarre avec un carnet de commandes limité, composé d'exigences de haut niveau qui ne sont affinées et détaillées que lorsqu'elles sont susceptibles d'être prises en compte dans l'itération suivante.

4.1 Sources des exigences

Les exigences ne sont pas comme des barres chocolatées, posées sur l'étagère pour que chacun les choisisse à sa guise. Dans l'introduction de ce chapitre, nous avons comparé les besoins à de l'eau à tirer d'un puits : c'est tout un effort que de les faire remonter à la surface. Par conséquent, le premier problème auquel est confronté l'ingénieur des exigences est le suivant : "Où sont les puits ?" Comme il n'y a pas d'exigence sans source, l'identification des sources potentielles d'exigences est l'une des premières activités de l'élucidation des exigences. Il ne suffit pas d'identifier ces sources uniquement au début d'un projet ou du développement d'un produit ; il s'agit d'un processus qui sera répété à plusieurs reprises.

Dès le début de l'élaboration des exigences, l'ingénieur des exigences doit s'engager à identifier, analyser et impliquer toutes les sources d'exigences pertinentes, car l'absence d'une source pertinente conduira inévitablement à une compréhension incomplète des exigences pertinentes. Et cela continuera jusqu'à la fin : l'identification des sources d'exigences est un processus qui nécessite une reconsidération constante.

Le principe 3 du chapitre 2 souligne la nécessité d'une compréhension commune (explicite et implicite) entre et parmi toutes les parties concernées. Comprendre le contexte du système à développer dans un certain domaine d'application est un prérequis pour pouvoir identifier les sources d'exigences pertinentes. La connaissance du domaine, une collaboration antérieure réussie, une culture et des valeurs communes et une confiance mutuelle sont des facteurs favorables à une compréhension commune, tandis que la distance géographique, l'externalisation ou les grandes équipes à forte rotation sont des obstacles.

Au chapitre 2, principe 4, nous avons présenté le contexte comme un concept essentiel pour comprendre et spécifier un système et ses exigences. Nous avons défini le contexte comme la partie de la réalité qui se situe entre la *limite du système* et la *limite du contexte*. Dans ce contexte, les entités influencent d'une manière ou d'une autre le système, voire interagissent avec lui, mais ne sont pas contenues dans le système lui-même.

La recherche de sources des exigences serait ainsi très simple : il suffirait de regarder dans le contexte ! Mais ce n'est pas si simple. Au début d'un processus de développement, le contexte n'a pas encore été défini ; les limites du système et du contexte doivent encore être déterminées. Par conséquent, la recherche de sources d'exigences est un processus itératif et récursif.

Les sources potentielles sont analysées en fonction de leur relation avec le futur système. Si vous ne trouvez aucune relation lors de l'analyse d'une source potentielle, cela signifie qu'elle

fait partie de l'environnement non pertinent et qu'elle ne sera pas analysée pour les exigences. Les sources potentielles qui semblent faire partie du futur système n'intéressent pas non plus l'ingénieur des exigences ; elles *appartiennent* aux développeurs. Seules les entités pour lesquelles l'analyse révèle une interaction, une interface ou une influence sur le futur système, mais qui restent (relativement) inchangées au cours du prochain développement, méritent d'être considérées comme des sources d'exigences. Dans cette analyse, les limites du système et du contexte sont délimitées, d'abord vagues, puis de plus en plus précises au fur et à mesure de l'identification des sources. À mesure que le contexte se précise, il devient plus facile d'identifier de nouvelles sources qui, à leur tour, permettent de mieux délimiter les frontières.

La recherche de sources d'exigences commence généralement par quelques sources évidentes, souvent identifiées par le client au début d'un effort de développement. L'élucidation initiale de ces sources permet de découvrir d'autres sources potentielles, qui sont ensuite analysées afin de déterminer si elles sont pertinentes pour le système. Au cours de cette analyse, de nouvelles sources potentielles peuvent apparaître. En fait, dans chaque effort d'élucidation, l'ingénieur des exigences restera attentif à la détection de nouvelles sources. Cela peut se poursuivre jusqu'à la fin de l'effort de développement. Cependant, nous essayons d'identifier rapidement les sources les plus importantes et les plus pertinentes, car toutes les autres activités d'ingénierie des exigences dépendent de cette identification précoce.

En ingénierie des exigences, nous distinguons trois grandes catégories de sources :

- Parties prenantes
- Documents
- (Autres) Systèmes

Ces catégories sont examinées plus en détail dans les sections suivantes.

4.1.1 Parties prenantes

Au chapitre 2, principe 2, vous avez appris que la partie prenante est une personne ou une organisation qui *influence les* exigences d'un système ou qui *est affectée* par ce système.

Les parties prenantes d'un système sont les principales sources d'exigences. Plus encore qu'avec d'autres sources, le fait de ne pas inclure une partie prenante pertinente aura un impact négatif majeur sur la qualité de l'ensemble final des exigences ; découvrir ces parties prenantes tardivement (ou les manquer complètement) peut conduire à des changements coûteux ou, en fin de compte, à un système inutilisable. Pour créer un système qui réponde aux besoins de toutes les parties prenantes, l'identification systématique des parties prenantes doit commencer dès le début de tout effort de développement et les résultats doivent être gérés tout au long du développement. Les parties prenantes se trouvent dans un large périmètre autour du système, allant des utilisateurs directs et indirects du système, des responsables (métier), du personnel informatique tel que les développeurs et les opérateurs, aux opposants et aux concurrents, aux institutions gouvernementales et réglementaires, et à bien d'autres encore. La question principale pour identifier une personne

ou une organisation en tant que partie prenante est la suivante : "Existe-t-il une relation pertinente entre la personne/l'organisation et le système ?"

Il est utile de considérer les parties prenantes comme des êtres humains faits de chair et de sang. Si vous identifiez une organisation comme partie prenante, posez-vous des questions telles que les suivantes : "Puis-je identifier une personne responsable de cette organisation ? Qui peut être considéré comme le premier contact de cette organisation ? Qui représente cette organisation au sein de notre entreprise ?" Par exemple, si le gouvernement est la partie prenante parce qu'une certaine loi est en jeu, il faut chercher quelqu'un qui représente le gouvernement comme la source à laquelle il faut s'adresser pour obtenir des informations. Dans ce cas, il n'est pas très utile d'identifier le Premier ministre comme cette personne ; le chef du service juridique interne serait un meilleur choix.

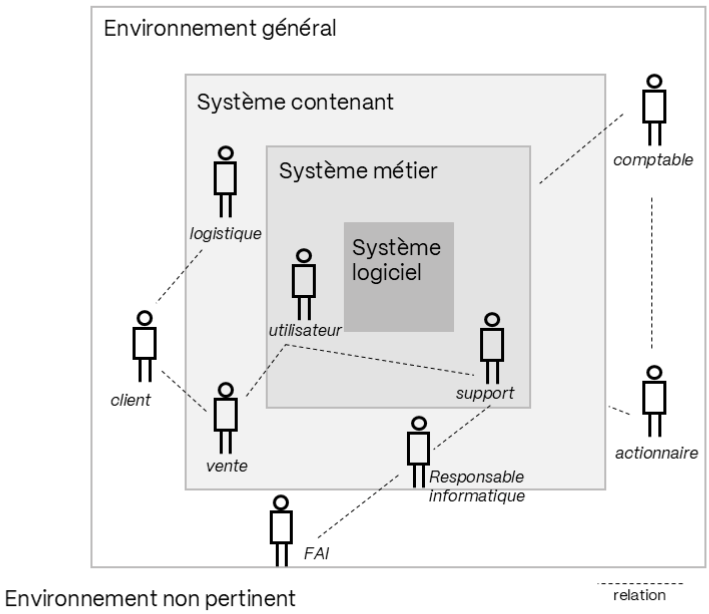


Figure 4.2 Le modèle de l'oignon d'Alexander

Il n'existe pas de technique standard pour identifier les parties prenantes, mais le modèle de l'oignon de Ian Alexander [Alex2005] peut constituer un bon point de départ, voir Figure 4.2. Ce modèle montre comment un système (logiciel) est entouré de trois couches de systèmes socio-techniques de niveau supérieur1 et de systèmes sociaux, chacun ayant ses propres parties prenantes :

1 Un système socio-technique est un système qui prend en compte des exigences couvrant les aspects matériels, logiciels, personnels et communautaires, tout en reconnaissant l'interaction entre les infrastructures complexes de la société et le comportement humain.

- Parties prenantes du système d'entreprise
- Les parties prenantes du système de contention
- Parties prenantes de l'environnement au sens large

Au début d'un effort de développement des exigences, quelques-unes de ces parties prenantes seront évidentes – souvent dans la première couche de l'oignon, par exemple, les utilisateurs finaux ou les gestionnaires d'applications. Ils peuvent servir de point de départ à la recherche d'autres parties prenantes. Après les avoir identifiées comme sources pertinentes, l'ingénieur des exigences analysera leurs relations, à la fois dans les couches internes et externes environnantes. Cette analyse permettra de découvrir de nouvelles parties prenantes qui, à leur tour, pourront avoir d'autres (et plus nombreuses) relations à analyser. On pourrait appeler cela le principe de la *boule de neige* [Good1961]: plus vous avez trouvé de parties prenantes, plus il est facile d'en trouver de nouvelles. Cependant, lorsqu'on arrive aux parties prenantes dans l'environnement élargi d'Alexander, toutes les relations extérieures se retrouvent dans l'environnement non pertinent, ce qui signifie qu'elles ne révèlent plus de nouvelles sources.

Outre le fait que les parties prenantes se réfèrent à d'autres parties prenantes, les documents peuvent souvent révéler de nouvelles parties prenantes. Les organigrammes, les descriptions de processus, les rapports de marketing et les documents réglementaires en sont de bons exemples. Pour plus d'informations sur la documentation en tant que source d'exigences, voir la section 4.1.2. Les checklists des groupes de parties prenantes typiques et de leurs rôles peuvent être un outil utile pour éviter d'oublier certaines parties prenantes potentielles peu visibles. L'analyse des parties prenantes des systèmes existants ou similaires peut également s'avérer utile.

En tant qu'ingénieur des exigences, vous collecterez de nombreuses données sur vos parties prenantes et les conserverez jusqu'à ce que votre travail soit terminé. Vous devez savoir qui sont les parties prenantes, comment vous pouvez les atteindre, quand et où elles sont disponibles, quelle est leur expertise, ainsi que leur pertinence en tant que source, quelle est leur attitude à l'égard du projet et leur influence sur celui-ci, quels sont leurs rôles dans l'entreprise, dans le projet et dans leur relation avec le système, etc.

En général, ces informations sont conservées dans une liste de parties prenantes, qui doit être tenue à jour, car à toutes les étapes, vous resterez en contact avec toutes les parties prenantes – certaines de manière intense et très étroite, d'autres de manière peu fréquente et superficielle. Voir Table 4.1 ci-dessous pour un exemple simplifié.

Table 4.1 Exemple de liste de parties prenantes

Nom	Service	Téléphone	Disponibilité	Rôle	Influence	Intérêt
Marlène	Propriétaire	482263	Uniquement le lundi	Parrainage	++	o
Pierre	Vente	481225	Permanent	Product Owner	++	+
Eva	Juridique	481237	Pas en juin	Consultant	+	-
Hassan	Logistique	242651	Permanent	Utilisateur	o	++
Mira	Service support	242424	Après 16 heures	Utilisateur	-	+
Natalia)*		481226	Permanent	Client	++	++

* Persona, créé, entretenu et représenté par l'équipe du *panel de clients*

Pour obtenir des informations pertinentes de la part des parties prenantes, il est essentiel d'entretenir une relation ouverte et de qualité avec elles. Cela repose principalement sur des caractéristiques comportementales telles que l'intégrité, l'honnêteté et le respect.

Une communication ouverte et fréquente sur vos plans, vos activités et vos résultats est essentielle. Il se peut que vous deviez faire passer les parties prenantes du statut d'opposants à celui de collaborateurs. En tant qu'ingénieur des exigences, vous devez comprendre ce que les parties prenantes attendent de vous. Vous devez également *vendre* votre travail en leur montrant les avantages de votre solution et en supprimant les obstacles que les parties prenantes rencontrent ou attendent sur le chemin de cette solution. Malheureusement, il n'est pas rare que certaines parties prenantes (principalement internes) prévoient ou subissent en fait des conséquences négatives des changements résultant de votre projet. Dans ce cas, il sera difficile d'obtenir leur coopération, même si vous en aurez certainement besoin. L'escalade vers les niveaux supérieurs de l'organisation peut alors être la seule façon de procéder, même si la relation qui en résultera sera loin d'être optimale. La gestion [Bour2009] des relations avec les parties prenantes est un moyen efficace de contrer les problèmes avec les parties prenantes.

Cela implique un processus continu d'obtention et de maintien du soutien et de l'engagement des parties prenantes en engageant les bonnes parties prenantes au bon moment et en comprenant et en gérant leurs attentes.

Afin d'impliquer les parties prenantes dans le processus d'élucidation, vous devez vous assurer qu'elles connaissent l'objet du projet et leur rôle au sein de celui-ci. Vous devez comprendre leurs besoins et essayer d'y répondre autant que possible dans le cadre de vos compétences dans le projet. Si les parties prenantes méritent d'être traitées avec respect, vous pouvez demander la même chose aux parties prenantes, du moins à celles qui sont activement engagées dans le projet. Cela signifie qu'elles doivent avoir du temps à vous

consacrer lorsque vous avez besoin d'elles. Elles doivent vous fournir les informations que vous demandez, ainsi que d'autres informations qu'ils jugent pertinentes. Leurs commentaires sur les produits de votre travail doivent être donnés en temps utile et elles doivent s'abstenir de tout commérage sur le projet, etc.

Les problèmes avec les parties prenantes surviennent généralement lorsque les droits et obligations de l'ingénieur des exigences et des parties prenantes en ce qui concerne le système proposé ou le projet en cours ne sont pas clairs ou lorsque les besoins respectifs ne sont pas suffisamment pris en compte. En cas de problème, une sorte d'*accord* ou de *contrat entre les parties prenantes* peut contribuer à clarifier les choses pour toutes les parties. Lorsque cela se produit au sein d'une organisation, l'approbation de la direction générale peut contribuer au succès d'une telle approche.

4.1.1.1 Une partie prenante particulière: L'utilisateur

Chaque système que nous développons aura une interaction avec certains utilisateurs ; sinon, pourquoi le développer ? Bien entendu, lorsque vous travaillez sur les exigences d'un sous-système technique intégré, caché dans une sorte de machine compliquée, les utilisateurs n'interagiront avec lui qu'indirectement, à travers plusieurs couches de systèmes environnants. Dans ce cas, ces utilisateurs ne seront pas vos principales sources d'exigences. Cependant, dans de nombreux systèmes, des êtres humains spécifiques auront une interface directe avec le système : les utilisateurs (finaux). Leur acceptation du système est vitale pour la réussite du projet. Ils sont donc au centre de vos préoccupations et feront l'objet d'une attention particulière pendant toute la phase d'ingénierie des exigences.

Il existe deux grandes catégories d'utilisateurs :

- Les *utilisateurs internes* sont directement liés à l'organisation pour laquelle le système est développé, comme le personnel, la direction, les sous-traitants. Ils sont pour la plupart en nombre limité, plus ou moins connus individuellement, et impliqués d'une manière ou d'une autre dans le projet. Il est relativement facile de les contacter et il est possible de les atteindre, de les influencer et de les motiver par le biais des canaux formels existants.
- Les *utilisateurs externes* sont en dehors de l'entreprise, tels que les clients (utilisateurs), les partenaires commerciaux, les civils. Leur nombre peut être (très) important, dans de nombreux cas ils ne sont pas connus individuellement, et ils peuvent être complètement ignorants ou indifférents au projet. Vous ne pouvez pas les influencer par les voies officielles. Pour entrer en contact avec eux, vous devrez peut-être prendre des mesures spéciales pour les inciter à participer, par exemple en faisant de la publicité, en promettant une récompense ou en leur donnant un accès gratuit à une version bêta. La constitution d'un panel d'utilisateurs ou le fait de s'adresser à la foule (parfois moyennant paiement) peuvent être des moyens utiles d'impliquer ces utilisateurs.

Sachez qu'en plus de ces catégories habituelles, il peut également être utile de prêter attention aux *utilisateurs abusifs*: les personnes qui tentent intentionnellement d'interagir

avec le système d'une manière préjudiciable, comme les pirates informatiques ou les concurrents. Il est rarement possible de les contacter ou de les influencer, mais vous pouvez essayer d'élaborer des mesures pour les décourager, les tenir à l'écart ou détecter les tentatives prévisibles d'utilisation abusive.

Cette catégorisation ne doit pas être considérée de manière très stricte. On peut imaginer des projets dans lesquels les utilisateurs extérieurs à l'entreprise sont peu nombreux et peuvent être atteints facilement, de sorte qu'ils peuvent être considérés comme *internes*. D'autre part, dans les grandes entreprises, la distance avec les utilisateurs peut être si grande qu'ils doivent être traités plus ou moins comme des *utilisateurs externes*.

Si vous avez une bonne connaissance de votre base d'utilisateurs, vous devriez faire une distinction entre les rôles des utilisateurs. Des rôles distincts auront généralement des besoins d'information différents, utiliseront le système à leur manière et pourront avoir des droits d'accès distincts aux fonctions et aux données – par exemple, les utilisateurs qui saisissent des données par rapport aux superviseurs qui vérifient ces saisies. Dans ce cas, veillez à inclure des représentants de tous les rôles concernés dans l'exercice d'élucidation.

Souvent, en particulier au début des efforts d'élucidation, ces informations font défaut. Ensuite, il est encore plus important de comprendre que l'*utilisateur* n'existe pas. L'*utilisateur* n'est pas une masse amorphe d'êtres humains identiques, mais plutôt un ensemble d'individus, chacun ayant ses propres habitudes, souhaits et besoins. Lorsqu'un système compte des milliers d'utilisateurs ou plus, il n'est évidemment pas possible d'adapter les exigences à leurs besoins individuels. D'un autre côté, une approche *unique* visant l'utilisateur *moyen* pourrait ne pas fonctionner non plus.

Dans ce cas, il est utile de discerner un certain nombre de types d'utilisateurs ou de groupes d'utilisateurs qui présentent certaines similitudes, souvent comportementales, au sein du groupe par rapport à d'autres groupes. Dans la pratique, il est souvent préférable d'avoir entre trois et sept groupes. Ensuite, en tant qu'ingénieur des exigences, vous traiterez chaque groupe comme une source distincte d'exigences. Une bonne technique consiste à utiliser des *personas* [Hump2017]. Les personas sont des individus fictifs qui décrivent des groupes d'utilisateurs typiques du système ayant des besoins, des objectifs, des comportements ou des attitudes similaires.

4.1.1.2 Personas

Il existe deux approches principales pour créer des personas :

- **Axé sur les données**

  Dans cette approche, les personas sont élaborés à l'aide de techniques de marketing, telles que les entretiens, les groupes de discussion et d'autres techniques de collecte de données ethnographiques. Ces personas sont appelés *personas qualitatifs*. Si les personas sont élaborés à partir d'une analyse statistique d'un large échantillon de votre base d'utilisateurs, ils sont appelés *personas quantitatifs*.

- **L'imagination**

  Comme alternative moins coûteuse et plus rapide, les personas peuvent être développés par l'imagination – par exemple, lors d'une session de brainstorming avec l'équipe du projet. Nous les appelons *personas ad hoc* ou *proto-personas*. En tant qu'ingénieur des exigences, vous devez être conscient que les personas ad hoc sont basés sur l'imagination et les hypothèses. Ces hypothèses doivent être vérifiées et confirmées tout au long du processus d'ingénierie des exigences.

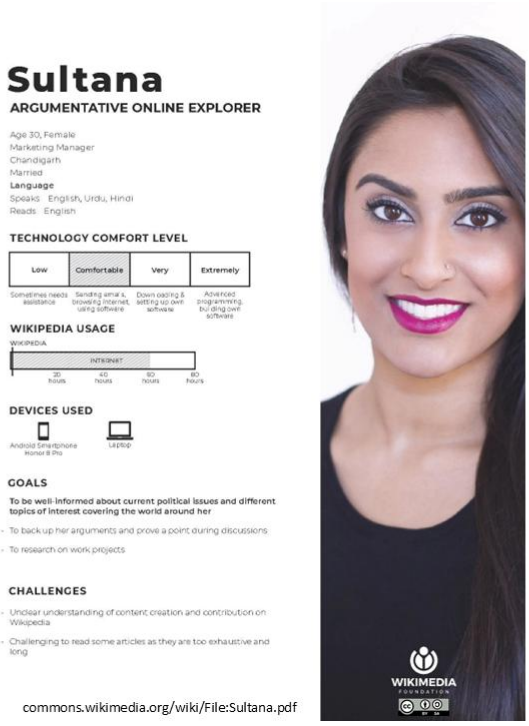


Figure 4.3 Exemple de Persona

En principe, les descriptions de persona contiennent des informations pertinentes pour le développement du système en question. En général, ces informations sont *enrichies* de données supplémentaires, telles que le nom, l'adresse, les loisirs et un dessin ou un portrait.

Cela donne un visage humain au concept abstrait de persona. Cela peut vous aider à comprendre que votre travail en tant qu'ingénieur des exigences ne concerne pas seulement les faits, mais aussi les émotions. Figure 4.3 donne un exemple de description de persona.

Si vous utilisez des personas dans votre projet, il peut être utile de rechercher quelques personnes correspondant aux descriptions des personas et de les traiter comme des représentants de chaque groupe. Dans ce cas, vous avez de véritables parties prenantes avec lesquelles vous pouvez communiquer. Cependant, n'oubliez jamais que le groupe que représente une telle partie prenante est un groupe artificiel qui n'existe pas dans le monde réel, mais uniquement dans le contexte du système ou du projet.

4.1.2 Documents

Les documents peuvent également constituer une source importante d'exigences. En tant qu'ingénieur des exigences, vous devez souvent lire beaucoup, surtout au début d'un projet. Tous les types de documents peuvent être pertinents : documents relatifs à l'entreprise, au domaine et au projet, descriptions de produits et de processus, documents juridiques et réglementaires, etc. Comme pour les parties prenantes, vous pouvez faire une distinction entre les documents *internes* et *externes*. Les documents internes peuvent être faciles à obtenir, mais ils peuvent être confidentiels et ne peuvent être partagés sans consentement explicite.

Souvent, vous devrez signer un accord de non-divulgation avant d'y avoir accès. Les documents externes peuvent être difficiles à trouver, mais ils sont généralement publics ; si ce n'est pas le cas, assurez-vous que vous êtes autorisé à y accéder et à les utiliser.

Les documents peuvent être un excellent moyen de trouver d'autres sources. Par exemple, la description d'un processus interne peut mentionner certains rôles comme étant impliqués dans ce processus, ce qui peut vous conduire à de nouvelles parties prenantes potentielles. Cependant, les documents peuvent également être des sources directes d'exigences, en particulier celles qui sont facilement négligées ou qui ne sont pas régulièrement mentionnées par les parties prenantes : contraintes dans les normes, lignes directrices de l'entreprise et autres documents juridiques ou réglementaires ; exigences détaillées dans les procédures et les instructions de travail ; nouvelles idées brillantes dans le matériel de marketing des concurrents. L'étude de la documentation peut vous aider à comprendre le contexte du système à développer, même en lisant des courriels entre les personnes qui ont pris l'initiative du projet. La lecture de solutions analogues à des problèmes et à des objectifs dans d'autres entreprises et d'autres domaines peut stimuler votre imagination et montrer une orientation possible pour votre projet actuel.

En tant qu'ingénieur des exigences, vous devez savoir qu'un document est toujours lié à certaines personnes : le ou les auteurs, le public (cible), un responsable ou un auditeur qui en vérifie le respect, quelqu'un qui vous en a signalé l'existence, etc. Toutes ces personnes peuvent avoir un rôle à jouer en tant que parties prenantes ; il vous appartient de déterminer si c'est le cas ou non. Vous devez toujours vérifier la validité et la pertinence d'un document et vous avez souvent besoin que les parties prenantes vous le confirment. Si vous deviez dériver des exigences à partir d'un document invalide ou obsolète, le système développé à partir de ces exigences échouerait probablement.

Tout comme les parties prenantes, les documents utilisés comme sources d'exigences doivent être gérés. Vous pouvez utiliser une liste de documents, comparable à la liste des parties prenantes mentionnée ci-dessus. Tous les documents doivent être conservés dans une sorte de bibliothèque commune et indexée, avec une identification unique permettant de les référencer. Les dates et les numéros de version sont importants pour éviter de travailler avec des versions obsolètes ; vous pouvez vérifier à intervalles réguliers si une version plus récente a été publiée et si cela influe sur les exigences. Il est préférable de travailler avec des versions finales, mais dans la pratique, on est souvent confronté à des

projets, et il faut donc également enregistrer le statut des documents. Conservez les anciennes versions dans une archive, car elles peuvent être importantes pour comprendre l'évolution d'un système et de ses exigences. La mise en place d'une gestion appropriée et précise des documents concernés dès le début d'un projet vous épargnera beaucoup de travail par la suite, lors de l'ingénierie des exigences, du développement et du déploiement. Il s'agit d'un bon point de départ pour établir la traçabilité en amont, comme indiqué à la section 6.6.

4.1.3 Autres Systèmes

Vous pouvez également considérer d'autres systèmes comme des sources d'information sur les exigences du système qui vous intéresse. Ici, vous pouvez faire une distinction entre les systèmes *internes* et *externes*, tout comme dans la documentation et avec les mêmes considérations sur l'accès et la confidentialité. Une autre distinction est celle des systèmes *similaires* par rapport aux systèmes d'*interface*.

Les systèmes similaires ont certaines fonctionnalités en commun avec le système à développer. Il peut s'agir de systèmes antérieurs ou anciens, de systèmes concurrents, de systèmes comparables utilisés dans d'autres organisations, etc. On les étudie souvent à travers leur documentation, mais on peut parfois les observer en action ou les essayer comme s'il s'agissait d'une sorte de prototype. Vous pouvez contacter leurs utilisateurs pour en savoir plus sur les fonctionnalités et les solutions de ces systèmes. Les systèmes précédents ou hérités sont souvent une bonne source pour identifier les exigences et les contraintes détaillées (fonctionnelles et qualitatives).

Toutefois, il faut être conscient que les contraintes (notamment techniques) du passé peuvent ne plus être pertinentes et ne plus restreindre votre espace de solution actuel.

Parfois, un nouveau système et un système existant coexisteront pendant une certaine période, ce qui entraînera des exigences supplémentaires, par exemple en ce qui concerne le partage des données. Les systèmes concurrents et comparables peuvent être étudiés pour leurs caractéristiques de solution et peuvent constituer une bonne source d'identification des *"Excitations"* (voir section `4.2.1`).

Les systèmes d'interface auront une relation directe avec le système pour lequel vous développez les exigences. Ils échangeront des données avec votre système en tant que source et/ou puits par le biais d'une interface (synchrone ou asynchrone, en temps réel ou par lots) (voir également la section 3.4.2 sur les interfaces de système dans la modélisation contextuelle). La configuration, le contenu et le comportement corrects d'une telle interface sont souvent essentiels pour garantir le fonctionnement de votre système, et vous devrez donc comprendre le système en détail. Vous pouvez étudier les systèmes d'interfaçage en consultant leur documentation, mais comme chaque détail (technique) est important, il peut être nécessaire de procéder à des simulations ou à des tests. En ce qui concerne les systèmes anciens ou patrimoniaux en particulier, vous ne pouvez pas vous fier à la mise à jour de leur documentation et vous aurez donc besoin d'une preuve. Pour comprendre une interface, vous devez également comprendre le contexte, la fonctionnalité et le

comportement du système d'interfaçage. Il vous sera utile de contacter les gestionnaires d'applications, les architectes ou les concepteurs de ces systèmes, en particulier si le système d'interfaçage lui-même doit être mis à jour pour prendre en compte la nouvelle interface. N'oubliez pas non plus qu'un système d'interfaçage aura lui-même des utilisateurs ; il peut être intéressant de considérer ces utilisateurs comme des parties prenantes dans l'*environnement plus large* d'Alexander de votre propre système.

4.2 Élucidation des exigences

Si nous reprenons l'analogie de l'eau tirée d'un puits, nous en sommes maintenant au point où nous avons trouvé le puits et où nous commençons à tirer sur la corde pour amener le seau rempli de l'eau requise (ou, dans ce cas, des exigences) à la surface. C'est ce que nous appelons *l'élucidation* : l'effort déployé par l'ingénieur des exigences pour transformer les désirs, les demandes, les souhaits, les besoins et les attentes implicites – qui jusqu'à présent étaient cachés dans leurs sources – en exigences explicites, compréhensibles, reconnaissables et vérifiables. Bien sûr, nous devrons utiliser tous les puits pour être complets et tirer sur la corde de la bonne manière pour nous assurer que nous faisons remonter toute l'eau à la surface. Dans la terminologie de l'ingénierie des exigences, nous disons que nous devons appliquer les bonnes *techniques d'élucidation*.

Une classification courante des techniques d'élucidation consiste à distinguer :

- Techniques de collecte
- Techniques de conception et de génération d'idées

À partir de ces catégories, vous pouvez sélectionner un large éventail de techniques d'élucidation, chacune ayant ses propres caractéristiques. Figure 4.4 donne une vue d'ensemble des techniques d'élucidation dans leurs catégories et sous-catégories.

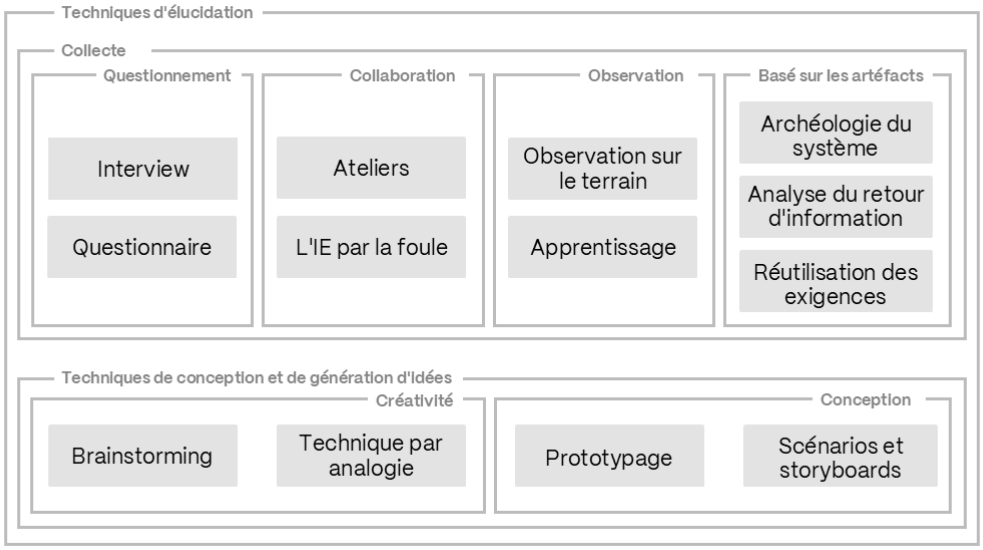


Figure 4.4 Un aperçu des techniques d'élucidation

Une compétence clé essentielle de l'ingénieur des exigences est la capacité à choisir les bonnes techniques (combinaison de techniques) dans les circonstances données. Le choix de la bonne solution peut dépendre de nombreux facteurs, tels que :

- Type de système

  Un système innovant entièrement nouveau bénéficiera davantage des techniques de conception et de génération d'idées, tandis qu'un système de remplacement dans un environnement de sécurité critique nécessitera des techniques de remise en question et une archéologie du système.

- Modèle de cycle de vie du développement logiciel

  Dans un projet en cascade, vous avez peut-être prévu des techniques approfondies telles que l'apprentissage ou les analogies, alors que dans un environnement agile, le brainstorming, le storyboard et le prototypage peuvent prévaloir.

- Personnes concernées

  Par exemple, l'observation sur le terrain ne sera probablement pas appréciée dans les entreprises où une grande confidentialité est indispensable ; une enquête globale peut être préférée à un grand nombre d'entretiens individuels.

- Structure organisationnelle

  Une organisation gouvernementale solide a besoin d'une approche totalement différente de celle d'une jeune entreprise ; une entreprise dispersée et très décentralisée a besoin d'une approche différente de celle d'une entreprise compacte implantée sur un seul site.

Les meilleurs résultats sont généralement obtenus par une combinaison de différentes techniques d'élucidation. Pour une approche systématique de leur sélection, voir [CaDJ2014].

Les techniques d'élucidation sont – ou du moins devraient être – capables de détecter tous les types d'exigences. Dans la pratique de l'ingénierie des exigences, cependant, les *exigences fonctionnelles* explicites sont souvent surestimées, et les *exigences* et *contraintes de qualité* plus implicites reçoivent moins d'attention.

Il peut en résulter un système qui, bien que toutes les exigences fonctionnelles soient remplies, n'est pas performant, est peu utilisable, n'est pas conforme aux lignes directrices architecturales ou ne répond pas à certaines autres exigences ou contraintes de qualité, et qui, par conséquent, ne sera pas accepté.

Les parties prenantes peuvent être des sources, mais vous trouverez souvent plus d'informations dans les documents. Pour l'obtention des *exigences de qualité*, l'application d'une liste de contrôle basée sur la norme ISO 25010 [ISO25010] peut aider à les détecter et à les quantifier – par exemple, lors de la préparation d'un entretien. Les *contraintes* peuvent être trouvées en considérant les restrictions de l'espace de solution potentiel – par exemple, les questions techniques, architecturales, juridiques, organisationnelles, culturelles ou

environnementales. La documentation pertinente peut souvent être identifiée par les membres du personnel.

4.2.1 Le modèle de Kano

L'une des principales circonstances à prendre en compte dans le choix d'une technique d'élucidation est la nature et l'importance de l'exigence que nous essayons de découvrir. Pour mieux comprendre la nature de certaines exigences, le modèle de Kano [Verd2014] est utile. Ce modèle, présenté dans Figure 4.5, classe les caractéristiques d'un système en trois catégories :

- *Excitation* (synonymes : facteurs de ravissement, exigences inconscientes)

  L'élément déclencheur est une caractéristique dont les clients ne sont pas conscients ; c'est pourquoi nous les appelons *inconscients*. Les clients ne demandent pas cette fonction parce qu'ils ne savent pas qu'elle est possible dans le système – par exemple, un smartphone qui peut être transformé en vidéoprojecteur. Au début, lorsque la fonction est nouvelle sur le marché, la plupart des clients ont des doutes à son sujet, mais lorsque certains utilisateurs précoces l'ont essayée et ont commencé à la faire connaître, de plus en plus de gens veulent l'avoir. Si elle absente, personne ne s'en plaindra, mais si elle est présente, elle peut constituer un élément de différenciation qui attire de nombreux clients.

- *Performance* (synonymes : facteurs de performance, exigences conscientes)

  Un facteur de satisfaction est quelque chose que les clients demandent explicitement (il s'agit donc d'exigences *conscientes*). Plus vous pouvez intégrer des facteurs de satisfaction dans votre système, plus la satisfaction des clients sera élevée. Le nombre d'objectifs et d'options vidéo d'un smartphone moderne en est un exemple. Étant donné que l'ajout de fonctions satisfaisantes entraîne généralement des coûts plus élevés, il est souvent nécessaire de procéder à une sorte d'analyse coûts/bénéfices pour décider du nombre de fonctions à incorporer dans le système.

- *Basique* (synonymes : facteurs de base, facteurs d'insatisfaction, exigences subconscientes)

  Un facteur d'insatisfaction est également une caractéristique que les clients ne demandent pas. Dans ce cas, cependant, la raison pour laquelle la demande n'est pas formulée est que la fonctionnalité est tellement évidente *(subconsciente)* que les clients ne peuvent pas imaginer qu'elle ne fasse pas partie du système ; ces fonctionnalités sont tacitement considérées comme indispensables. Imaginez un smartphone sans GPS. Si un facteur d'insatisfaction est inclus dans un système, les clients ne le remarqueront pas parce qu'ils pensent que le système ne peut pas exister sans lui. Toutefois, si vous ne tenez pas compte d'une telle exigence et que vous ne

l'intégrez pas dans le système, les clients seront très mécontents et refuseront d'utiliser le système.

Le modèle de Kano considère les exigences du point de vue du client. Il se concentre sur les caractéristiques de différenciation, par opposition aux besoins exprimés. En gardant le modèle de Kano à l'esprit, vous pouvez trouver plus d'exigences que si vous vous concentrez uniquement sur les besoins explicitement formulés par les parties prenantes. Comme nous le verrons plus loin dans ce chapitre, toutes les catégories peuvent être liées à des techniques d'élucidation distinctes.

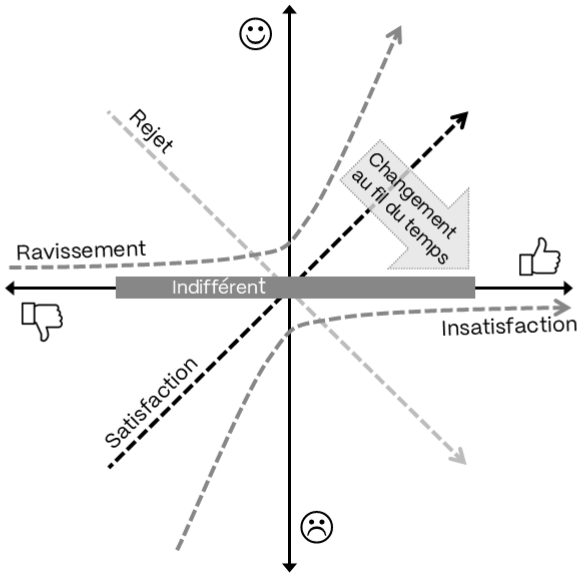


Figure 4.5 Le modèle de Kano

En fait, le modèle original de Kano contient deux catégories supplémentaires, les exigences indifférentes (ou je m'en fiche) et les exigences rejetées (ou je déteste). Ces catégories ne font pas l'objet d'une grande attention dans la plupart des manuels d'ingénierie des exigences, mais elles peuvent s'avérer utiles pour vous en tant qu'ingénieur des exigences. Supposons, par exemple, que les développeurs souhaitent ajouter une certaine fonctionnalité au système pour des raisons techniques. Si, après analyse, vous constatez que les clients sont indifférents à cette fonction, vous pouvez l'inclure dans le système. Toutefois, s'il s'agit d'une exigence de rejet, vous devez dire aux développeurs de chercher une alternative moins nocive, car la mise en œuvre de cette exigence peut s'avérer être une erreur coûteuse.

Il est intéressant d'observer, en travaillant avec le modèle Kano, que les besoins ont tendance à changer au fil du temps. Si quelqu'un introduit une nouvelle fonctionnalité, il n'y a aucune certitude quant à la manière dont le marché réagira à cette fonctionnalité. Parfois, les clients y seront indifférents et la caractéristique ne survivra que si elle n'augmente pas le prix du produit.

Si les clients la rejettent, la fonction sera probablement retirée du produit dès que possible. Toutefois, lorsque cette caractéristique plaira aux clients (peut-être initialement une avant-garde), elle deviendra un produit d'appel, un argument de vente unique pour lequel les clients sont prêts à payer le prix. Au fur et à mesure que les clients découvrent, expérimentent et apprécient cette nouvelle fonctionnalité, elle deviendra un élément de satisfaction explicitement demandé. Progressivement, lorsque des systèmes similaires commencent à mettre en œuvre la même fonctionnalité, les clients peuvent oublier que les systèmes n'incluaient pas cette fonctionnalité à l'origine et la considèrent comme acquise, ce qui en fait un facteur d'insatisfaction. C'est pourquoi de nombreux systèmes contiennent des fonctionnalités que les utilisateurs considèrent comme indispensables sans savoir pourquoi et donc sans les demander explicitement.

> Un bon exemple est la fonction appareil photo des téléphones portables, pour laquelle ce processus a pris moins de 20 ans. La première fois qu'un appareil photo a été intégré à un téléphone portable, la plupart des clients sont restés perplexes : personne n'avait demandé cette fonction et la plupart des clients pensaient "Si je veux prendre une photo, j'ai besoin d'un appareil photo." Cependant, certains utilisateurs précoces l'ont essayé et ont découvert la commodité de prendre des photos sans appareil photo dédié et de pouvoir les partager instantanément avec d'autres personnes sans avoir à les imprimer. Ils ont apprécié la fonction de l'appareil photo comme une source de plaisir et toutes les marques ont commencé à l'intégrer dans leurs téléphones, la transformant en source de satisfaction : plus les photos étaient bonnes, plus l'utilisateur était satisfait. De nos jours, lorsque l'on achète un nouveau téléphone portable, tout le monde tient pour acquis qu'il sera équipé d'un appareil photo, ce qui est devenu une source d'insatisfaction : "Si je ne peux pas prendre de photo avec ce téléphone portable, il ne sert à rien."

Comment pouvez-vous classer une caractéristique spécifique ? Vous utilisez la technique de l'analyse Kano. Pour une fonctionnalité spécifique, vous posez deux questions à un groupe représentatif d'utilisateurs potentiels : (1) "Que penseriez-vous si cette fonctionnalité était présente dans le système ?" et (2) "Que ressentiriez-vous si cette fonction était absente du système ?" Vous les laissez noter les réponses sur une échelle de 5 points entre "j'adore" et "je déteste", puis vous reportez la réponse moyenne sur la matrice d'analyse de Kano, comme indiqué à l'adresse Figure 4.5. La cellule qui s'affiche vous donne la classification de Kano pour la caractéristique.

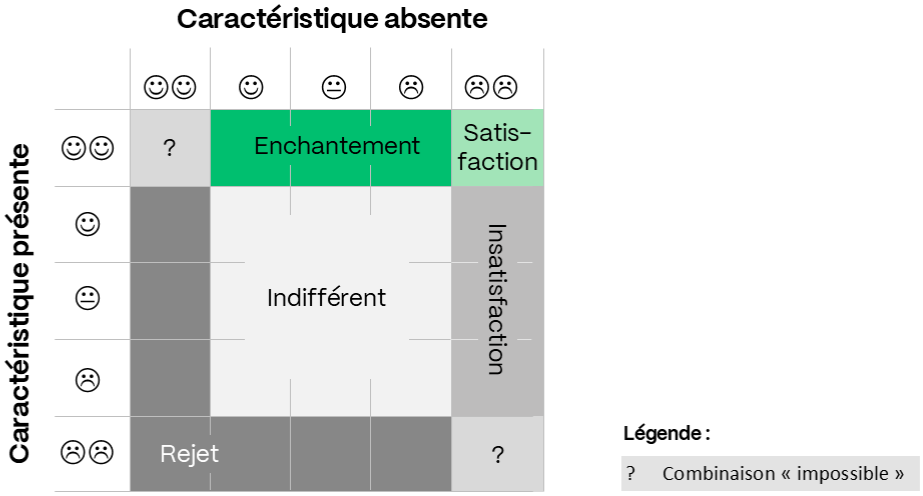


Figure 4.6 Matrice d'analyse de Kano

La question suivante est : pourquoi se préoccuper de l'analyse Kano dans le cadre de la définition des besoins ?

Comme nous l'expliquons dans les sections suivantes, vous aurez besoin de différentes techniques pour trouver ces différentes catégories de caractéristiques. D'elles-mêmes, les parties prenantes parleront principalement de leurs satisfactions, c'est-à-dire des exigences conscientes qu'elles demandent explicitement. Il est beaucoup plus difficile de détecter les autres catégories, mais il existe heureusement plusieurs techniques utiles pour y parvenir.

4.2.2 Techniques de collecte

Avec les *techniques de collecte,* vous examinez les différentes sources que vous avez identifiées et vous en tirez les exigences. Ces techniques établies ont été couramment utilisées tout au long de l'ingénierie des exigences et produisent principalement des éléments satisfaisants et insatisfaisants.

Les techniques de collecte peuvent être subdivisées en quatre catégories :

- Techniques d'interview
- Techniques de collaboration
- Techniques d'observation
- Techniques basées sur des artefacts

Les *techniques d'interview* sont toujours utilisées dans le cadre d'une interaction avec les parties prenantes. L'ingénieur des exigences pose des questions appropriées aux parties prenantes afin de les laisser réfléchir et d'obtenir des réponses à partir desquelles les exigences peuvent être dérivées.

Des exemples de techniques d'interview :

- **Interviews**

  En raison de leur souplesse, les *interviews* sont probablement l'une des techniques d'élucidation les plus fréquemment utilisées. Ils ne nécessitent pas d'outils spécifiques, et peuvent être utilisés pour obtenir des exigences de haut niveau ainsi que des exigences très spécifiques. En général, une interview est une session individuelle entre un ingénieur des exigences (interviewer) et une partie prenante individuelle (interviewé), mais un petit groupe d'interviewés est également possible. En règle générale, les exigences formulées lors d'un interview sont satisfaisantes, car la personne interrogée exprime des informations conscientes. La technique de l'interview n'est pas trop compliquée et la plupart des gens la comprennent bien. Toutefois, des objectifs clairs et une bonne préparation sont nécessaires pour obtenir des résultats utiles. Les interviews peuvent révéler des informations détaillées et offrir une certaine flexibilité en fonction des réponses données. Elles prennent beaucoup de temps, et cette technique est donc moins appropriée lorsque vous souhaitez atteindre un grand nombre de parties prenantes.

- **Questionnaires**

  À l'aide d'un *questionnaire*, un groupe plus large de parties prenantes est invité à répondre, oralement, par écrit ou sur une page Web, au même ensemble de questions, qui sont présentées de manière structurée. Les questionnaires quantitatifs sont utilisés pour confirmer des hypothèses ou des exigences préalablement formulées. Ils utilisent des questions fermées (seules des réponses prédéfinies sont autorisées) et peuvent donc être évalués rapidement et fournir des informations statistiques. D'autre part, les questionnaires qualitatifs utilisent des questions ouvertes et peuvent permettre de trouver de nouvelles exigences. Ils ont tendance à produire des résultats complexes et sont donc généralement plus longs à préparer et à évaluer. En général, les questionnaires sont une technique privilégiée pour les grands groupes. Sachez toutefois que l'élaboration d'un bon questionnaire demande beaucoup d'efforts. Un questionnaire est souvent l'étape suivante après l'obtention d'une idée préliminaire basée sur une série d'interviews afin de valider ces idées au sein d'un groupe plus large.

Dans la catégorie des *techniques de collaboration,* nous trouvons tous les types de collaboration entre l'ingénieur des exigences et d'autres personnes (parties prenantes, experts, utilisateurs, clients, etc.). En voici quelques exemples :

- **Ateliers**

  Le terme « *atelier* » est un terme générique qui désigne des techniques axées sur le travail en groupe, allant de petites réunions informelles à des événements organisés réunissant plusieurs dizaines, voire plusieurs centaines de parties prenantes. Une bonne définition est la suivante : "Un atelier sur les exigences est une réunion structurée au cours de laquelle un groupe soigneusement sélectionné de parties prenantes et d'experts en contenu travaillent ensemble pour définir, créer, affiner et

conclure des produits livrables (tels que des modèles et des documents) qui représentent les exigences de l'utilisateur." [Gott2002]. Un atelier permet d'obtenir une bonne vision globale en peu de temps grâce à l'interaction entre les participants. Si vous avez besoin de plus de détails, des interviews de suivi ou des questionnaires sont appropriés. Les ateliers peuvent servir de technique de rassemblement, mais ils peuvent également être utilisés dans le cadre de techniques de créativité (voir section 4.2.3).

- **Ingénierie des exigences basée sur le Crowd**

  Dans l'ingénierie des exigences *basée le crowd* (également connue sous le nom de plateforme) (voir [GreA2017]), l'élucidation est transformée en un effort participatif avec une foule de parties prenantes, en particulier les utilisateurs, ce qui permet d'obtenir des exigences plus précises et, en fin de compte, de meilleurs logiciels. Le pouvoir du crowd réside dans la diversité des talents et de l'expertise disponibles en son sein. Comme la quantité de données obtenues du crowd sera importante, une plateforme automatisée pour le traitement de ces données est essentielle. Cette plateforme devrait offrir des fonctionnalités orientées vers la communauté qui soutiennent la collaboration et le partage des connaissances et favorisent l'engagement de groupes plus importants de parties prenantes dans la collecte, l'analyse et le développement des exigences logicielles, ainsi que la validation et la hiérarchisation de ces exigences d'une manière dynamique et axée sur l'utilisateur.

Les *techniques d'observation* sont également appliquées en relation avec les parties prenantes. Les parties prenantes sont observées alors qu'elles sont engagées dans leurs processus (métier) normaux, dans leur contexte habituel, sans intervention directe de l'ingénieur des exigences. Les techniques d'observation sont particulièrement utiles pour identifier les personnes insatisfaites. Vous pouvez observer des activités, des séquences, des données, etc. particulières qui sont tellement communes aux parties prenantes qu'elles ne les mentionnent pas, et ces aspects ne sont donc pas facilement mis en évidence par les techniques de collecte.

Les techniques d'observation courantes sont les suivantes :

- **Observation sur le terrain**

  Lors de l'*observation sur le terrain*, l'ingénieur des exigences observe (la plupart du temps) les utilisateurs finaux dans leur environnement pendant qu'ils effectuent les activités pour lesquelles un système doit être développé. L'observation sur le terrain est généralement utilisée dans les situations où l'interaction risquerait de distraire les utilisateurs ou interférerait avec le processus lui-même et risquerait de fausser les résultats. Elle peut même être appliquée sans que les sujets observés en soient informés, par exemple en s'asseyant avec d'autres patients dans la salle d'attente d'un dentiste pour observer leur comportement. L'observation sur le terrain vous permettra de détecter des besoins (souvent détaillés) qui ne seraient pas facilement

décelés avec d'autres techniques, par exemple parce que les actions et les comportements sont trop compliqués à exprimer en mots.
Il faut savoir que l'observation sur le terrain demande beaucoup de préparation, un œil aiguisé et beaucoup de temps. La vidéo est très utile pour saisir le comportement des parties prenantes. Elle peut être utilisée conjointement avec l'observation directe sur le terrain et peut même la remplacer dans les situations où la présence effective de l'ingénieur des exigences n'est pas autorisée ou souhaitée. La vidéo offre la possibilité d'un post-traitement qui permet d'étudier en détail des actes et des procédures difficiles à observer.

- **Apprentissage**

  L'*apprentissage* diffère de l'observation sur le terrain en ce qu'il est participatif. Dans le cadre de l'apprentissage, l'ingénieur des exigences *(l'apprenti*) effectue une sorte de stage dans l'environnement dans lequel le système en question sera utilisé (ou est déjà utilisé) et des utilisateurs expérimentés *(les maîtres*) enseignent à l'apprenti comment les choses fonctionnent. L'apprenti participe mais n'intervient pas ; il se comporte comme un novice dans le domaine et a le droit de faire des erreurs et de poser des questions "idiotes". L'objectif est d'acquérir une connaissance approfondie du domaine, de l'entreprise et des processus avant de commencer l'élucidation proprement dite des exigences. Un suivi au moyen d'interviews et de questionnaires sera souvent nécessaire pour vérifier les idées initiales. Les exigences qui en résultent peuvent ensuite être documentées et validées. La durée optimale d'un tel stage dépend de nombreux facteurs (par exemple, la complexité du processus, le caractère très répétitif, la disponibilité du maître et de l'apprenti), mais varie généralement entre un jour et plusieurs semaines. Sachez que l'apprentissage peut être difficile ou impossible à organiser dans certains domaines, tels que la médecine, l'aviation ou l'armée.

Les *techniques basées sur les artefacts* n'utilisent pas les parties prenantes (directement) mais plutôt des produits d'activités tels que des documents et des systèmes, ou même des images, des fichiers audio et vidéo, comme sources d'exigences. Ces techniques permettent de trouver des éléments satisfaisants et insatisfaisants (parfois très détaillés). L'examen détaillé des produits d'activités (souvent mal structurés, obsolètes ou en partie non pertinents) est généralement une tâche qui prend du temps. Néanmoins, les techniques basées sur les artefacts sont très efficaces et offrent des avantages significatifs, en particulier lorsque les parties prenantes ne sont pas facilement disponibles.

Voici quelques exemples de techniques basées sur les artefacts :

- **Archéologie du système**

  Dans l'*archéologie du système*, les exigences sont extraites des systèmes existants – tels que les systèmes patrimomiaux, les systèmes concurrents ou même les systèmes analogues – en analysant leur documentation (conceptions, manuels) ou leur mise en œuvre (code, commentaires, scripts, user stories, cas de test). Cette technique est principalement utilisée lorsqu'un système existant a été utilisé pendant de

nombreuses années et doit être remplacé par un nouveau système pour une raison quelconque ; le nouveau système doit couvrir les mêmes fonctionnalités que l'ancien, ou du moins certaines parties de celles-ci. L'archéologie des systèmes prend souvent beaucoup de temps mais peut révéler des exigences et des contraintes détaillées qui ne sont pas facilement détectées autrement. Toutefois, vous aurez besoin de temps supplémentaire pour vérifier, par d'autres moyens, si ces exigences sont toujours valables et pertinentes.

- **Analyse des documents**

  Analyse de documents [Bowe2009] est une méthode de recherche qualitative qui permet d'acquérir des connaissances par l'étude structurée et l'interprétation de textes et d'autres formes d'information, telles que des chiffres.
  Les étapes suivantes sont reconnues :
  1. Collecter des documents
     Les documents peuvent être trouvés en interrogeant les clients et les autres parties prenantes. Les documents analysés précédemment peuvent faire référence à d'autres documents, ce qui fait de l'analyse des documents un processus itératif.
  2. Sélectionner les documents
     A partir d'un large éventail de documents collectés, déterminer ceux qui sont pertinents pour l'élicitation. Tenez compte de la validité, de l'authenticité et du contenu.
  3. Analyser les documents
     Examiner les documents sélectionnés pour en extraire les exigences. Il peut s'agir d'exigences fonctionnelles et de qualité, ainsi que de contraintes.
  4. Interpréter les résultats
     L'ensemble des exigences extraites est rarement sans ambiguïté. Ils peuvent être décrits à différents niveaux d'abstraction, se contredire, se chevaucher, être vagues, incomplets et ouverts à de multiples interprétations. La *triangulation* (comparaison des exigences provenant de différentes sources) permet d'obtenir un ensemble cohérent d'exigences pertinentes.

- **Analyse du retour d'information**

  Il existe de nombreuses façons de recueillir les *réactions* des utilisateurs (potentiels) et des clients, que ce soit sur un système existant ou sur un prototype. Les données de retour d'information peuvent être structurées (par exemple, une note de 5 étoiles dans un magasin d'applications) ou non structurées (comme des commentaires d'évaluation). Elles peuvent être recueillies par le biais d'enquêtes en ligne et de formulaires de contact, lors de tests bêta ou A/B, sur les médias sociaux, ou même sous la forme de remarques de clients reçues dans un centre d'appel. Souvent, la quantité de données est très importante et l'analyse prend du temps. Toutefois, le retour d'information peut être très utile pour comprendre les *difficultés et les avantages de* l'utilisateur. Les notes négatives et les remarques critiques vous aideront à détecter les mécontents qui passent inaperçus. Les notes positives et les

compliments vous donneront des informations supplémentaires sur les personnes satisfaites. Parfois, les commentaires peuvent même contenir des idées innovantes qui peuvent être transformées en "facteurs d'excitation". L'analyse du retour d'information peut donc aboutir à l'ajustement des exigences existantes, mais aussi à la découverte de nouvelles exigences.

- **Réutilisation des exigences**

  De nombreuses organisations disposent déjà d'un grand nombre d'exigences qui ont été formulées et élaborées dans le passé pour des systèmes antérieurs. Nombre de ces exigences peuvent également s'appliquer à un nouveau système, en particulier celles qui ont été dérivées d'un modèle de domaine global. Par conséquent, la *réutilisation* des exigences existantes permet d'économiser beaucoup de temps et d'argent, car il est possible d'éviter leur élucidation. Toutefois, cela ne fonctionne que si ce recueil d'exigences existantes est à jour, géré efficacement, facilement disponible et largement documenté, ce qui n'est malheureusement pas souvent le cas. Même si la réutilisation est possible, n'oubliez pas que vous devez toujours valider avec les parties prenantes si ces exigences réutilisables sont pertinentes et valables dans la nouvelle situation, que ce soit directement ou avec quelques ajustements.

4.2.3 Techniques de conception et de génération d'idées

Dans le passé, l'ingénierie des exigences s'est concentrée sur la collecte et la documentation des exigences nécessaires auprès de toutes les parties prenantes concernées en appliquant les techniques de collecte présentées dans la section précédente. L'influence croissante des logiciels en tant que moteur de l'innovation dans de nombreuses entreprises exige désormais de plus en plus un nouveau positionnement de l'ingénierie des exigences en tant qu'activité créative de résolution de problèmes. Cela implique l'application d'autres techniques qui ne considèrent plus les parties prenantes (et leurs documents et systèmes) comme la seule et unique source d'exigences. Les systèmes innovants ont besoin de caractéristiques nouvelles, voire perturbatrices, que les parties prenantes actuelles ne peuvent pas (encore) imaginer.

Des *techniques de conception et de génération d'idées* sont apparues pour répondre à ce besoin. Ces techniques sont destinées à stimuler la créativité, principalement au sein des équipes, pour la génération d'idées et peuvent fournir des moyens supplémentaires d'élaborer une idée donnée. Ces techniques peuvent donner lieu à des exigences nouvelles et innovantes qui sont souvent source d'excitation. Il existe de nombreuses techniques différentes dans cette vaste catégorie, certaines remarquablement simples, d'autres très élaborées. Nous examinerons quelques exemples de deux sous-catégories :

- Techniques de créativité
- Techniques de conception

En outre, nous nous pencherons sur le domaine émergent du *design thinking*.

Les *techniques de créativité* stimulent la créativité afin de trouver ou de créer de nouvelles exigences qui ne peuvent être recueillies directement auprès des parties prenantes parce que celles-ci ne sont pas conscientes de la faisabilité de certaines nouvelles caractéristiques ou innovations (techniques). Ces techniques sont généralement appliquées au sein d'équipes informatiques diverses et pluridisciplinaires composées d'analystes, d'ingénieurs des exigences, de développeurs, de testeurs, de product owners, de gestionnaires d'applications, etc., avec ou sans représentants métier, des utilisateurs, des clients et d'autres parties prenantes. Les techniques stimulent la réflexion hors des sentiers battus et sans frontières, ainsi que l'élaboration des idées de chacun. Malheureusement, aucun d'entre eux ne garantit le succès de la création, car plusieurs mécanismes de notre cerveau doivent se conjuguer pour donner naissance à des idées créatives.

> L'industrie des jeux est un exemple évident de l'importance des techniques de créativité. Vous pouvez bien sûr demander aux joueurs de vous faire part de leurs besoins grâce à une technique de collecte et vous apprendrez ainsi ce que les joueurs aiment ou n'aiment pas dans les jeux actuels. Cependant, pour développer un jeu réussi, vous devez surprendre les joueurs avec quelque chose de nouveau ; vous devez découvrir leurs facteurs d'excitation. C'est précisément là que les techniques de créativité entrent en jeu.

Plusieurs conditions préalables ont été identifiées comme des facteurs importants pour l'émergence de la créativité :

- Le hasard – et donc le temps – pour qu'une idée se présente
- La connaissance du sujet, qui augmente les chances de trouver une idée qui fasse la différence
- La motivation, car notre cerveau ne peut être créatif que si son propriétaire en tire un bénéfice direct
- La sûreté et la sécurité, car les idées inutiles ne doivent pas avoir de conséquences négatives

Deux exemples de techniques de créativité sont présentés ici :

- **Brainstorming**

  Le *brainstorming* (voir [Osbo1948]) favorise le développement de nouvelles idées pour une question ou un problème donné. Comme pour la plupart des techniques de créativité, le point crucial du brainstorming est de différer le jugement en séparant la recherche d'idées de l'analyse des idées. Voici quelques lignes directrices générales pour le brainstorming :
  La quantité prime sur la qualité.
  La libre association et la pensée visionnaire sont explicitement souhaitées.
  La reprise et la combinaison des idées exprimées sont permises et souhaitées.
  Il est interdit de critiquer les idées des autres participants, même si elles semblent absurdes.

Après une séance de brainstorming, les idées qui ont émergé sont classées, évaluées et priorisées. Les idées sélectionnées servent ensuite de base à la poursuite de l'élucidation.

- **Technique par analogie**

  La *technique par analogie* (voir [Robe2001]) aide à développer des idées sur des sujets critiques et complexes. Elle utilise des analogies pour soutenir la réflexion et générer des idées. Son succès ou son échec est principalement influencé par le choix d'une analogie appropriée pour le problème donné. L'analogie choisie peut être proche (par exemple, le même problème dans une autre entreprise) ou éloignée (par exemple, comparer une organisation à un organisme vivant) du problème original. L'application de la technique par analogie se fait en deux étapes :
  Élaborer en détail les aspects de l'analogie choisie sans se référer au problème original.
  Transférer tous les aspects identifiés de l'analogie au problème original.
  Les concepts et les idées qui en résulteront serviront ensuite de point de départ à d'autres demandes de renseignements.

Les techniques de conception permettent d'explorer et d'élaborer les idées générées par les techniques de créativité et aident également à clarifier et à concrétiser les besoins vagues des parties prenantes. Ils s'appuient fortement sur des objets visuels ou tangibles, sur la coopération au sein de l'équipe et sur le retour d'information de la part des clients.

Les techniques les plus courantes dans cette catégorie sont les suivantes :

- **Prototypage**

  Par *prototype* (en relation avec l'élucidation ; voir également la section 3.7 pour plus d'informations), nous entendons une sorte de produit intermédiaire qui est créé ou diffusé pour générer un retour d'information. Les prototypes peuvent aller de simples esquisses sur papier à des versions préliminaires fonctionnelles d'un système. Ils permettent aux futurs utilisateurs d'expérimenter le système d'une manière plus ou moins tangible et d'étudier certaines caractéristiques, encore floues, pendant la phase d'ingénierie des exigences et avant la mise en œuvre proprement dite. Comme nous le verrons dans la section sur la validation (`4.4.2`), les prototypes sont principalement utilisés pour vérifier que les exigences définies précédemment ont été mises en œuvre correctement. Toutefois, si les utilisateurs sont correctement guidés et que leurs réactions sont analysées, cette technique peut également être utilisée pour définir de nouvelles exigences. Elle peut être particulièrement utile pour détecter les exigences non fonctionnelles, les insatisfactions et les contraintes, ou toute autre caractéristique qui ne peut pas être facilement comprise ou définie d'emblée dans les modèles et la documentation.

- **Scénarios et storyboards**

  Le mot *scénario* vient du théâtre, où il est utilisé pour désigner le plan d'une pièce, d'un opéra ou d'une œuvre similaire, indiquant une séquence de scènes avec leurs

personnages. En informatique, nous utilisons ce terme pour décrire un flux d'actions pour un système, y compris les utilisateurs impliqués (que nous appelons généralement ici acteurs). Grâce à des scénarios, vous pouvez explorer différentes façons de mettre en œuvre un processus dans un système. Grâce à leur structure légère, ils sont faciles à développer et peuvent être modifiés rapidement. De la même manière que pour les prototypes, les scénarios et les storyboards peuvent être utilisés à la fois pour l'obtention (précoce) et la validation (ultérieure) des exigences.
Les scénarios peuvent être documentés sous forme écrite ou visuelle. La forme visuelle d'un scénario s'appelle un *storyboard*.
Un storyboard est typiquement une sorte de bande dessinée avec une série de panneaux qui montrent l'interaction de certains personas avec le système. Voir Figure 4.7 pour un exemple. Les scénarios et les storyboards sont utiles pour l'élaboration précoce d'idées en termes de processus et d'activités.

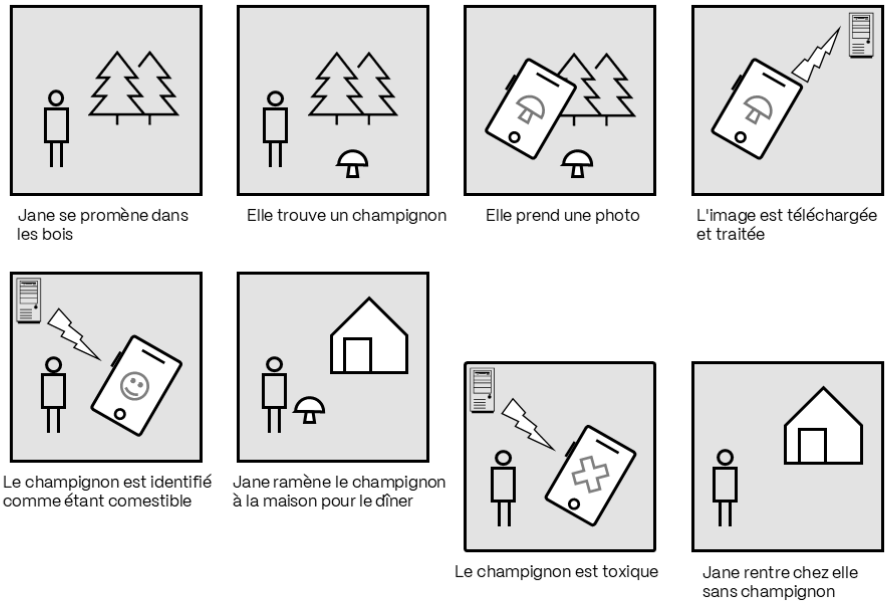


Figure 4.7 Exemple de storyboard

Le *Design thinking* n'est pas tant une technique qu'un concept, une attitude, une philosophie, une famille de processus et souvent une boîte à outils remplie de techniques. L'accent est mis sur l'innovation et la résolution de problèmes. Il existe plusieurs variantes du design thinking, qui utilisent principalement des techniques légères, visuelles et agiles. Deux principes de base se retrouvent dans toutes les variantes :

- **Empathie**

  La première étape pour les concepteurs consiste à trouver le véritable problème qui se cache derrière le problème donné. Ils tentent de comprendre ce que les parties prenantes pensent, ressentent et font réellement lorsqu'elles interagissent avec un système. C'est pourquoi nous parlons souvent de design thinking comme d'une conception centrée sur l'homme. Les personas, la cartographie de l'empathie et la co-création du client sont des techniques courantes à cette fin.

- **Créativité**

  Une caractéristique commune au design thinking est le *diamant*: l'alternance de la pensée divergente et de la pensée convergente. La pensée divergente vise à explorer une question plus largement et plus profondément, en générant de nombreuses idées différentes, et la pensée convergente concentre, sélectionne, élague et combine ces idées en un seul résultat final. Un modèle de base, le modèle à *double diamant*, est illustré à l'adresse Figure 4.8 (voir [DeCo2007]).

Un traitement détaillé du design thinking dépasse le cadre de ce manuel de niveau Fondation.

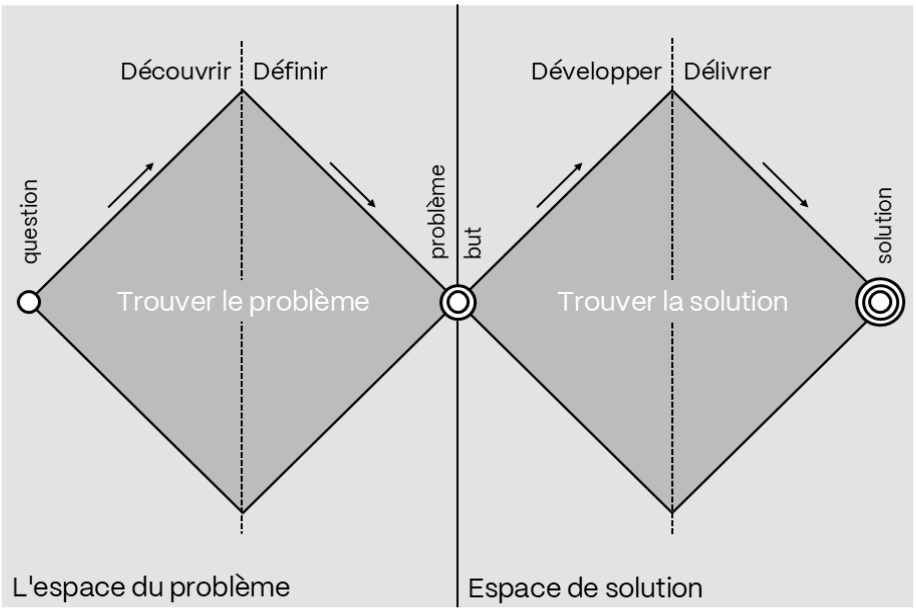


Figure 4.8 Le double diamant

4.3 Résoudre les Conflits concernant les Exigences

Au cours de la phase d'élucidation, vous rassemblez un large éventail d'exigences provenant de différentes sources, à l'aide de différentes techniques et à différents niveaux d'abstraction et de détail. Les techniques d'élucidation que vous utilisez ne garantissent pas en elles-mêmes que cette collection dans sa totalité forme un ensemble unique, cohérent et convenu d'exigences qui capture l'essence du système. Pendant et après l'élucidation d'un ensemble d'exigences pour un certain système, vous pouvez découvrir que certaines exigences sont conflictuelles : elles peuvent être incohérentes, incompatibles, contradictoires. Il se peut que des exigences soient contradictoires (par exemple, "tout le texte doit être en noir et blanc" contre "tous les messages d'erreur doivent être rouges") ou que certaines parties prenantes aient une opinion différente sur la même exigence (par exemple, "tous les messages d'erreur doivent être rouges" contre "les messages d'erreur de l'utilisateur doivent être rouges, tous les autres messages d'erreur doivent être bleus").

Comme nous ne pouvons pas développer un système (ou une partie spécifique d'un système) sur la base d'exigences contradictoires, les conflits doivent être résolus avant que le développement ne puisse commencer. En tant qu'ingénieur des exigences, c'est vous qui devez vous assurer que toutes les parties prenantes parviennent à une compréhension commune (voir chapitre 2, principe 3) de l'ensemble des exigences dans la mesure où elles les concernent et qu'elles se mettent d'accord sur cet ensemble.

Mais qu'est-ce qu'un conflit ? Un conflit est un désaccord entre des personnes : "une interaction entre des agents (individus, groupes, organisations, etc.), où au moins un agent perçoit des incompatibilités entre ses pensées/idées/perceptions et/ou sentiments et/ou volonté et ceux de l'autre agent (ou des autres agents), et se sent limité par l'action de l'autre" [Glas1999]. Dans un conflit d'exigences, deux parties prenantes ou plus ont un avis différent, voire contradictoire, sur une certaine exigence ou leurs exigences ne peuvent pas être mises en œuvre en même temps dans un certain système ; voir Figure 4.9.

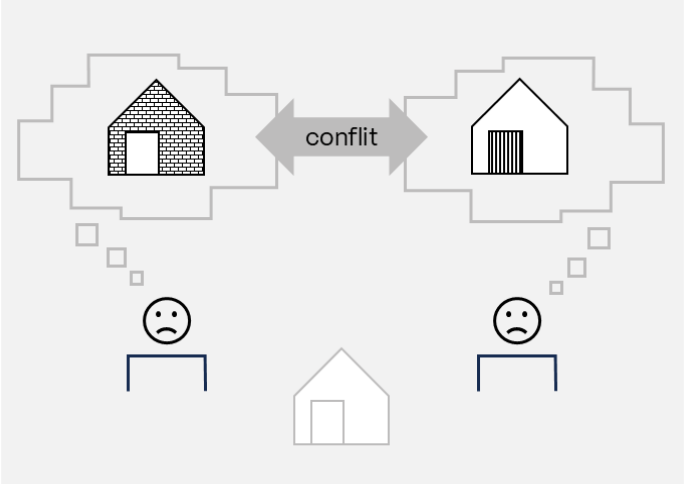


Figure 4.9 Un conflit d'exigences

Gérer les conflits liés aux exigences peut être difficile, pénible et prendre beaucoup de temps, surtout lorsque des questions personnelles sont en jeu. Cependant, nier ou ignorer les conflits n'est pas une option, et l'ingénieur des exigences doit donc chercher activement des moyens de les résoudre. Au final, toutes les parties prenantes doivent comprendre et accepter toutes les exigences qui les concernent. Si certaines parties prenantes ne sont pas d'accord, cette situation doit être reconnue comme un conflit qui doit être résolu en conséquence.

4.3.1 Comment résoudre un conflit d'exigences ?

Pour résoudre correctement un conflit d'exigences, il convient de suivre les étapes suivantes :

- **Identification du conflit**

  Nous sommes souvent confrontés à des conflits dans notre vie quotidienne. Ils nous procurent une sensation désagréable, et une stratégie courante consiste donc à les éviter, à les ignorer ou à les nier. Cela peut rendre les conflits difficiles à identifier. La plupart d'entre eux ont tendance à être cachés et ne peuvent être détectés que par une observation attentive. Il existe de nombreux indicateurs auxquels vous pouvez prêter attention, tant au niveau de la communication que de la documentation :
  Dans la communication, vous pouvez observer des comportements tels que le déni, l'indifférence, la pédanterie, la demande permanente de détails supplémentaires, les interprétations délibérément incorrectes, la dissimulation ou la délégation.
  Dans la documentation, vous pouvez trouver des éléments tels que des déclarations contradictoires des parties prenantes, des résultats contradictoires de l'analyse des documents ou des systèmes, des incohérences entre différents niveaux de détail et une utilisation incohérente des termes.
  Si vous observez de tels indicateurs, cela ne signifie pas nécessairement qu'il y a un conflit d'exigences, mais vous devriez certainement vous méfier. Une discussion approfondie avec les parties prenantes peut alors faire remonter à la surface un conflit caché.

- **Analyse du conflit**

  Une fois qu'un conflit a été identifié, l'ingénieur des exigences doit d'abord déterminer s'il s'agit d'un conflit d'exigences ou non. Après tout, un conflit d'exigences relève de la responsabilité première de l'ingénieur des exigences ; les autres conflits peuvent être résolus par d'autres participants, tels qu'un chef de service ou un chef d'équipe.
  L'ingénieur des exigences doit comprendre parfaitement la nature du conflit avant de tenter de le résoudre. Cela signifie que vous devrez recueillir davantage d'informations sur le conflit lui-même et sur les parties prenantes impliquées.
  De nombreux aspects méritent l'attention :

  - *Sujet* : la portée, le problème ou la véritable question qui sous-tend le conflit.
  - *Exigences concernées*: quelles sont les exigences spécifiques concernées ?
  - *Parties prenantes impliquées*: qui n'est pas d'accord avec qui sur quoi ?
  - *Opinions* des parties prenantes : laissez-les exprimer leur point de vue aussi clairement que possible afin que toutes les parties en conflit comprennent le problème sous-jacent.
  - La *cause* du conflit : quelle est la raison de la divergence d'opinions ?
  - L'*histoire* du conflit : qu'est-ce qui s'est passé auparavant et qui influence ces opinions aujourd'hui ?
  - *Les conséquences*: estimation des coûts et des risques associés à la résolution ou à la non-résolution du conflit.

- *Les contraintes du projet* : les contraintes personnelles, organisationnelles, spécifiques au contenu ou spécifiques au domaine peuvent déterminer l'espace de solutions.
- L'analyse de ces informations vous aidera à reconnaître le type de conflit (pour plus d'informations, voir la section 4.3.2) et vous indiquera des moyens de le résoudre.

- **Résolution des conflits**

  Une fois qu'il a bien compris la nature du conflit, l'attitude des parties prenantes et les contraintes du projet, l'ingénieur des exigences choisit une technique de résolution appropriée. De nombreuses techniques peuvent être utilisées, comme l'explique la section 4.3.3. La première étape devrait toujours consister à faire accepter la technique choisie par les parties prenantes concernées avant de l'appliquer. Si certaines parties prenantes n'acceptent pas d'emblée l'application d'une certaine technique, elles n'en accepteront certainement pas le résultat, de sorte qu'au bout du compte, le conflit ne sera pas résolu. En principe, l'ingénieur des exigences n'est pas l'une des parties prenantes concernées. Vous pouvez et devez donc appliquer les techniques de résolution sélectionnées de manière objective et strictement neutre, et accueillir favorablement tout résultat découlant de l'application de la technique.

- Documentation de la résolution du conflit. La résolution des conflits peut influencer les exigences d'une manière qui n'est pas évidente pour quelqu'un qui n'a pas été impliqué dans le conflit. L'ensemble des exigences qui en résulte peut sembler illogique ou inefficace. Par conséquent, la résolution du conflit doit être correctement documentée et communiquée en ce qui concerne des aspects tels que les suivants :
  - Hypothèses concernant le conflit et sa résolution
  - Alternatives potentielles envisagées
  - Contraintes influençant la technique et/ou la résolution choisies
  - La manière dont le conflit a été résolu, y compris les raisons pour la résolution choisie
  - Décideurs et autres contributeurs
  - Si vous ne documentez pas la résolution, au bout d'un certain temps, les parties prenantes risquent d'oublier ou d'ignorer les décisions qui ont été prises. Et plus tard dans le projet, les développeurs peuvent ne pas comprendre le raisonnement qui sous-tend la conception d'un système particulier et le mettre en œuvre d'une manière différente.

Vous ne devez pas craindre les conflits d'exigences, car ils se produiront toujours. Cela ne devrait pas vous surprendre ; en fait, vous devriez être inquiet si vous ne détectez aucun conflit. Ils sont assez courants, donc si vous ne les trouvez pas, c'est que vous en avez probablement manqué. Mais ne les ignorez jamais. Si vous ne résolvez pas immédiatement tous les conflits d'exigences que vous remarquez, ils réapparaîtront plus tard dans le processus de développement. Et comme Barry Boehm [Boeh1981] l'a déjà constaté il y a longtemps, plus on découvre un problème tard, plus il sera coûteux de le résoudre.

4.3.2 Les types de conflits

Pour mieux comprendre la nature d'un conflit, il est utile de distinguer différents types de conflits. Cela permet de sélectionner les techniques de résolution appropriées.

Nous discernons six types de conflits :

- **Conflit de sujets**

  Un *conflit de sujets* se produit lorsque les parties en conflit ont réellement des besoins factuels différents, principalement en raison de l'utilisation prévue du système dans des environnements différents. Un bon exemple est un système destiné à être utilisé dans différents pays, chacun ayant sa propre législation. Il peut être difficile de résoudre un tel conflit car les faits sous-jacents ne peuvent être modifiés. La première chose à faire est alors d'analyser et de documenter ces faits en détail et de faire en sorte que les parties en conflit se mettent d'accord sur la nature exacte du conflit.

- **Conflit de données**

  Il y a *conflit de données* lorsque certaines parties se réfèrent à des données incohérentes provenant de sources différentes ou interprètent les mêmes données d'une manière différente. Cela peut être dû à une mauvaise communication, à des données de base manquantes, à des différences culturelles, à des préjugés existants, etc. Les estimations en particulier, telles que les ventes futures, peuvent facilement générer un conflit de données car elles sont souvent basées sur des hypothèses. Détecter un conflit de données n'est pas facile, car en tant qu'ingénieur des exigences, vous pouvez penser que vos propres sources sont justes et que votre propre interprétation est évidente. En raison de ce biais, vous soupçonnez souvent un autre type de conflit au départ.
  Comprendre comment les gens peuvent arriver à une interprétation différente demande beaucoup d'empathie. La communication – encore et encore – est essentielle pour détecter et résoudre ce type de conflit.

- **Conflit d'intérêt**

  Un *conflit d'intérêts* est basé sur les différentes positions des parties en conflit, formées par des objectifs personnels, des objectifs liés à un groupe ou des objectifs liés à un rôle. Vous devez comprendre les préoccupations et les besoins des parties prenantes concernées avant de pouvoir résoudre ce type de conflit. Cependant, gardez à l'esprit que dans le cas d'intérêts personnels, les parties prenantes ne révèlent souvent pas leurs véritables motivations et avancent des arguments apparemment factuels, mais essentiellement artificiels. Si la discussion porte sur un conflit d'intérêts, vous pouvez observer les parties en conflit qui tentent de convaincre l'autre de suivre leurs arguments et de comprendre les besoins du rôle ou du groupe. La résolution peut bénéficier de l'identification et du renforcement

d'intérêts communs. La recherche d'une compréhension mutuelle des avantages et des inconvénients pour les deux parties peut être le point de départ d'une solution.

- **Conflit de valeur**

  Un *conflit de valeurs* repose sur des différences de valeurs et de principes entre les parties prenantes concernées. Par rapport à un conflit d'intérêts, un conflit de valeurs est plus individuel et lié à des perspectives globales et à long terme. Les valeurs sont plus stables que les intérêts et changent rarement à court terme. Si un conflit de valeurs est à l'origine d'une discussion, les parties en conflit souligneront pourquoi leurs arguments sont importants de leur point de vue, révélant ainsi leurs valeurs et principes internes. Ils ont tendance à insister sur leurs arguments et ne sont pas prêts à renoncer. Pour résoudre ces conflits, il faut rechercher des valeurs plus élevées qui unissent les parties. Les conflits de valeurs sont notoirement difficiles à résoudre et le mieux que l'on puisse faire est de parvenir à une compréhension et à une reconnaissance mutuelles des principes de l'autre.

- **Conflit relationnel**

  Un *conflit relationnel* est généralement basé sur des expériences négatives avec une autre partie dans le passé, ou dans des situations comparables avec des personnes similaires. Souvent, les émotions et les mauvaises communication entrent en jeu, ce qui rend le conflit beaucoup plus difficile à résoudre. Les parties en conflit abusent des discussions sur les exigences pour exprimer leur colère à l'égard du comportement de l'autre partie, oubliant les faits, les chiffres et l'équité. Ramener la discussion aux exigences est rarement utile ; parfois, il suffit d'unir les parties autour d'une valeur plus élevée pour réussir. Dans la plupart des cas, vous devrez transmettre le problème à d'autres parties prenantes ou à un niveau d'autorité supérieur; l'échange de personnes est une solution possible. Sachez qu'un conflit relationnel coïncide souvent avec d'autres types de conflits, par exemple un conflit d'intérêts. L'analyse de la cause première et la résolution de l'autre type de conflit peuvent alors constituer le meilleur moyen d'améliorer la relation.

- **Conflit structurel**

  Nous qualifions un conflit de *structurel* lorsqu'il implique une inégalité de pouvoir, une concurrence pour des ressources limitées ou des dépendances structurelles entre les parties. Le déséquilibre qui en résulte (souvent perçu par une seule des parties) entraîne des problèmes de communication et de prise de décision. Une autre raison pouvant expliquer ces conflits réside dans les restrictions liées aux ressources ou dans la dépendance vis-à-vis des produits livrables fournis par une autre partie. Les parties peuvent utiliser la discussion sur les exigences pour modifier ou préserver le statu quo. La hiérarchie peut être utilisée à mauvais escient pour faire passer des décisions. Pour les conflits structurels également, il est souvent nécessaire de faire

remonter le problème à d'autres parties prenantes ou à un niveau d'autorité plus élevé.

La plupart des conflits d'exigences peuvent être classés dans l'une des catégories suivantes : conflit de sujet, conflit de données, conflit d'intérêts ou conflit de valeurs. Les conflits relationnels et structurels ne sont souvent pas directement liés aux exigences et, par conséquent, l'ingénieur des exigences n'est peut-être pas la partie appropriée pour les résoudre. Toutefois, en réalité, la plupart des conflits relèvent de plus d'une catégorie, car les différentes causes interagissent. Il est donc conseillé de prêter attention à tous les types de conflits, même si la solution ne relève pas de votre propre responsabilité. Si quelqu'un d'autre doit résoudre le conflit, assurez-vous qu'il le fasse ; tant qu'un conflit n'est pas résolu, il continuera à avoir un impact négatif sur votre travail en tant qu'ingénieur des exigences.

4.3.3 Techniques de résolution des conflits

En fonction du type et du contexte (parties prenantes, contraintes, etc.) d'un conflit, une technique de résolution appropriée est sélectionnée. Les techniques couramment utilisées sont les suivantes [PoRu2015] :

Accord

Un *accord* résulte d'une discussion entre les parties prenantes concernées, qui doit se poursuivre jusqu'à ce qu'elles comprennent parfaitement leurs positions respectives et s'accordent sur une certaine option privilégiée par toutes les parties. Cela peut prendre beaucoup de temps, surtout lorsque plusieurs parties sont impliquées. En cas de succès, il apportera une motivation supplémentaire aux parties prenantes, de sorte que le résultat a de bonnes chances d'être durable. La recherche d'un accord est fréquente dans les conflits de données. Si cette technique n'aboutit pas dans un délai acceptable, d'autres techniques peuvent être utilisées par la suite.

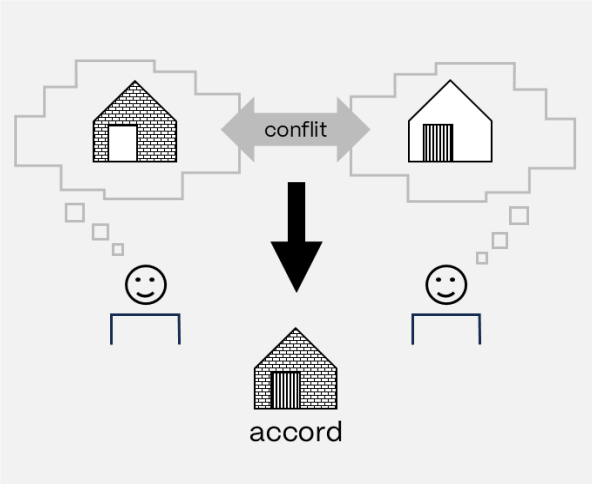


Figure 4.10 Accord

Compromis

Un *compromis* est assez similaire à un accord. Dans ce cas, les parties prenantes se mettent d'accord sur une option qui n'a pas leur préférence, mais avec laquelle elles peuvent vivre parce qu'elles considèrent qu'il vaut mieux accepter le compromis que de poursuivre le conflit. Par conséquent, un compromis peut également être durable. Le compromis peut contenir de nouveaux éléments qui ne figuraient pas dans les préférences initiales des parties prenantes et qui peuvent avoir été introduits par l'ingénieur des exigences. Un bon compromis est une alternative dans laquelle toutes les parties se sentent à l'aise avec l'équilibre entre l'abandon de certaines choses et l'obtention de quelque chose d'autre en retour. Si un accord ne peut être trouvé à temps, le compromis est souvent la solution suivante. Elle est adaptée aux conflits de sujets et peut également fonctionner pour les conflits d'intérêts et les conflits structurels.

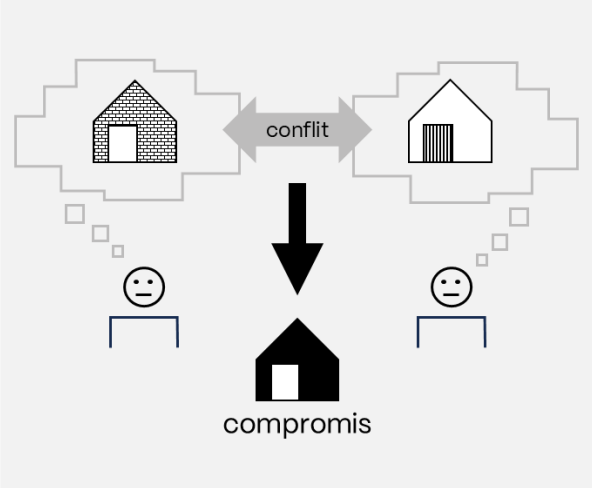


Figure 4.11 Compromis

Vote

Le *vote* fonctionne le mieux lorsqu'il s'agit de faire un choix relativement simple entre un ensemble clair d'exigences contradictoires. Les parties prenantes qui participent au vote (généralement pas seulement les parties en conflit, mais toutes les parties prenantes concernées) doivent bien comprendre les alternatives et les conséquences de leur vote. Afin d'éviter les influences dues à des dépendances ou à un déséquilibre des pouvoirs, il est préférable que le vote se fasse de manière anonyme et avec l'aide d'un modérateur neutre. La procédure de vote elle-même doit faire l'objet d'un accord entre les parties prenantes avant le vote proprement dit. Le vote est un moyen rapide et facile de résoudre les conflits, mais la partie qui perd le vote est déçue et peut avoir besoin d'attention. Le vote peut fonctionner pour la plupart des types de conflits et peut être un bon moyen de résoudre les conflits de sujets et d'intérêts.

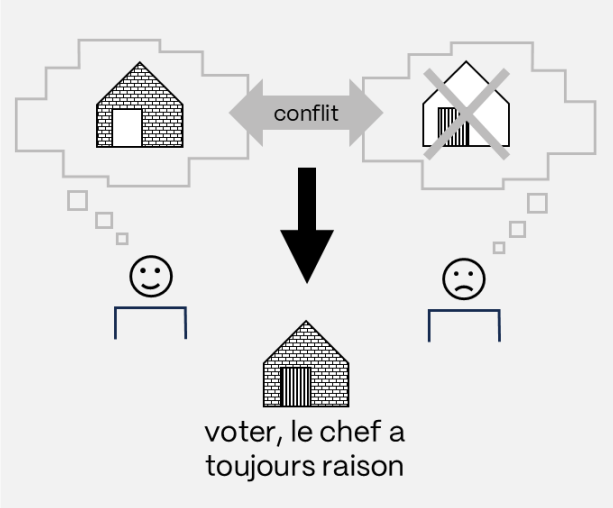


Figure 4.12 Vote

Le chef a toujours raison

Si un accord ou un compromis ne peut être trouvé et qu'au moins une des parties en conflit refuse de participer au vote, il est possible de *passer outre*. Elle est souvent appliquée sous pression, lorsque le temps manque pour utiliser des techniques plus pratiques. En règle générale, on passe outre en transférant le choix entre des exigences contradictoires à un décideur dont l'autorité ou la hiérarchie est supérieure à celle de toutes les parties en conflit et qui dispose d'un pouvoir suffisant pour faire appliquer la décision. C'est donc un bon moyen de résoudre les conflits d'intérêts et les conflits structurels. Dans cette situation, il est particulièrement important que le décideur comprenne parfaitement les alternatives, la position des parties en conflit et les conséquences de la décision. Une variante du "Passer outre" consiste à confier la décision à un tiers, par exemple un expert externe. Dans ce cas, il est important d'obtenir d'abord un accord entre les parties prenantes sur le décideur. Comme pour le vote, vous devrez peut-être prêter attention au *perdant*.

Définition de variantes

La *définition de variantes* est souvent envisagée pour des raisons de conflit de sujets, d'intérêts et de valeurs. Nous avons vu qu'il n'est pas possible de mettre en œuvre des exigences contradictoires dans un seul et même système. La définition de variantes signifie que nous élaborons des solutions distinctes pour toutes les exigences contradictoires. Pour ce faire, on développe généralement un système qui peut être configuré à l'aide de paramètres afin de présenter les caractéristiques souhaitées. Cette solution peut sembler parfaite, mais elle a un prix : il faut beaucoup de temps pour définir la solution et une complexité croissante (ainsi que des coûts supplémentaires) est introduite dans le système, tant pour le développement que pour l'exploitation et la maintenance. Cette technique n'est donc réalisable que si l'on dispose de suffisamment de temps et de budget.

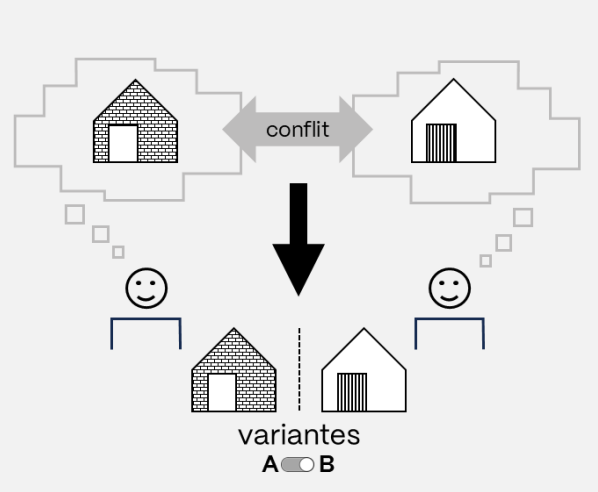


Figure 4.13 Définition de variantes

Techniques auxiliaires

En outre, il existe plusieurs *techniques auxiliaires* qui ne sont généralement pas utilisées seules, mais plutôt pour assister les techniques susmentionnées.

Dans le cadre de *Considérer tous les faits* (CAF), vous envisagez des solutions alternatives en fonction d'un certain nombre de critères prédéfinis – par exemple, le coût, le temps, le risque, les ressources disponibles. L'évaluation de ces critères peut permettre de mieux cerner les avantages et les inconvénients des différentes solutions et d'identifier la *meilleure* d'entre elles.

Points forts/Points faibles (PMI, voir [DeBo2005]) est un outil de brainstorming et de prise de décision. Elle encourage l'examen d'idées et de concepts sous plusieurs angles et est donc précieuse pour la résolution des conflits. Dans le cadre du PMI, les participants (généralement toutes les parties prenantes concernées) identifient d'abord tous les aspects positifs (plus) des alternatives, puis les aspects négatifs (moins), et enfin les points intéressants, c'est-à-dire les éléments qui doivent être examinés plus en détail. La solution qui présente le plus d'avantages et le moins d'inconvénients est la solution préférée.

En fait, le CAF et le PMI sont tous deux des variantes de la *matrice de décision*, une approche *méthodique* de la résolution des conflits. Les exigences contradictoires sont évaluées sur la base d'un nombre (plus important) de critères, après quoi les notes attribuées à ces aspects sont utilisées pour calculer une note finale (pondérée) pour les alternatives. Le score *le plus élevé* l'emporte alors, comme dans l'*alternative 1* de l'exemple de Table 4.1 ci-dessous. En fait, la hiérarchisation (voir la section 6.8) est alors utilisée comme technique de résolution. Comme indiqué précédemment, ces techniques sont généralement considérées comme auxiliaires : elles permettent de mieux comprendre les alternatives et donc de faciliter la technique de résolution choisie. Elles peuvent même être utilisées comme une seule technique si toutes les parties prenantes concernées acceptent le résultat.

Table 4.1 : Exemple de matrice de décision

		Alternative 1 : iOS uniquement		Alternative 2 : Android et iOS	
Critère	Poids	Score	Pondéré	Score	Pondéré
Clients existants	2	3	6	4	8
Coût du développement	1	3	3	2	2
Time to Market	3	4	12	2	6
Réputation	2	2	4	4	8
Experience Utilisateur	1	5	5	3	3
Total			30		27

4.4 Validation des exigences

Au chapitre 2, principe 6, nous avons souligné l'importance de la validation des exigences pour éviter que les parties prenantes ne soient insatisfaites. Étant donné que les exigences constituent la base du développement ultérieur du système, nous devons nous assurer de leur qualité dès le départ afin de réduire les efforts inutiles en aval, tant au niveau des exigences individuelles que des produits d'activités qui les contiennent (Figure 4.14).

Nous devons valider la couverture des besoins des parties prenantes par notre documentation, le degré d'accord entre toutes les parties prenantes et la vraisemblance de nos hypothèses sur le contexte du système avant de transmettre les exigences aux développeurs ou aux fournisseurs. Bien que le niveau de détail puisse varier, cela s'applique aussi bien aux approches de développement itératives que séquentielles.

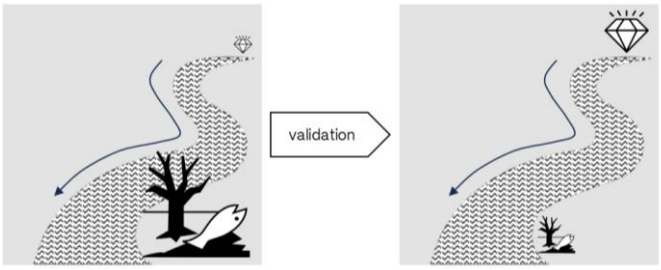


Figure 4.14 La qualité en amont réduit le gaspillage en aval

La validation ajoute du temps et des coûts au projet, de sorte que son efficacité doit être une préoccupation de l'ingénieur des exigences. Il est donc important de surveiller et d'analyser en permanence les défauts qui surviennent au cours du développement et en cours d'exploitation. Si la cause première de ces défauts semble se trouver dans les exigences, le processus de validation des exigences a échoué d'une manière ou d'une autre. C'est pourquoi, en tant qu'ingénieur des exigences, vous devez rechercher en permanence et activement des possibilités de l'améliorer.

4.4.1 Aspects importants de la validation

En ce qui concerne le concept de validation, certains aspects sont importants pour en tirer le meilleur parti (voir également [PoRu2015]) :

- **Impliquer les bonnes parties prenantes**

  En tant qu'ingénieur des exigences, vous devez décider qui vous souhaitez inviter à participer à la validation. À cet égard, un aspect important à considérer est le degré d'indépendance entre les personnes impliquées dans l'élucidation des exigences et celles qui les valident. Un faible niveau d'indépendance (inviter des parties prenantes qui ont déjà participé à l'élucidation) est peu coûteux et facile à organiser, mais il peut faire oublier certains défauts en raison de l'objectif propre, des angles morts, des intérêts conflictuels ou des hypothèses erronées de ces personnes. Un degré plus élevé d'indépendance (par exemple, en invitant des examinateurs ou des auditeurs externes) demande plus de temps et d'efforts pour être organisé et réalisé et entraîne des coûts (initiaux) plus élevés, mais peut à long terme être plus efficace pour trouver des défauts plus nombreux et plus graves. Par conséquent, un risque plus élevé dans la portée du projet et/ou le contexte du système exige un degré d'indépendance plus élevé.

- **Séparer l'identification et la correction des défauts**

  Il peut être tentant de corriger chaque défaut dès qu'il est détecté. Toutefois, cette méthode de travail ne s'avère généralement ni efficace ni efficiente, car les défauts peuvent s'influencer mutuellement. Un défaut découvert ultérieurement au cours de la validation peut invalider la correction d'un défaut antérieur. Une exigence initialement marquée comme défectueuse peut s'avérer correcte lorsque toutes les exigences ont été étudiées. Vous pouvez décider de ne pas corriger certains défauts (mineurs) en raison de l'effort nécessaire par rapport à l'ensemble des défauts constatés. Et après tout, les personnes impliquées dans la validation des exigences devraient se concentrer sur la recherche de défauts et non sur le développement d'idées sur la manière de les corriger. Il est donc recommandé de commencer par sélectionner (un ensemble cohérent) d'exigences à valider et de décider s'il convient ou non de corriger certains défauts constatés seulement après avoir vérifié l'ensemble.

- **Valider selon des points de vue différents**

  Une validation correcte est toujours un effort de groupe, et non une activité réalisée par les ingénieurs des exigences seuls. Les meilleurs résultats sont obtenus lorsque la validation est effectuée par une équipe interdisciplinaire dans laquelle les participants sélectionnés apportent leur propre expertise. En général, on peut dire que l'entrée, la sortie et les pairs doivent être représentés. Dans les projets itératifs, l'équipe agile actuelle est un choix raisonnable, mais le degré d'indépendance peut être faible et des validateurs supplémentaires devraient être invités ; dans les projets séquentiels, une équipe spécifique peut être composée pour chaque effort de validation distinct. En fonction de la phase du projet, il est utile que les entreprises, les utilisateurs, les développeurs, les testeurs, les opérateurs et les gestionnaires d'applications apportent leur contribution ; parfois, des experts en la matière ou des spécialistes sur des sujets tels que la performance, la sécurité et l'utilisabilité peuvent être ajoutés.

- **Répéter la validation**

  Dans les projets séquentiels, la plupart des exigences sont définies et documentées au cours de la phase initiale et validées de manière approfondie à la fin de cette phase. Toutefois, ce n'est pas le seul moment de validation. Pendant le reste du projet, de nouvelles connaissances peuvent conduire à la mise à jour, à l'approfondissement et à l'élargissement de l'ensemble des exigences initiales. Cela pourrait compromettre la qualité, la cohérence et l'uniformité des exigences, ce qui pourrait nécessiter des validations supplémentaires. Elles sont souvent planifiées lors des étapes du projet.
  Dans les projets itératifs, de nombreux rituels agiles incluent des efforts de validation. La planification du sprint, le raffinement du backlog, les revues de sprint et même les réunions quotidiennes sont autant d'occasions de valider et d'améliorer les exigences. Cependant, ces efforts se concentrent souvent sur des exigences individuelles et détaillées et la vue d'ensemble peut être négligée. Une validation initiale de l'ensemble du Product Backlog au début d'un projet ou d'un incrément est un bon début. D'autres initiatives utiles sont des sprints de renforcement répétés et une validation globale supplémentaire au moment de la mise en production.

4.4.2 Techniques de validation

En ce qui concerne les autres techniques, l'ingénieur des exigences peut choisir parmi une large gamme de techniques de validation qui diffèrent en termes de formalité et d'effort. De nombreux facteurs influencent le choix de ces techniques, par exemple le modèle de cycle de vie du développement logiciel, la maturité du processus de développement, la complexité et le niveau de risque du système, les exigences légales ou réglementaires et la nécessité d'un audit.

Souvent, au cours d'un projet, le degré d'effort et de formalité augmente vers la fin, lorsque les décisions finales concernant le système et sa mise en œuvre doivent être prises. Vous constaterez également que la quantité, la valeur et le niveau de détail du retour

d'information des parties prenantes augmentent à mesure que les produits d'activités à valider deviennent plus concrets et plus détaillés. Cela implique l'application de différentes techniques de validation à différents stades du projet. Au début d'un projet, des cycles de validation et de retour d'information fréquents, courts et légers sont préférables, comme c'est habituellement le cas dans les approches agiles. La qualité est ainsi garantie dès le départ. Plus tard dans le projet, des techniques ponctuelles plus formelles et plus longues seront utilisées.

Par ailleurs, vous pouvez également observer un changement dans l'orientation des activités de validation. Dans les premières phases d'un projet, les techniques sont principalement utilisées pour valider la *spécification des* exigences. Dans les phases ultérieures, l'objectif de ces mêmes techniques peut être de valider leur *mise en œuvre*.

En général, nous distinguons trois catégories de techniques de validation (voir Figure 4.15) :

- Techniques de revue
- Techniques exploratoires
- Développement d'échantillons

Les techniques de revue et le développement d'échantillons sont dits statiques, car ils se concentrent sur l'analyse des spécifications d'un système sans l'exécuter. Dans les techniques exploratoires, la validation se concentre sur le comportement réel (ou simulé) du système en fonctionnement ; ces techniques sont dites dynamiques.

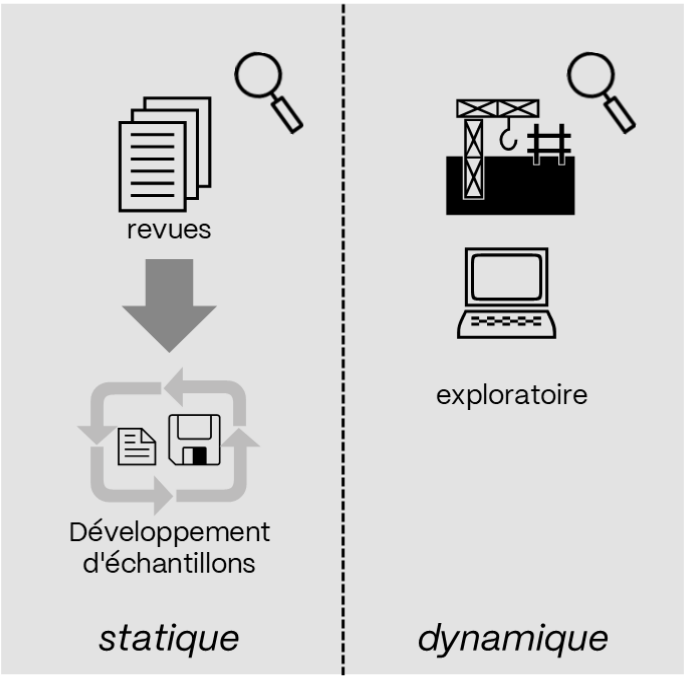


Figure 4.15 Catégories de techniques de validation

La caractéristique commune des *techniques de revue* est qu'elles reposent sur l'étude visuelle des premiers travaux et des travaux intermédiaires. Elles vont de l'informel au très formel et peuvent être appliquées dès le début d'un projet jusqu'à la mise en œuvre du

système. Dans la plupart des cas, la revue des exigences est limité aux premières phases d'un projet. Généralement, lors d'une revue, nous vérifions les produits d'activités *statiques* qui définissent ou décrivent le fonctionnement du système. Pour plus d'informations sur les revues, voir [OleA2018].

Les *évaluations informelles* suivent généralement le cycle auteur-réviseur. Un auteur envoie un travail à un groupe de personnes en leur demandant de le valider. Il s'agit généralement d'un petit groupe de membres de l'équipe, de pairs et/ou d'utilisateurs impliqués dans le projet. Les auteurs peuvent choisir eux-mêmes le groupe ou sa composition peut être prescrite par le règlement de l'entreprise. Après une courte période (qui n'est souvent pas prédéfinie), l'auteur recueille tous les commentaires et les utilise pour mettre à jour le travail en cours. Une bonne pratique consiste à documenter les commentaires dans un registre de revue et à suivre la manière dont ils sont traités. Toutefois, en raison de la nature informelle de ce type de revue, les auteurs sont libres de décider s'ils veulent utiliser les commentaires et de quelle manière. Souvent, la revue est répétée sur plusieurs versions préliminaires jusqu'à ce que l'auteur soit satisfait de la qualité.

Comme elles sont informelles, on peut s'attendre à ce qu'elles soient peu utiles pour valider et améliorer la qualité des exigences. Toutefois, si tous les participants sont attachés à la qualité et s'ils sont capables et désireux de consacrer suffisamment de temps au processus de revue, les revues informelles constituent un moyen de validation facile, peu coûteux et accessible. En fait, cette approche est courante pour les premières versions draft. Pour la version finale d'un produit, une technique plus formelle peut s'avérer plus appropriée.

Les revues formelles suivent une méthode de travail prescrite. Elles sont souvent utilisées pour des travaux importants ou marquants, pour des versions finales et dans des situations où des risques élevés sont en jeu. Bien qu'il existe de nombreux types de revues formelles, ils peuvent être divisés en deux groupes principaux :

- **Relectures techniques**

  Par essence, l'auteur d'un produit d'activités l'explique étape par étape à un public dans le cadre d'une session interactive. Dans la pratique, il existe deux types d'évaluation : (1) les évaluateurs se joignent à la réunion sans aucune préparation et écoutent l'auteur en posant des questions ad hoc ; ou (2) ils obtiennent le produit du travail avant la réunion et préparent des questions à l'intention de l'auteur. Les participants à l'audience peuvent faire des commentaires, identifier les failles et suggérer des alternatives. L'auteur donne plus d'explications si nécessaire et peut discuter des solutions pour les faiblesses identifiées et évaluer les alternatives par rapport aux idées originales. Il y a deux occasions où l'on peut utiliser au mieux les "relectures techniques" : (a) dans une phase initiale du projet pour discuter de la faisabilité d'un certain concept de système ou d'une ébauche de solution ; et (b) lors du transfert d'un produit d'activités intermédiaire à une autre partie qui l'utilisera comme intrant pour un développement ultérieur. Dans les projets itératifs, les

relectures techniques sont surtout présentes sous la forme de sessions régulières d'amélioration avant une itération et de revues de sprint à la fin de celle-ci.

- **Inspections**

  Les *inspections* font partie des techniques de revue les plus formelles. Dans ce cas, la responsabilité de la revue n'incombe pas à l'auteur mais à un responsable de la revue indépendant, souvent appelé *modérateur*. Une inspection est normalement réalisée sous la forme d'une réunion avec le modérateur, l'auteur et un groupe *d'examinateurs*. Les examinateurs sont sélectionnés parmi des pairs, des métiers, des utilisateurs et/ou des experts. Il leur est demandé de vérifier le produit d'activités sur la base de leur expertise spécifique, de s'assurer qu'il est conforme aux standards, normes et réglementations applicables, et de l'évaluer par rapport aux objectifs convenus. Souvent, cette vérification par les examinateurs est effectuée au cours d'une préparation individuelle approfondie avant la réunion proprement dite, guidée par des listes de contrôle détaillées. Lors de la réunion de revue, l'auteur joue le rôle de l'audité, expliquant ce qui n'est pas clair et essayant de comprendre les commentaires des réviseurs et les conséquences pour le produit d'activités. En règle générale, une inspection suit un processus strict et documenté, géré par le modérateur, qui se concentre sur la recherche de défauts et la mesure d'aspects qualitatifs définis, et qui fournit une piste d'audit détaillée. Sous cette forme, les inspections sont souvent utilisées pour décider de la validation d'un produit d'activités pour une prochaine étape du processus de développement, ou même pour la mise en œuvre finale. Les inspections sont principalement appliquées aux systèmes et processus métiers critiques (pour la sécurité). Dans les approches agiles, cette méthode formelle de revue est intégrée à la méthodologie elle-même – par exemple, avec les cérémonies Scrum (raffinement, planification, revue de sprint).

Les *techniques exploratoires* offrent à un groupe de parties prenantes et d'utilisateurs potentiels la possibilité d'acquérir une expérience pratique d'une version intermédiaire (d'une partie) du système en cours de développement. Contrairement aux revues, les techniques exploratoires sont *dynamiques*: elles examinent le comportement (réel ou simulé) du système opérationnel tel qu'il est perçu par les utilisateurs par le biais des interfaces utilisateur. Les participants sont invités à utiliser le système d'une manière similaire à l'utilisation prévue en production. Ils sont relativement libres de le faire, mais certaines orientations sont parfois données. Après une période d'utilisation, les participants font part de leurs expériences et de leurs commentaires sur le comportement actuel du système à l'ingénieur des exigences. Il peut s'agir de défauts constatés et de suggestions d'amélioration.

Les techniques exploratoires sont courantes dans les approches de développement itératives et de design thinking sur la conception. En fait, le développement incrémental habituel, qui commence par la publication d'un *produit minimum viable (MVP),* suivi de l'ajout de fonctionnalités étape par étape, tout en mesurant soigneusement les réactions du marché et en ajustant le système en conséquence, peut être considéré comme une validation exploratoire des exigences en production.

Les techniques exploratoires courantes sont les suivantes :

- **Prototypage**

  Lors de la validation par *prototypage*, une première version spécifique du système est remise à un groupe de parties prenantes pour évaluation. Cette version peut être construite explicitement à des fins de validation, après quoi elle est abandonnée ; nous appelons cela un prototype exploratoire ou jetable. Bien entendu, les prototypes évolutifs, qui sont continuellement mis à jour et étendus jusqu'à ce qu'ils aboutissent au produit final, peuvent également être utilisés pour la validation au cours de leur développement. L'essence de tout prototype est que, de l'extérieur, il ressemble au système prévu, permettant aux parties prenantes d'acquérir une expérience pratique alors que la structure interne peut encore être inachevée, inopérante, voire complètement absente. Lorsque vous utilisez un prototype pour la validation, vous pouvez le faire construire pour vérifier une caractéristique spécifique, telle que l'interface utilisateur, la sécurité ou la performance.

- **Elucidation et validation vont de pair**

  Comme nous l'avons vu à la section `4.2.3`, le prototypage et le storyboard peuvent également être utilisés comme techniques d'élucidation. En fait, ces techniques soutiennent à la fois l'élucidation et la validation, qui vont de pair : lors de la validation des exigences élucidées à un moment antérieur, vous détecterez presque certainement de nouvelles exigences dans le retour d'information des participants. Ces deux aspects du prototypage sont très présents dans les approches de design thinking (voir [LiOg2011]).

- **Tests alpha et bêta**

  Lors des tests alpha et bêta, une version de pré-production du système, dotée de toutes les fonctionnalités et fonctionnant parfaitement, est fournie aux utilisateurs finaux afin qu'ils puissent l'utiliser dans le cadre des processus opérationnels prévus, dans un environnement réaliste.
  Les *tests alpha* sont effectués sur le site du développeur dans un environnement simulé. Le groupe de participants est relativement restreint, des conseils peuvent être donnés et il est possible d'observer l'interaction des utilisateurs avec le système – par exemple, dans un laboratoire d'utilisabilité.
  Les *tests bêta* sont effectués sur les sites des utilisateurs finaux en production réelle (ou dans l'environnement choisi par les utilisateurs finaux). Le système est proposé (le plus souvent gratuitement) à un groupe d'utilisateurs (parfois sélectionnés mais généralement inconnus), avec la demande implicite de valider son apparence et son comportement. Lors d'un test bêta, il est important d'encourager tous les participants à donner leur avis et de leur fournir un moyen facile de le faire. L'analyse de ce retour d'information après une période d'utilisation prolongée peut donner des indications précieuses sur la qualité des exigences. Elle est particulièrement utile pour vérifier certaines hypothèses émises lors de l'élucidation et du développement.

- **Test A/B**

  Les *tests A/B* sont souvent effectués avec une version publiée du système dans un environnement pleinement opérationnel, mais ils peuvent également être appliqués avec des versions préliminaires dans un environnement de test protégé. L'essence du test A/B est que le système est proposé à différents groupes d'utilisateurs (la plupart du temps sélectionnés au hasard) dans deux variantes qui diffèrent en termes de conception ou de fonctionnalité et qui réalisent les objectifs de l'utilisateur d'une manière différente. La réaction des deux groupes est mesurée et comparée ; cette méthode fonctionne mieux lorsque les groupes sont suffisamment importants pour permettre une analyse statistique. L'analyse fournira alors des informations sur la qualité des exigences sous-jacentes et sur la justesse des hypothèses antérieures. Les tests A/B jouent un rôle important dans *le Lean Startup*, l'une des approches de design thinking (voir [Ries2011]).

Dans le cadre d'un *développement d'échantillons*, vous fournissez un ensemble d'exigences aux développeurs, qui tentent de produire des produits intermédiaires communs (par exemple, des conceptions, du code, des cas de test, des manuels) sur la base de ces données. Le système lui-même n'est pas (encore) opérationnel, de sorte que ce type de validation est *statique*, comme dans le cas de la revue. Au cours de cet effort, les développeurs peuvent détecter des défauts tels que des imprécisions, des omissions et des incohérences qui les empêchent de produire le résultat escompté. Bien entendu, ces défauts seront corrigés. En même temps, la quantité et la gravité des défauts détectés sont une indication de la qualité des exigences. Si cette qualité n'est pas suffisante, une validation plus poussée est nécessaire – par exemple, des revues supplémentaires.

Une validation similaire peut être effectuée par les ingénieurs des exigences eux-mêmes. Dans ce cas, vous essayez de documenter un ensemble d'exigences dans une forme de représentation différente du type original : généralement, en convertissant une spécification d'exigences créée en langage naturel en un modèle pertinent, ou un modèle spécifique en une description textuelle. Cet exercice est particulièrement utile pour détecter les omissions. Si vous rencontrez de sérieux problèmes lors de cette conversion, cela indique la nécessité d'une validation supplémentaire.

4.5 Pour en savoir plus

Glinz et Wieringa [GlWi2007] expliquent la notion et l'importance des parties prenantes. Alexander [Alex2005] explique comment classer les parties prenantes. Bourne [Bour2009] traite de la gestion des parties prenantes. Lim, Quercia et Finkelstein [LiQF2010] étudient l'utilisation des réseaux sociaux pour l'analyse des parties prenantes. Humphrey [Hump2017] parle des profils d'utilisateurs.

Zowghi et Coulin [ZoCo2005] présentent une vue d'ensemble des techniques d'élucidation des exigences. Gottesdiener [Gott2002] a écrit un manuel classique sur les ateliers dans l'ingénierie des exigences. Carrizo, Dieste et Juristo [CaDJ2014] étudient la sélection des techniques d'élucidation adéquates.

Maalej, Nayebi, Johann et Ruhe [MNJR2016] examinent l'utilisation du retour d'information explicite et implicite de l'utilisateur pour la définition des exigences. Maiden, Gitzikis et Robertson [MaGR2004] examinent comment la créativité peut favoriser l'innovation dans l'IE.

Le livre de Moore [Moor2014] est un classique de la gestion des conflits. Glasl [Glas1999] discute de la manière de gérer les conflits. Grünbacher et Seyff [GrSe2005] expliquent comment parvenir à un accord en négociant sur les exigences lors de la validation des exigences ou de la résolution des conflits.

La validation est abordée dans tous les manuels d'IE ; voir [Pohl2010], par exemple.

5 Processus et structure de travail

Chaque fois qu'un travail doit être effectué de manière systématique, un *processus* est nécessaire pour façonner et structurer la méthode de travail et la création des produits d'activités.

> Définition 5.1. Processus *(Process)* :
> A set of interrelated activities performed in a given order to process information or materials.

Un processus d'ingénierie des exigences (IE) organise la manière dont les tâches d'IE sont exécutées en utilisant les pratiques appropriées et en produisant les produits d'activités requis. Toutefois, il n'existe pas de processus d'IE unique et éprouvé (voir la section 1.4). Par conséquent, les ingénieurs en exigences doivent configurer un processus d'IE sur mesure qui s'adapte à la situation donnée.

Le processus d'IE façonne le flux d'informations et le modèle de communication entre les participants impliqués dans l'IE (par exemple, les clients, les utilisateurs, les ingénieurs en exigences, les développeurs et les testeurs). Il définit également les produits d'activités de l'IE à utiliser ou à produire. Un processus d'IE approprié fournit le cadre dans lequel les ingénieurs des exigences peuvent obtenir, documenter, valider et gérer les exigences.

Dans ce chapitre, vous découvrirez les facteurs qui influencent le processus d'IE et vous apprendrez à configurer un processus approprié à partir d'un ensemble de facettes de processus.

5.1 Facteurs d'influence

Plusieurs facteurs d'influence doivent être pris en compte lors de la configuration d'un processus d'IE. Avant de commencer la configuration d'un processus d'IE, ces facteurs doivent être étudiés et analysés.

D'une part, cette analyse fournit des informations sur la manière de configurer le processus d'IE. Par exemple, lorsque l'analyse indique que les parties prenantes n'ont qu'une vague idée de leurs besoins, il convient de choisir un processus d'IE qui favorise l'exploration des besoins. D'autre part, les facteurs d'influence limitent l'espace des configurations de processus possibles. Par exemple, si les parties prenantes ne sont disponibles qu'au début du projet de développement de système, un processus qui s'appuie sur un retour d'information continu de leur part ne serait pas adapté. Ci-dessous, nous abordons les facteurs importants pour le processus d'IE.

Adéquation global du processus. Lors de la définition ou de la configuration d'un processus d'IE, il est essentiel de connaître et de comprendre le processus de développement global choisi pour le système à développer – définir un processus d'IE qui ne correspond pas au

processus global n'a pas de sens. Le processus global peut nécessiter des produits d'activités que le processus d'IE doit fournir. La terminologie utilisée pour le processus d'IE doit être alignée sur la terminologie du processus global. En particulier, la terminologie des produits d'activités doit être harmonisée. Cela permet d'éviter les confusions et les malentendus. Il facilite également l'introduction du processus d'IE ainsi que la formation et l'accompagnement des personnes qui doivent travailler selon ce processus. Par exemple, si le système est développé selon un processus linéaire et planifié qui repose sur l'existence d'une spécification complète des exigences du système et d'un glossaire du système à la fin de la phase des exigences, le processus d'IE choisi doit s'intégrer dans la phase des exigences du processus global et produire les deux produits d'activités requis.

Contexte du développement. Le contexte du développement informe également le processus d'IE. Les éléments à prendre en compte sont la relation client-fournisseur-utilisateur, le type de développement, les questions contractuelles et la confiance. Lors de l'analyse du contexte du développement, il convient de répondre à un certain nombre de questions :

- Relation client-fournisseur-utilisateur : Existe-t-il un client désigné qui commande le système et le paie et un fournisseur qui développe le système ? Le client et le fournisseur font-ils partie de la même organisation ou appartiennent-ils à des organisations différentes ? Dans le premier cas, quelles sont les personnes qui jouent le rôle de client et quelles sont celles qui jouent le rôle de fournisseur ? Qui sont les utilisateurs du système ? Les utilisateurs appartiennent-ils à l'organisation du client ?

  Si ce n'est pas le cas, utilisent-ils le système comme un produit ou un service pour interagir avec le client (par exemple, dans le commerce électronique) ou achètent-ils le système comme un produit ou un service au client (par exemple, une application mobile) ?

- Type de développement : Quel est le cadre organisationnel pour le développement d'un système ? Les types typiques sont les suivants :
  - Un fournisseur spécifie et développe un système pour un client spécifique qui l'utilisera.
  - Une organisation développe un système dans l'intention de le vendre en tant que produit ou service à de nombreux clients dans un certain segment de marché.
  - Un fournisseur configure un système pour un client à partir d'un ensemble de composants prêts à l'emploi.
  - Un fournisseur améliore et fait évoluer un produit existant.

- *Contrat*: existe-t-il un contrat ou un accord similaire définissant formellement les prestations, les coûts, les délais, les responsabilités, etc. ? Les contrats peuvent être des contrats classiques à prix fixe entre un client et un fournisseur, avec des fonctionnalités, des délais et des coûts fixes, ou peuvent simplement fournir un cadre financier, tandis que les fonctionnalités sont définies de manière itérative.

- *La confiance*: Les parties concernées se font-elles confiance ? Si, par exemple, le client et le fournisseur ne se font pas confiance, les exigences doivent être spécifiées de manière plus détaillée que ce qui serait nécessaire dans une relation de confiance.

Disponibilité et capacité des parties prenantes. La disponibilité des parties prenantes limite les options de configuration du processus d'IE. Par exemple, un processus nécessitant une interaction étroite et continue avec les parties prenantes ne peut pas être choisi si les principales parties prenantes ne sont disponibles que pendant une courte période au début du processus.

La capacité des parties prenantes influence également le processus : moins les parties prenantes sont en mesure d'exprimer clairement leurs besoins et moins elles connaissent leurs besoins réels, plus le processus d'IE doit tenir compte de l'exploration des besoins.

Compréhension commune. L'ingénierie des exigences n'est que peu nécessaire lorsque les parties prenantes, les ingénieurs des exigences, les concepteurs et les développeurs ont une vision commune du problème et des exigences (voir chapitre 2, principe 3). Par conséquent, plus la compréhension commune est bonne, plus le processus d'IE peut être léger [GlFr2015].

Complexité et criticité. Le degré de détail avec lequel les exigences doivent être spécifiées dépend fortement de la complexité et de la criticité du système à développer. Lorsqu'un système est complexe et/ou critique en termes de sûreté ou de sécurité, le processus d'IE choisi doit permettre une spécification détaillée des exigences critiques, y compris des modèles formels ou semi-formels et une validation solide – par exemple, en vérifiant les modèles qui expriment le comportement prescrit ou en construisant des prototypes.

Contraintes. De toute évidence, tous les facteurs d'influence limitent l'espace des configurations possibles d'un processus d'IE. Lorsque nous parlons de contraintes, nous entendons celles qui sont explicitement imposées, par exemple, par le client ou un organisme de réglementation. Ces contraintes peuvent impliquer la création obligatoire de certains produits d'activités et le respect d'un processus obligatoire pour la production de ces produits d'activités. Les clients ou les régulateurs peuvent également exiger un processus d'IE conforme à une norme donnée.

Temps et budget disponibles. Si les calendriers et les budgets sont serrés, le temps et le budget disponibles pour l'IE doivent être utilisés à bon escient, ce qui implique généralement le choix d'un processus d'IE léger. Le choix d'un processus d'IE itératif permet de hiérarchiser les exigences et de mettre en œuvre les plus importantes dans les limites du budget et du calendrier impartis.

Volatilité des exigences. Si de nombreuses exigences sont susceptibles de changer, il est conseillé d'opter pour un processus d'IE itératif et adapté aux changements.

Expérience des ingénieurs des exigences. Le processus d'IE choisi doit correspondre aux compétences et à l'expérience des ingénieurs des exigences impliqués. Dans le cas contraire, du temps et un budget supplémentaires doivent être consacrés à la formation et à l'accompagnement du processus choisi. Il est préférable de choisir un processus assez simple que les ingénieurs des exigences peuvent gérer correctement plutôt qu'un processus sophistiqué et compliqué qui les surcharge.

5.2 Facettes du processus d'ingénierie des exigences

Définir le processus d'IE à partir de zéro pour chaque projet d'IE est un gaspillage d'efforts. Lorsque les facteurs d'influence le permettent, le processus doit être configuré à partir d'éléments préexistants. Afin de fournir des indications sur la manière de configurer un processus d'IE approprié, nous décrivons trois facettes avec deux instances chacune, ainsi que les critères de sélection à prendre en compte pour chaque instance [Glin2019]. Plus loin, dans la section 5.3, nous utilisons ces facettes pour configurer les processus d'IE. Figure 5.1 présente une vue d'ensemble des facettes et des instances.

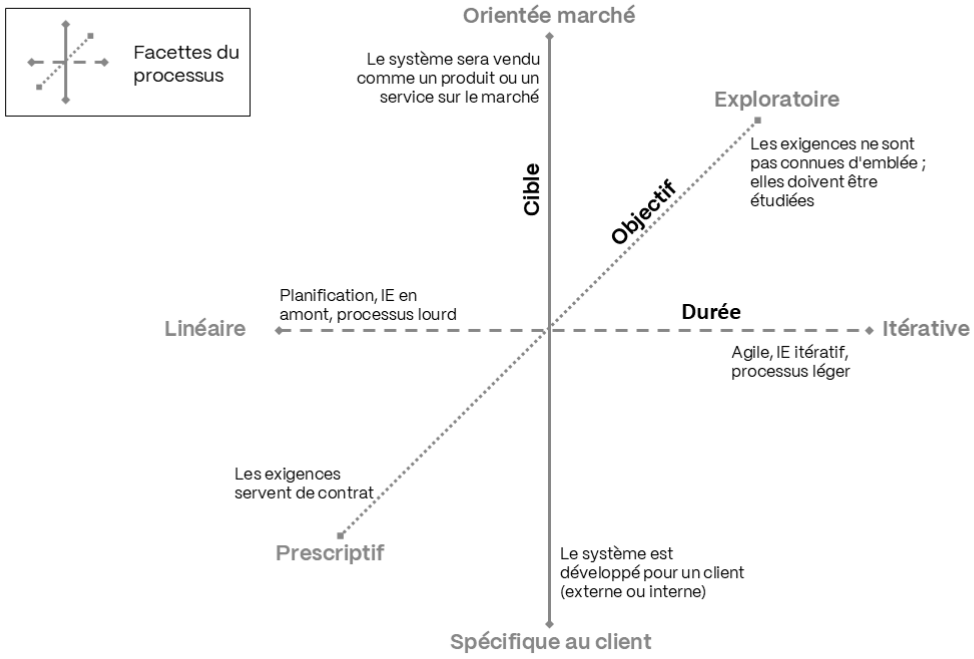


Figure 5.1 Facettes du processus d'IE

Les facettes peuvent être considérées comme couvrant un espace tridimensionnel d'options de configuration du processus. Chaque instance de facette est assortie de critères de sélection.

L'applicabilité de ces critères découle de l'analyse des facteurs d'influence examinés à la section 5.1 ci-dessus. Notez qu'il n'est pas nécessaire que tous les critères soient remplis pour choisir une instance d'une facette.

5.2.1 Facette temporelle : linéaire versus itérative

La facette temporelle traite de l'organisation des activités de l'IE sur une échelle temporelle. Nous distinguons les processus linéaires et itératifs.

Dans un *processus d'IE linéaire*, les exigences sont spécifiées à l'avance dans une seule phase du processus. L'idée est de produire une spécification complète des exigences qui ne

nécessite pas ou peu d'adaptation ou de changements au cours de la conception et de l'implémentation du système. La création d'un cahier des charges complet dès le départ nécessite un processus complet. Ainsi, dans la plupart des cas, les processus d'IE linéaires sont des processus lourds.

Critères de choix d'un processus d'IE linéaire :

- Le processus de développement du système est planifié et essentiellement linéaire.
- Les parties prenantes sont disponibles, connaissent leurs exigences et peuvent les spécifier dès le départ.
- Une spécification des exigences complète est nécessaire comme base contractuelle pour l'externalisation ou l'appel d'offres de la conception et de l'implémentation du système.
- Les autorités réglementaires exigent une spécification complète et formellement publiée des exigences à un stade précoce du développement.

Dans un *processus d'IE itératif*, les exigences sont spécifiées de façon incrémentale, en commençant par les objectifs généraux et certaines exigences initiales, puis en ajoutant ou en modifiant des exigences à chaque itération. L'idée est de mêler la spécification des exigences à la conception et à l'implémentation du système. En raison de la brièveté des boucles de rétroaction et de la capacité à prendre en compte les changements ou les éléments oubliés dans les itérations ultérieures, les processus itératifs d'IE peuvent être des processus légers.

Critères de choix d'un processus d'IE itératif :

- Le processus de développement du système est itératif et agile.
- De nombreuses exigences ne sont pas connues à l'avance, mais elles apparaîtront et évolueront au cours du développement du système.
- Les parties prenantes sont disponibles de sorte que de courtes boucles de rétroaction peuvent être établies comme moyen d'atténuer le risque de développer le mauvais système.
- La durée du développement permet de dépasser une ou deux itérations.
- La capacité à modifier facilement les exigences est importante.

5.2.2 Facette de l'objectif : prescriptif versus exploratoire

La facette objectif traite de l'objectif et du rôle des exigences dans le développement d'un système. Nous distinguons les processus d'IE prescriptifs et exploratoires.

Dans un *processus d'IE prescriptif*, la spécification des exigences constitue un contrat : toutes les exigences sont obligatoires et doivent être mises en œuvre. L'idée est de créer une spécification des besoins qui peut être mise en œuvre avec peu ou pas d'interaction entre les parties prenantes et les développeurs.

Critères de choix d'un processus d'IE prescriptif :

- Le client exige un contrat fixe pour le développement du système, souvent avec des fonctionnalités, un champ d'application, un prix et un délai fixes.
- La fonctionnalité et la portée ont la priorité sur le coût et les délais.
- Le développement du système spécifié peut faire l'objet d'un appel d'offres ou être sous-traité.

Dans un *processus d'IE exploratoire*, seuls les objectifs sont connus a priori, tandis que les exigences concrètes doivent être élucidées. L'idée est que les exigences ne sont souvent pas connues a priori mais doivent être explorées.

Critères de choix d'un processus d'IE exploratoire :

- Les parties prenantes n'ont au départ qu'une vague idée de leurs exigences.
- Les parties prenantes sont fortement impliquées et fournissent un retour d'information continu.
- Les délais et le coût ont la priorité sur la fonctionnalité et la portée.
- Le client se contente d'un contrat-cadre portant sur les objectifs, les ressources et le prix à payer pour une période donnée ou un certain nombre d'itérations.
- Il n'est pas clair a priori quelles exigences seront effectivement implémentées et dans quel ordre elles le seront.

5.2.3 Facette cible : spécifique au client versus orientée marché

La facette cible prend en compte le type de développement : quel type de développement visons-nous avec le processus d'IE ? Au niveau élémentaire, nous distinguons les processus d'IE spécifiques au client et les processus d'IE orientés vers le marché.

Dans un *processus d'IE spécifique* au client, le système est commandé par un client et développé par un fournisseur pour ce client. Notez que le fournisseur et le client peuvent faire partie de la même organisation. L'idée est que le processus d'IE reflète la relation client-fournisseur.

Critères de choix d'un processus d'IE spécifique au client :

- Le système sera principalement utilisé par l'organisation qui l'a commandé et qui paie pour son développement.
- Les parties prenantes importantes sont principalement associées à l'organisation du client.
- Des personnes individuelles peuvent être identifiées pour les rôles de parties prenantes.
- Le client veut une spécification des exigences qui puisse servir de contrat.

Dans un *processus d'IE orienté marché*, le système est développé comme un produit ou un service pour un marché, ciblant des segments d'utilisateurs spécifiques. L'idée est que l'organisation qui développe le système dirige également le processus d'IE.

Critères de choix d'un processus d'IE orienté marché :

- L'organisation à l'initiative du développement ou l'un de ses clients a l'intention de vendre le système dans un segment de marché quelconque en tant que produit ou service.
- Les utilisateurs potentiels ne sont pas identifiables individuellement.
- Les ingénieurs des exigences doivent concevoir les exigences de manière à ce qu'elles correspondent aux besoins prévus des utilisateurs ciblés.
- Les Product Owners, les responsables du marketing, les concepteurs numériques et les architectes systèmes sont les principales parties prenantes.

5.2.4 Conseils et mises en garde

Il est important de noter que les critères indiqués ci-dessus sont des critères heuristiques. Ils ne doivent pas être considérés comme un ensemble de règles fixes toujours applicables. Par exemple, l'externalisation du développement du système se fait de préférence avec un processus d'IE prescriptif plutôt qu'avec un processus exploratoire. En effet, le contrat entre le client et le fournisseur est généralement basé sur une spécification complète des exigences. Cependant, il est également possible de négocier un contrat d'externalisation basé sur un processus d'IE exploratoire.

Il peut y avoir des conditions préalables au choix de certaines facettes du processus ou le choix peut entraîner des conséquences qui doivent être prises en compte. Voici quelques exemples :

- Les processus linéaires d'IE ne fonctionnent que si un processus sophistiqué de changement des exigences est en place.
- Les processus d'IE linéaires impliquent de longues boucles de rétroaction : il peut s'écouler des mois, voire des années, entre la rédaction d'une exigence et l'observation de ses effets dans le système implémenté. Pour réduire le risque de développer le mauvais système, les exigences doivent être validées de manière intensive quand on utilise un processus d'IE linéaire.
- Dans un processus orienté marché, le retour potentiel d'information des utilisateurs est le seul moyen de valider si le produit répondra réellement aux besoins du segment d'utilisateurs ciblé.
- Dans un contexte agile, un processus d'IE itératif et exploratoire est le mieux adapté. Les itérations ont une durée fixe (généralement de 2 à 6 semaines). Le Product Owner joue un rôle central dans le processus d'IE, en coordonnant les parties prenantes, en organisant les produits d'activités d'IE et en communiquant les exigences à l'équipe de développement.

Les trois facettes mentionnées ci-dessus ne sont pas totalement indépendantes : le choix effectué pour une facette peut influencer ce qui peut ou doit être choisi pour les autres facettes.

Voici quelques exemples :

- Linéaire et prescriptif sont souvent choisis ensemble, ce qui signifie que lorsque les ingénieurs des exigences optent pour un processus d'IE linéaire, ils optent généralement pour un processus qui est à la fois linéaire et prescriptif.
- Les processus d'IE exploratoires sont généralement aussi des processus itératifs (et vice versa).
- Le processus d'IE orienté marché ne se combine pas bien avec les facettes linéaires et prescriptives.

5.2.5 Autres considérations

La mesure dans laquelle un processus d'IE doit être mis en place et suivi, ainsi que le volume des exigences à produire dans le cadre de ce processus, dépendent du degré de compréhension commune et de la criticité du système.

Plus la compréhension commune est bonne et plus la criticité est faible, plus le processus d'IE peut être simple et léger.

Lorsque le temps et le budget disponibles pour l'IE sont limités, les ressources disponibles doivent être utilisées avec précaution. Le choix d'un processus itératif et exploratoire est utile. En outre, le processus doit se concentrer sur l'identification et le traitement des exigences qui sont essentielles au succès du système.

Enfin, le processus d'IE doit correspondre à l'expérience des ingénieurs des exigences. Plus les compétences et l'expérience de ces personnes sont faibles, plus le processus d'IE doit être simplifié – il n'est pas judicieux de définir un processus sophistiqué si les personnes impliquées ne sont pas en mesure de le mettre en œuvre correctement.

5.3 Configuration d'un processus d'ingénierie des exigences

Dans un contexte concret de développement de système, les ingénieurs des exigences ou les personnes responsables de l'IE doivent configurer le processus d'IE à appliquer. Nous recommandons d'analyser d'abord les facteurs d'influence (voir section 5.1), puis de sélectionner une combinaison appropriée des facettes du processus décrites à la section 5.2.

5.3.1 Combinaisons typiques de facettes

Trois combinaisons de facettes (ou leurs variantes) se rencontrent fréquemment dans la pratique [Glin2019]. Dans ce qui suit, nous décrivons brièvement chacune d'entre elles et les caractérisons en termes de cas d'application principal, de produits d'activités typiques et de flux d'informations typiques. En outre, nous fournissons un exemple. Figure 5.2 montre les trois configurations typiques du processus dans l'espace des trois facettes.

Processus d'IE participatif : Itératif, Exploratoire et Spécifique au client

Un processus d'IE participatif est généralement choisi dans un contexte agile lorsqu'un client commande un système et qu'une équipe de développement le conçoit et le met en œuvre. L'accent est mis sur l'exploration des exigences dans une série d'itérations en étroite collaboration entre les parties prenantes du côté du client, les ingénieurs des exigences et l'équipe de développement.

Principal cas d'application :	Le fournisseur et le client collaborent étroitement ; les parties prenantes sont fortement impliquées à la fois dans les processus d'IE et de développement.
Produits d'activités typiques :	Produits d'activités typiques : Product Backlog avec des User Stories et/ou des descriptions de tâches, vision, prototypes
Flux d'information typique :	Interaction continue entre les parties prenantes, les Product Owners, les ingénieurs des exigences et les développeurs

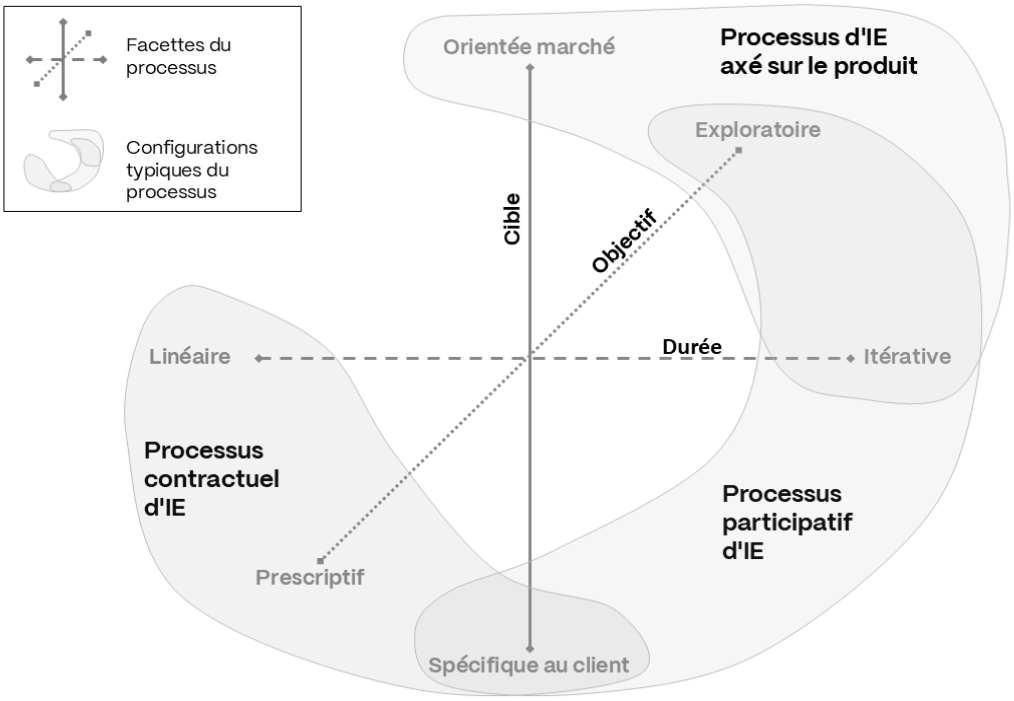


Figure 5.2 Trois configurations typiques du processus d'IE et leur relation avec les trois facettes

Exemple : Dans une compagnie d'assurance, l'unité commerciale qui vend des assurances aux petites et moyennes entreprises a l'idée d'un nouveau produit permettant d'assurer les clients contre les dommages causés par une attaque de pirates informatiques. Ils demandent à l'unité informatique de l'entreprise de former une équipe de développement chargée de concevoir et de développer une nouvelle application capable de gérer le nouveau produit d'assurance dans le cadre du système existant d'aide à la vente de produits d'assurance. Le système de gestion des contrats d'assurance existant doit également être adapté en conséquence. Au-delà de quelques exigences initiales, l'unité commerciale contractante n'a pas d'idée précise sur l'aspect du nouveau produit et sur la manière dont il doit être pris en charge par les systèmes informatiques de l'entreprise. Les services informatiques de l'entreprise ont adopté le développement agile pour tous leurs projets il y a quelques années.

Dans cette situation, un processus d'IE participatif est approprié. Il s'inscrit dans le processus agile global que le service informatique de l'entreprise emploiera pour développer le nouveau système et adapter les systèmes existants. Les parties prenantes de l'unité commerciale et les ingénieurs des exigences de l'informatique de l'entreprise peuvent définir ensemble les exigences relatives au nouveau produit d'assurance. Le processus étant itératif, l'équipe de développement peut mettre au point un prototype de produit commercialisable minimum (MMP) qui aide la direction de l'unité commerciale à décider d'inclure ou non le produit envisagé dans son portefeuille ou d'écarter l'idée. Il existe une relation client-fournisseur claire entre l'unité opérationnelle et le service informatique de l'entreprise, de sorte qu'un processus d'IE orienté vers le client est tout à fait approprié.

Processus contractuel d'IE : Généralement linéaire, prescriptif et spécifique au client

Un processus contractuel d'IE est généralement choisi lorsque le développement d'un système fait l'objet d'un appel d'offres et est externalisé auprès d'un fournisseur dont le contrat est basé sur une spécification des exigences complète. Il s'agit également d'un processus adapté à l'IE dans le cadre de grands projets de développement de systèmes qui appliquent un processus de développement de type "waterfall" (cascade).

Principal cas d'application :	La spécification des exigences constitue la base contractuelle pour le développement d'un système par des personnes non impliquées dans la spécification et avec peu d'interaction avec les parties prenantes après la phase des exigences.
Produits d'activités typiques :	Spécification classique des exigences du système, consistant en des exigences textuelles et des modèles
Flux d'information typique :	Principalement des parties prenantes vers les ingénieurs des exigences

Exemple : Un constructeur automobile développe une nouvelle plate-forme automobile, à partir de laquelle une famille de modèles sera dérivée. L'une des principales décisions de conception de la nouvelle plateforme consiste à se débarrasser des dizaines d'unités de contrôle électronique (ECU) actuellement utilisées dans les voitures et à les remplacer par un seul ordinateur de contrôle qui gère un ensemble d'applications de contrôle de la conduite et d'assistance à la conduite. L'objectif est de réduire les coûts de matériel, de se débarrasser des interactions indésirables entre les calculateurs et de réduire le temps et les efforts consacrés à la mise à jour du logiciel. Les ingénieurs responsables des systèmes électroniques de la nouvelle plate-forme ont rédigé un cahier des charges. L'entreprise a demandé à un grand fabricant de systèmes de contrôle automobile de créer une spécification des exigences du système pour le nouveau système de contrôle centralisé de la voiture. Par la suite, le constructeur automobile lancera un appel d'offres pour la conception et la mise en œuvre du système sur la base de cette spécification. Le constructeur exigera que l'implémentation soit effectuée en plusieurs itérations afin de faciliter les essais et l'intégration du système dans la nouvelle plate-forme automobile.

Dans ce cas, un processus contractuel d'IE est approprié. Le processus global est linéaire : le système ne sera conçu et implémenté qu'une fois la spécification des besoins achevée. Le fait que la mise en œuvre soit itérative n'a pas d'incidence sur le processus d'IE. En fonction de la qualité du cahier des charges existant et de la disponibilité des parties prenantes chez le constructeur automobile, il convient d'opter pour un processus d'IE linéaire ou itératif.

Il est évident qu'un processus d'IE orienté vers le client est nécessaire. L'existence d'une spécification des exigences du client et le fait que la spécification des exigences du système sera utilisée pour soumissionner la conception et l'implémentation du système exigent un processus d'IE normatif.

Processus d'IE orienté produit : Itératif, Exploratoire & Orienté marché

Un processus d'IE axé sur le produit est généralement choisi lorsqu'une organisation développe un système en tant que produit ou service destiné au marché. Dans la plupart des cas, un processus d'IE orienté produit s'accompagne d'un processus de développement de produit agile. Le Product Owner et les concepteurs numériques jouent un rôle majeur dans ce processus : ils influencent et façonnent fortement le produit.

Principal cas d'application :	Une organisation spécifie et développe un logiciel afin de le vendre ou de le distribuer en tant que produit ou service
Produits d'activités typiques :	Product Backlog avec des User Stories et/ou des descriptions de tâches, vision, prototypes, retour d'information utilisateur
Flux d'information typique :	Interaction entre le Product Owner, le marketing, les ingénieurs des exigences, les concepteurs numériques, les développeurs ainsi que le retour d'information des clients/utilisateurs

Exemple : Une entreprise de médias charge son service informatique interne de renouveler entièrement l'application mobile d'actualités qu'elle vend à ses abonnés (une partie du contenu restant en libre accès). À partir des commentaires des utilisateurs, l'entreprise tient un long registre des critiques des clients et des suggestions d'amélioration.

En particulier, de nombreux utilisateurs reprochent à l'application existante de ne pas être assez réactive, d'offrir une mauvaise assistance pour signaler les problèmes et les suggestions, et de ne pas permettre de zoomer à deux doigts sur le texte ou les images. Le service marketing de l'entreprise estime également que la présentation de l'application est dépassée. Ils prévoient qu'avec une nouvelle présentation, ils pourraient gagner davantage d'abonnés. Le PDG de l'entreprise a décidé que le département informatique collaborerait avec une agence de design externe pour l'aspect visuel de l'application. La direction de l'entreprise de médias souhaite une version minimale du produit comme preuve de concept, puis de nouvelles versions intermédiaires toutes les trois semaines, qui pourront être examinées par le service marketing et le conseil d'administration de l'entreprise.

Dans cette situation, un processus d'IE orienté produit est le mieux adapté . Bien qu'il existe une relation client-fournisseur entre la direction de l'entreprise et son service informatique interne, l'accent est clairement mis sur la création d'un produit renouvelé dans le segment des applications mobiles d'actualités. Le processus d'IE doit être exploratoire, car les exigences au-delà des informations contenues dans le registre existant du retour d'information des utilisateurs ne sont pas claires. Le processus de développement global doit être itératif selon la décision de la direction de l'entreprise. Comme les exigences doivent être explorées, un processus d'IE itératif est le mieux adapté ici.

5.3.2 Autres processus d'IE

Les trois combinaisons décrites ci-dessus couvrent un grand nombre de situations qui se présentent dans la pratique. Cependant, il peut y avoir des situations où aucune des configurations de processus susmentionnées ne convient. Par exemple, des contraintes réglementaires peuvent imposer l'utilisation d'un processus conforme à des normes données telles que la norme ISO/IEC/IEEE 29148 [ISO29148]. Dans ce cas, le processus d'IE doit être créé de toutes pièces par des experts en processus ou l'une des configurations susmentionnées doit être adaptée à la situation donnée.

5.3.3 Comment configurer les Processus d'IE

Nous recommandons une procédure en cinq étapes pour configurer un processus d'IE.

1. *Analyser les facteurs d'influence.* Analysez votre situation par rapport à la liste des facteurs d'influence de la section 5.1.
2. *Évaluer les critères de facette.* Sur la base de l'analyse de l'étape 1, parcourez la liste des critères de sélection des facettes figurant à la section 5.2. Vous pouvez attribuer à chaque critère une valeur sur une échelle de cinq points (−−, −, 0, +, ++).

3. *Configurer*. Si l'analyse des critères donne un résultat clair en ce qui concerne les trois configurations typiques mentionnées ci-dessus, choisissez cette configuration. Dans le cas contraire, il convient d'opter pour une adaptation différente du processus, guidée par l'objectif général d'atténuer le risque de développer le mauvais système. Par exemple, imaginons une situation où le client exige qu'une spécification des exigences du système soit créée dès le départ, ce qui nécessite un processus d'IE linéaire et prescriptif. Cependant, lors de vos premières réunions avec le client, vous avez remarqué que pour un sous-système important, le client n'a pas d'idée précise sur ce qu'il doit construire, ce qui nécessite un processus d'IE exploratoire. Une solution potentielle pourrait consister à choisir un processus d'IE contractuel comme cadre général du processus d'IE, mais à créer un sous-projet qui recueille progressivement les exigences de ce sous-système important, en créant des prototypes en deux ou trois itérations (guidées par un sous-processus d'IE participatif), puis à intégrer les résultats dans la spécification des exigences du système.
4. *Déterminer les produits d'activités*. Sur la base de votre analyse et de la configuration du processus, définissez les principaux produits d'activités de l'IE qui seront produits. Veillez à ce que les produits d'activités de l'IE soient alignés sur les produits d'activités du processus de développement global.
5. *Sélectionner les pratiques appropriées*. Pour les tâches à accomplir – par exemple, l'obtention d'exigences – sélectionnez les pratiques qui conviennent le mieux à la situation donnée. Un grand nombre de ces pratiques, ainsi que des conseils sur le lieu et le moment de leur application, sont présentés dans les chapitres 2, 4 et 6 du présent manuel.

Il n'existe pas de processus d'IE unique et éprouvé. Sur la base d'une analyse des facteurs d'influence, un processus d'IE spécifique doit être adapté à chaque entreprise d'IE. La configuration d'un processus d'IE à partir d'un ensemble de facettes de processus constitue un moyen simple de personnalisation.

5.4 Pour en savoir plus

Armour [Armo2004] et Reinertsen [Rein1997], [Rein2009] proposent des réflexions générales sur les processus et les flux d'informations dans les processus.

Bien que le manuel de Robertson et Robertson [RoRo2012] soit intitulé "Mastering the Requirements Process", il s'agit d'un manuel général sur tous les aspects de l'IE.

Wiegers et Beatty [WiBe2013] consacrent un chapitre à l'amélioration des processus d'IE. L'ouvrage de Sommerville et Sawyer [SoSa1998] contient un recueil de bonnes pratiques à utiliser dans le cadre des processus d'IE.

6 Pratiques de gestion des exigences

Les exigences ne sont pas gravées dans la pierre, éternellement présentes du passé au futur ; elles sont vivantes ! Elles naissent grâce à l'élucidation, grandissent grâce à la documentation et sont façonnées grâce à la validation. À l'âge adulte, elles travaillent dans le cadre de l'implémentation et, après une vie d'exploitation que l'on espère longue et prospère, elles prennent leur retraite dans l'oubli. Tout au long de leur cycle de vie, leurs parents, les ingénieurs des exigences, s'occupent d'elles. Nous les soignons dans leur enfance, nous les éduquons dans leur jeunesse, nous les accompagnons dans leurs relations et nous les aidons à trouver un bon emploi dans un système sain. C'est ce que nous appelons la gestion des exigences.

Bien entendu, il existe de meilleures définitions, plus formelles, de la gestion des exigences. La norme ISO/IEC/IEEE 29148:2018 [ISO29148] définit la gestion des exigences comme*"les activités qui identifient, documentent, maintiennent, communiquent, tracent et suivent les exigences tout au long du cycle de vie d'un système, d'un produit ou d'un service*.". Dans le glossaire du CPRE [Glin2025], la gestion des exigences est définie comme "*le processus de gestion des exigences existantes et des produits d'activités liés aux exigences, y compris le stockage, la modification et la traçabilité des exigences.*". Le glossaire du CPRE nous indique également que la *gestion des exigences* fait partie intégrante de l'ingénierie des exigences : "*L'approche systématique et disciplinée de la spécification et de la gestion des exigences dans le but de ...".*

La gestion des exigences peut se faire à différents niveaux :

- Les exigences individuelles
- Les produits d'activités qui contiennent ces exigences
- Le système lié aux produits d'activités et aux exigences qu'ils contiennent

Dans la pratique, la gestion des exigences est principalement effectuée au niveau du produit d'activités. Généralement, un produit d'activités contient plusieurs exigences individuelles (par exemple, une description d'interface externe), tandis que d'autres produits d'activités ne contiennent qu'une seule exigence (par exemple, une seule user story dans un projet agile) ou représentent l'ensemble des exigences d'un système (par exemple, la spécification des exigences logicielles). Sachez que tous les produits d'activités des trois niveaux doivent être gérés et assurez-vous de connaître les relations entre eux.

Le texte ci-dessus décrit *ce qu'est* la gestion des exigences. Le reste de ce chapitre est consacré au *comment*: toutes sortes de pratiques applicables pour faire fonctionner la gestion des exigences. Avant d'entrer dans les détails de la gestion des exigences, examinons quelques principes de base pour la faire fonctionner. Si l'on veut gérer quelque chose, il faut être capable de le reconnaître, de le stocker et de le retrouver. Il est donc indispensable de disposer d'une identification unique, d'un degré approprié de normalisation, d'éviter la redondance, d'un dépôt central et d'un accès géré.

Dans la section `6.1`, nous examinons brièvement les situations qui influencent la valeur, l'importance et les efforts impliqués dans la gestion des exigences.

La section 6.2 suit les exigences dans leur cycle de vie en tant que partie des produits d'activités que les ingénieurs des exigences et les autres membres du personnel informatique produisent et utilisent lors du développement, de l'implémentation et de l'exploitation d'un système informatique.

Au cours du cycle de vie d'une exigence, plusieurs versions des produits d'activités (et des exigences qu'ils contiennent) sont créées, en commençant par un premier projet 0.1 qui, après une série de changements majeurs et mineurs, évolue vers, par exemple, une version finale 3.2. Le contrôle des versions est abordé à la section 6.3.

Lors du développement et de l'utilisation de systèmes informatiques, il n'est pas pratique de traiter toutes les exigences au cas par cas. Par conséquent, les ensembles cohérents d'exigences sont reconnus comme des configurations et des baselines, comme expliqué dans la section 6.4.

Afin de traiter efficacement les produits d'activités et les exigences, nous devons être en mesure de les identifier et de collecter des données à leur sujet. C'est le sujet de la section 6.5.

La section 6.6 traite de la traçabilité des exigences. La traçabilité est une caractéristique de qualité particulièrement importante des exigences, comme vous l'avez peut-être déjà compris en lisant les définitions de la gestion des exigences ci-dessus. Sans traçabilité, il est impossible de relier le comportement réel d'un système aux exigences initiales des parties prenantes.

La section 6.7 traite des modifications apportées aux exigences pendant leur durée de vie. Au cours des premières phases de leur existence, les changements peuvent être fréquents, mais après validation, les exigences doivent être stables. Toutefois, des changements se produiront encore. Pour les appliquer de manière ordonnée, il convient de mettre en place un processus défini de gestion du changement.

Par nature, les exigences diffèrent en importance et en valeur. Généralement, les ressources pour les élaborer sont limitées, de sorte que toutes les exigences ne sont pas implémentées. Cela signifie que les parties prenantes devront décider quand une certaine exigence sera implémentée ou même si elle sera implémentée ou non. L'établissement de priorités, décrit à la section 6.8, peut étayer cette décision.

6.1 Qu'est-ce que la gestion des exigences ?

Dans l'introduction, nous avons déjà vu que la gestion des exigences signifie la gestion des exigences existantes et des produits d'activités liés aux exigences, y compris le stockage, la modification et la traçabilité des exigences. Mais pourquoi les gérer ?

Nous gérons les exigences parce qu'elles sont vivantes ; elles sont créées, utilisées, mises à jour et supprimées à nouveau au cours de leur développement et de leur exploitation. Tout au long de ce cycle de vie, nous devons nous assurer que toutes les parties concernées ont accès aux versions correctes de toutes les exigences qui les concernent. Si nous ne gérons pas correctement les exigences, nous courons le risque que certaines parties négligent des

exigences, s'en tiennent à des exigences dépassées, travaillent avec des versions erronées, négligent des relations, etc. Cela peut sérieusement entraver l'efficacité et l'efficience du développement et de l'utilisation du système. En d'autres termes, la valeur d'une bonne gestion des exigences réside dans l'amélioration de l'efficacité et de l'efficience d'un système.

Cela signifie que la valeur de la gestion des exigences ne peut être séparée de la valeur du système en question et de son contexte. Dans la pratique, nous pouvons constater d'énormes différences dans l'importance et le niveau de la gestion des exigences, ainsi que dans les efforts qu'elle implique [Rupp2020], allant d'une tâche subsidiaire informelle d'un ingénieur des exigences avec une feuille de calcul, à une fonction à plein temps d'un gestionnaire d'exigences dédié avec une base de données d'exigences assistée par un outil.

Une gestion plus approfondie des exigences est nécessaire lorsque le nombre d'exigences, de parties prenantes et de développeurs est plus important, que la durée de vie prévue est plus longue, que le système subit davantage de modifications ou que les exigences de qualité sont plus élevées, que le processus de développement est plus complexe, que les normes et les réglementations sont plus strictes et qu'il est nécessaire de disposer d'une piste d'audit détaillée.

Nous constatons souvent que la gestion des exigences est quelque peu négligée au début d'un projet, lorsqu'une petite équipe travaille sur un ensemble évident d'exigences de haut niveau. Par la suite, la complexité augmente et l'équipe perd la vue d'ensemble, ce qui entraîne des problèmes de qualité et une diminution de l'efficacité. Il faut alors déployer beaucoup d'efforts pour atteindre le niveau de contrôle requis. Il est plus efficace d'investir des efforts dès le début d'un projet pour mettre en place les ressources et les processus de gestion des exigences en gardant à l'esprit les exigences attendues à la fin du projet.

6.2 Gestion du cycle de vie

Comme indiqué dans l'introduction, les exigences et les produits d'activités qui les contiennent ont une vie. Nous les voyons créés, élaborés, validés, consolidés, implémentés, utilisés, modifiés, maintenus, retravaillés, remaniés, retirés, archivés et/ou supprimés. C'est ce que nous entendons par leur cycle de vie : au cours de sa vie, une exigence peut se trouver dans un nombre limité d'états et peut passer de chaque état à plusieurs autres états prédéfinis – définis par les transitions d'état. Figure 6.1 présente un *diagramme d'état* simplifié comme modèle du cycle de vie d'une seule exigence. Par souci de clarté, les transitions d'état ne sont pas étiquetées. Par exemple, le passage de la catégorie " *En cours de construction* " à la catégorie " *Implémenté*"( ) peut être déclenché par une décision de mise en production prise par le Product Owner. Les deux zones grises indiquent où se situent les états dans ces domaines dans le processus global de développement du système.

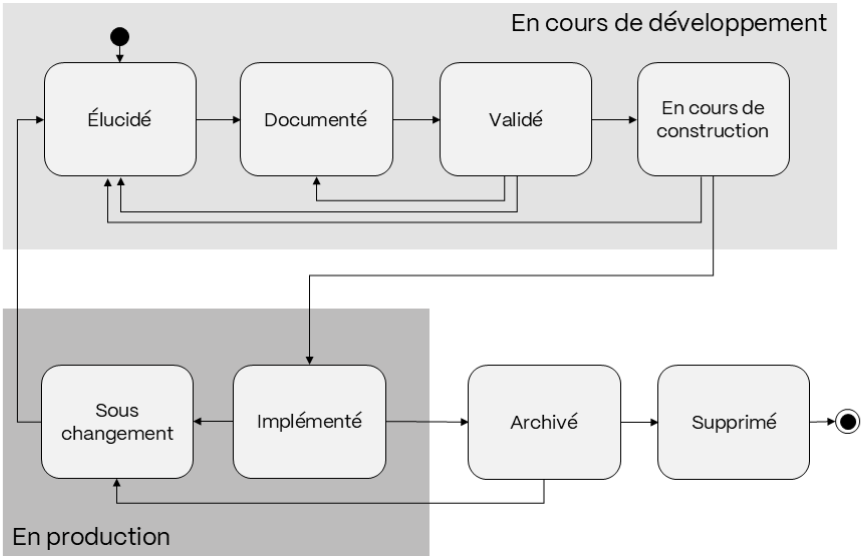


Figure 6.1 Diagramme d'état simplifié d'un cycle de vie des exigences.

Le fait que les produits d'activités et les exigences individuelles aient leurs propres cycles de vie, qui ne se chevauchent que partiellement, complique la situation. Par exemple, une *étude de définition d'* un produit d'activités est *en cours de modification*; cela ne signifie pas nécessairement que toutes les exigences contenues dans le produit d'activités doivent être modifiées. Et pour la même *étude de définition*, l'état *implémenté* n'a pas de sens ; seules certaines exigences qu'il contient seront implémentées – ou mieux : certains codes, basés sur ces exigences.

Un autre facteur de complication peut être que, dans la pratique, la vision du cycle de vie des exigences est différente selon les rôles. En tant qu'ingénieur des exigences, pour retracer votre travail, vous vous intéressez à différents états par rapport au chef de projet, et à d'autres états encore par rapport au chef de produit ou à un gestionnaire du changement : dans le diagramme ci-dessus, votre intérêt pourrait s'arrêter à " *validé*", alors que pour le chef de projet, il ne commence qu'à " *documenté*".

Les ingénieurs des exigences gèrent activement le cycle de vie de leurs produits. La gestion du cycle de vie implique :

- Définir des modèles de cycle de vie pour vos produits d'activités et les exigences qu'ils contiennent avec
- Les états qu'un produit d'activités ou une exigence peut prendre
- Les transitions autorisées entre ces états
- Les événements qui déclenchent le passage d'un état à un autre
- Veiller à ce que seules les transitions explicitement autorisées se produisent
- Enregistrement des états réels que prennent les produits d'activités et les exigences
- Enregistrement des transitions effectives
- Rendre compte de ces états et transitions

En d'autres termes, assurez-vous de connaître l'état dans lequel se trouvaient, se trouvent et se trouveront vos exigences, la manière dont elles peuvent changer et les raisons de ces changements.

Par exemple, en tant qu'ingénieur des exigences, on peut vous demander d'indiquer qui a approuvé quelle version d'une exigence à publier en tant que contribution à la phase de codage. Le suivi des états des exigences au cours de leur cycle de vie peut également être utile pour établir des tableaux de bord et des rapports sur l'état d'avancement d'un projet. Il peut s'agir d'un bon moyen d'organiser le travail et d'identifier les exigences sur lesquelles il faut travailler en premier.

L'état d'un produit d'activités dans le cadre de la gestion du cycle de vie est souvent enregistré dans un attribut (voir section 6.5). Il peut également être utile de documenter les dates de début et de fin de cet État dans les attributs. Dans les projets agiles, l'état d'un produit d'activités (élément) peut être dérivé de sa position dans le backlog du produit, le backlog de tâches et/ou sur le tableau des tâches. En outre, le fait de répondre aux critères de la *definition of ready* et de la *definition of done* peut fournir des informations pertinentes, car le fait de répondre à ces critères signifie en fait que l'on atteint un état suivant.

L'exhaustivité et le niveau de détail de la gestion du cycle de vie doivent être adaptés aux besoins du client, du projet et du système. Par exemple, les états "en développement", "en production" et "archivé" peuvent suffire. Dans les projets complexes ou critiques, vous pouvez avoir besoin d'un modèle beaucoup plus détaillé des états, de procédures strictes concernant les transitions d'état et d'une piste d'audit qui montre ce qui s'est passé pendant le projet.

6.3 Contrôle des versions

Il est courant que les produits d'activités et les exigences individuelles faisant partie d'un produit d'activités subissent certaines modifications au cours de leur cycle de vie (voir la section 6.7 pour plus d'informations sur la gestion de ces modifications). Après chaque modification, le produit d'activité est différent de ce qu'il était auparavant : il est devenu une nouvelle version.

Nous voulons contrôler les versions de ces produits pour deux raisons :

- Il arrive que les changements se fassent mal. Au bout d'un certain temps, des défauts sont constatés ou les avantages escomptés ne sont pas réalisés. Dans ce cas, nous pouvons mettre en œuvre de nouveaux changements dans une prochaine version, mais nous pouvons également décider de revenir à une version antérieure et de poursuivre à partir de là. Ou peut-être qu'en y réfléchissant bien, nous préférons finalement la version antérieure.
- Nous voulons connaître l'histoire du produit d'activités, comprendre son évolution depuis son origine jusqu'à sa situation actuelle. Cela peut nous aider lorsque nous

devons décider de changements futurs, ou simplement répondre à des questions sur les raisons pour lesquelles le produit d'activités actuel est ce qu'il est.

Le contrôle de version nécessite la mise en place de trois mesures :

- Une *identification* de chaque version, pour distinguer les différentes versions d'un produit d'activités. Il s'agit du numéro de version, souvent complété par une date de version.
- Une description claire de chaque *changement*. Vous devez être capable de dire et de comprendre la différence entre une certaine version et son prédécesseur. Cette description des modifications doit être clairement liée au numéro de version.
- Une politique stricte en matière de *stockage* des versions, vous permettant de localiser et de récupérer les anciennes versions. À moins que des limitations de stockage ne l'imposent, vous devez conserver toutes les versions antérieures de tous vos produits d'activités, faute de quoi vous risquez de ne pas pouvoir restaurer une version si vous en avez besoin. D'un autre côté, il est rare que le stockage soit illimité. Il est donc judicieux de mettre en place une politique d'archivage et de nettoyage des produits d'activités qui ne sont plus utilisés.

En général, un produit d'activités contient plusieurs exigences. Si une seule exigence de ce produit d'activités change, cette exigence et le produit d'activités doivent recevoir un nouveau numéro de version, tandis que les exigences inchangées de ce produit d'activités conservent leur ancien numéro de version. Cela pourrait vite devenir très confus. Une solution pratique pourrait consister à numéroter les versions au niveau du produit d'activités uniquement et à laisser toutes les exigences de ce produit hériter du numéro de version et de l'historique des modifications du produit d'activités.

Les numéros de version se composent généralement de deux parties (au moins) :

- *Version*. En principe, la version commence à *zéro* tant que le produit d'activités est en cours de développement. Lorsqu'il est officiellement approuvé, publié et/ou lancé, nous lui attribuons la *première* version. Ensuite, la version n'est augmentée que pour les mises à jour majeures et substantielles.
- *L'incrément*. Il commence généralement à *un* et est incrémenté à chaque changement (visible de l'extérieur), du côté du contenu ou souvent uniquement textuel ou rédactionnel. Une sous-incrémentation supplémentaire peut être utilisée pour la correction des fautes de frappe uniquement. L'incrément *neuf* est parfois utilisé pour désigner une version finale juste avant l'approbation ou la publication.

Un nouveau numéro de version est attribué à chaque modification formelle.

Souvent, un changement dans l'état du cycle de vie d'un produit d'activités n'est pas considéré comme une raison d'incrémenter le numéro de version, à moins qu'il ne soit accompagné d'un changement dans le contenu ou le texte. Si, par exemple, une exigence reçoit l'état validé et le numéro de version 1.0 après approbation, il n'est pas nécessaire de modifier ce numéro de version si l'état passe à "en cours de construction", puis à "implémenté". L'état peut finalement se terminer par archivé tout en conservant le même numéro de version 1.0.

6.4 Configurations et baselines

Supposons que vous conserviez, comme indiqué ci-dessus, toutes les versions de toutes les exigences que vous développez au cours d'un projet. Vous aurez alors une base de données en constante expansion, remplie d'exigences, et vous commencerez à perdre la vue d'ensemble. Un jour, votre client se présente à votre bureau et vous demande : "Nous avons mis en place votre système dans toutes nos succursales. Il semble qu'il y ait maintenant un problème avec les calculs de notre bureau de Barcelone. Pouvez-vous me dire quelle version des exigences de calcul ils utilisent là-bas ?" Si vous ne pouvez pas répondre à cette question, vous regretterez de ne pas avoir accordé plus d'attention à la gestion des configurations.

Alors, qu'est-ce qu'une configuration ? Vous trouverez une définition dans le glossaire du CPRE [Glin2025], mais en résumé, pour un ingénieur des exigences, une configuration est un ensemble cohérent de produits d'activités logiquement liés qui contiennent des exigences. Nous sélectionnons cet ensemble dans un but précis, généralement pour préciser quelles exigences sont ou étaient valables dans une certaine situation.

Cela permet de définir les propriétés suivantes pour une configuration correcte :

- *Connexion logique*. L'ensemble des exigences de la configuration sont regroupées en vue d'un certain objectif.
- *Cohérence*. L'ensemble des exigences ne présente pas de conflits internes et peut être intégré dans un système.
- *Caractère unique*. La configuration elle-même et les exigences qui la composent sont clairement et uniquement identifiées.
- *Inchangeabilité*. La configuration est composée d'exigences sélectionnées, chacune ayant une version spécifique qui ne sera jamais modifiée dans cette configuration.
- *Base de la réinitialisation*. La configuration permet de revenir à une configuration précédente si des changements indésirables semblent s'être produits.

Une configuration est documentée comme un produit d'activités, avec une identification unique, un état, un numéro de version et une date, comme tout autre produit d'activités. Cependant, une configuration étant par définition immuable, elle n'aura toujours qu'une seule version (par exemple, 1.0).

Une configuration a toujours deux dimensions [CoWe1998] :

- **La dimension du produit**

  Il indique les exigences incluses dans cette configuration spécifique. Parfois, une configuration contient toutes les exigences disponibles, mais en général, il s'agit d'une certaine sélection – par exemple, toutes les exigences qui sont implémentées dans la version française d'un système. La version britannique du même système pourrait alors avoir une configuration différente.

- La dimension de la *version*

  Dans une configuration spécifique, chaque exigence sélectionnée est présente dans exactement une, et une seule, version. Il peut s'agir de la dernière version ou d'une version antérieure, en fonction de l'objectif de la configuration elle-même. Dès qu'une seule version différente d'une exigence unique est sélectionnée, il s'agit d'une nouvelle configuration. Imaginez un système pour lequel une nouvelle version sera implémentée avec certaines exigences dans une version supérieure : cette nouvelle version aura alors une configuration différente.

Figure 6.2 donne un autre exemple de configurations différentes consistant en des ensembles spécifiques de versions d'exigences.

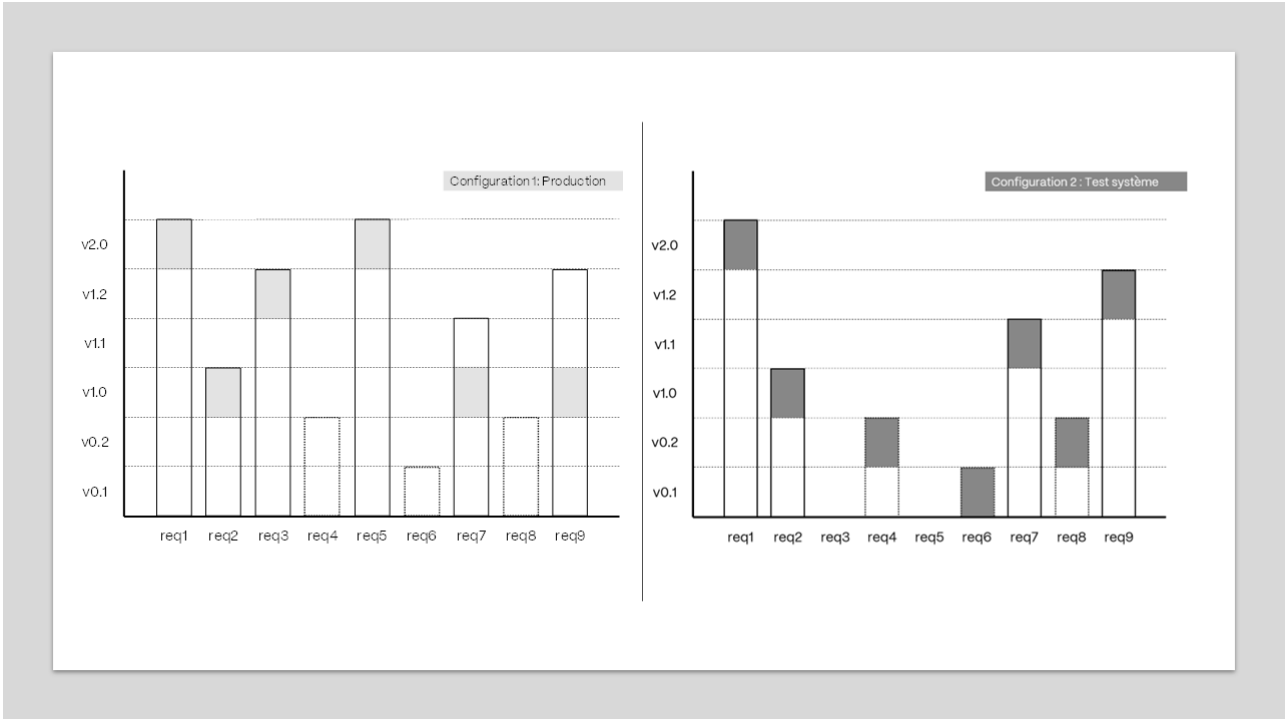


Figure 6.2 Exemple de configurations

La figure ci-dessus montre un exemple de différentes configurations d'un certain système. Il présente un ensemble de neuf exigences. Certaines d'entre elles en sont encore aux premiers stades de développement – par exemple, l'exigence 6 avec la version v0.1. D'autres exigences ont eu plus de versions – par exemple, l'exigence 1, qui est finalisée et a déjà fait l'objet d'une mise à jour majeure, ce qui fait qu'elle est maintenant la version v2.0.
L'image de gauche montre la configuration actuellement en production. Il se compose de R1 v2.0, R2 v1.0, R3 v1.2 (cette exigence a fait l'objet de deux mises à jour mineures après son implémentation), R5 v2.0, R7 v1.0 et R9 v1.0. R4, R6 et R8, en cours de développement, ne sont pas présents dans cette configuration, de même que les nouvelles versions de R7 et R9.
L'image de droite montre la configuration qui, au même moment, est présente dans l'environnement de test du système. Certaines exigences (R1, R2) sont identiques, d'autres ne sont plus présentes (R3, R5), les exigences en cours de développement (R4, R6, et R8) sont incluses ici, et deux exigences (R7 et R9) sont présentes dans une version plus élevée que dans la configuration de l'environnement de production.

Dans de nombreux projets, certaines configurations sont traitées de manière particulière : ces configurations sont appelées "baselines". Une *baseline* est une configuration stable, validée et contrôlée qui marque une étape ou un autre type de *point d'arrêt* dans le projet. Un exemple peut être la configuration à la fin de la phase de conception, juste avant de commencer la phase de codage, ou la configuration qui est valide lors de la mise en service d'une certaine version.

Dans un projet agile, le backlog du sprint sert de référence au début de l'itération suivante. Les niveaux de référence sont utiles à des fins de planification, car ils représentent un point de départ stable pour la phase suivante. Ils sont souvent gelés et mis de côté comme un point d'ancrage dans la vie trépidante d'un projet. Si quelque chose tourne mal dans le projet, l'équipe peut revenir à la situation de référence et repartir de là.

Pour l'ingénieur des exigences, c'est principalement la configuration des produits d'activités contenant des exigences qui est importante. Mais dans la pratique, la configuration d'un projet a une portée beaucoup plus large et contient des versions sélectionnées des produits d'activités de tous les membres de l'équipe, tels que les exigences, les conceptions, le code et les cas de test. Dans les projets complexes, la gestion de la configuration peut être un travail à plein temps, réalisé avec des outils dédiés.

6.5 Attributs et vues

En tant qu'ingénieur des exigences, votre travail consiste à produire toutes sortes de documents contenant des exigences. Ces exigences devront être gérées, sinon vous et votre équipe perdrez rapidement la vue d'ensemble. Pour gérer les exigences, vous devez collecter et conserver des données à leur sujet – des métadonnées, des données sur les données. Les métadonnées rendent les produits d'activités tangibles et gérables ; grâce aux métadonnées, vous pouvez fournir et obtenir des informations sur les exigences et répondre aux questions pertinentes pendant et après le cycle de vie du projet ou du produit. Pensez à des questions telles que*"Quelles sont les exigences prévues pour la prochaine version ?*"

ou*"Quel est le degré d'effort que cette diffusion est susceptible de nécessiter ?*" ou "*Combien d'exigences ont une priorité élevée ?*"

Si l'on considère les exigences comme des entités au sujet desquelles des informations sont requises, les caractéristiques de ces exigences sont appelées *attributs*. Dans ce chapitre, nous avons déjà vu quelques attributs communs, tels que l'identifiant unique, le numéro de version, l'état, plusieurs dates. Les attributs à définir pour les exigences dépendent des besoins d'information des parties prenantes du projet et du système. Au début d'un projet, un *schéma d'attributs* doit être établi pour permettre à l'ingénieur des exigences de répondre à ces besoins.

Les normes pertinentes constituent un bon point de départ. La norme ISO [ISO29148] mentionne :

Identification. Chaque exigence doit avoir un identifiant unique et immuable, tel qu'un numéro, un nom ou un moyen mnémotechnique. Sans une identification appropriée, la gestion des exigences est très difficile.

- *Priorité pour les parties prenantes*. La priorité (convenue) de l'exigence du point de vue des parties prenantes. Voir la section `6.8` pour plus d'informations sur la manière de déterminer cette priorité.
- *Dépendance*. Il existe parfois une dépendance entre les exigences. Cela peut signifier qu'une exigence de faible priorité doit être implémentée en premier parce qu'une autre exigence de priorité élevée en dépend.
- *Risque*. Il s'agit de la possibilité que l'implémentation de l'exigence entraîne des problèmes, tels que des dommages, des coûts supplémentaires, des retards, des actions en justice. Par nature, il s'agit d'une estimation, qui doit être basée sur un consensus entre les parties prenantes.
- *Source*. Quelle est l'origine de l'exigence, d'où vient-elle ? Vous pouvez avoir besoin de ces informations pour la validation, la résolution de conflits, la modification ou la suppression.
- *Justification*. La justification vous donne la raison pour laquelle l'exigence est nécessaire, les objectifs des parties prenantes qui doivent être remplis par l'exigence.
- *Difficulté*. Il s'agit d'une estimation de l'effort nécessaire pour implémenter l'exigence. Cet attribut est nécessaire pour la planification et l'estimation des projets.
- *Type*. Cet attribut indique s'il s'agit d'une exigence fonctionnelle, d'une exigence de qualité ou d'une contrainte.

Il existe de nombreuses façons de stocker ces informations. Elle peut être contenue dans des documents ou stockée dans une feuille de calcul ou une base de données, les exigences étant représentées par des lignes et leurs attributs par des colonnes. Dans un contexte agile, les exigences peuvent être consignées sur des "story cards", où les rubriques de la carte sont les attributs. Comme nous l'avons vu au chapitre 7, les outils de gestion des exigences doivent permettre de stocker des données sur les exigences et d'établir des rapports à leur sujet.

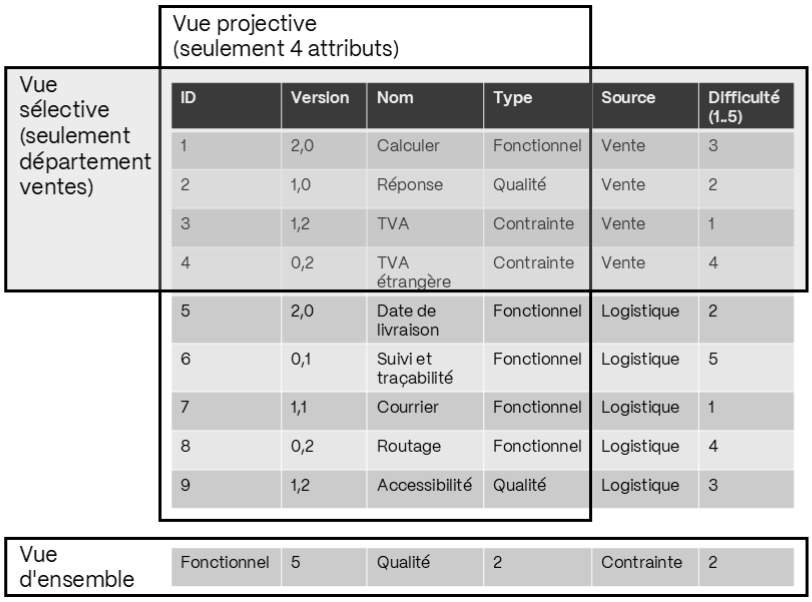
Les attributs vous permettent de fournir des informations sur vos produits d'activités et les exigences qu'ils contiennent. La manière la plus simple de procéder consiste à produire un rapport contenant toutes les données relatives à toutes les versions de toutes les exigences. Pour tout système autre que le plus simple, un tel rapport sera inutile car personne ne sera en mesure de superviser toutes les informations en raison de leur complexité écrasante. Vous devez donc adapter vos rapports en fonction des besoins d'information de vos publics cibles. Pour ce faire, on utilise les *vues* [Glin2025].

Une vue est un moyen (souvent prédéfini) de filtrer et de trier les données de vos produits d'activités, ce qui permet d'obtenir un rapport qui montre précisément ce dont le public a besoin, ni plus ni moins. Une vue est définie dans le but explicite de fournir des informations pertinentes à un groupe cible spécifique.

Nous distinguons trois types de vues :

- *Vues sélectives*. Ces vues donnent des informations sur une sélection délibérée des exigences plutôt que sur l'ensemble des exigences. Par exemple, une vue sur les dernières versions des exigences, ou sur toutes les exigences dont l'état est *validé*, ou sur les exigences dont la priorité pour les parties prenantes *est élevée*; l'accent peut être mis sur un sous-système, ou au contraire fournir une vue d'ensemble abstraite du système par le biais de ses exigences de haut niveau uniquement.
- *Vues projectives*. Une vue projective montre une sélection de toutes les données (attributs) des exigences – par exemple, seulement l'identification, le numéro de version et le nom.
- *Vues agrégées*. Dans une vue agrégée, vous trouverez des résumés, des totaux ou des moyennes, calculés à partir d'un ensemble d'exigences. Un exemple serait le nombre total d'exigences par département : par exemple, 4 pour les ventes, 5 pour la logistique.

Figure 6.3 donne un exemple de ces types de vues.



ID	Version	Nom	Type	Source	Difficulté (1..5)
1	2,0	Calculer	Fonctionnel	Vente	3
2	1,0	Réponse	Qualité	Vente	2
3	1,2	TVA	Contrainte	Vente	1
4	0,2	TVA étrangère	Contrainte	Vente	4
5	2,0	Date de livraison	Fonctionnel	Logistique	2
6	0,1	Suivi et traçabilité	Fonctionnel	Logistique	5
7	1,1	Courrier	Fonctionnel	Logistique	1
8	0,2	Routage	Fonctionnel	Logistique	4
9	1,2	Accessibilité	Qualité	Logistique	3
Fonctionnel	5	Qualité	2	Contrainte	2

Figure 6.3 Différents types de vues

Dans la plupart des cas, une combinaison de vues est utilisée – par exemple, si vous voulez fournir une liste avec les ID, les numéros de version, les noms et les types (= projectif) de toutes les exigences pour le département des ventes (= sélectif).

6.6 Traçabilité

Tout au long de ce manuel, nous avons évoqué le thème de la traçabilité [GoFi1994]. Sans une traçabilité adéquate, l'ingénierie des exigences est difficilement réalisable, car il est impossible de faire ce qui suit :

- Fournir la preuve qu'une certaine exigence est satisfaite
- Prouver qu'une exigence a été mise en œuvre et par quels moyens
- Démontrer la conformité du produit avec les lois et les normes applicables
- Rechercher les produits d'activités manquants (par exemple, vérifier s'il existe des cas de test pour toutes les exigences)
- Analyser les effets d'une modification des exigences (voir section 6.7)

Dans de nombreux cas, en particulier pour les systèmes critiques de sécurité, les normes de processus exigent même explicitement la mise en œuvre de la traçabilité.

La traçabilité permet de répondre à trois types de questions (voir également Figure 6.4) :

- *Traçabilité amont*: Quelle est l'origine d'une exigence donnée ? Où a-t-elle été trouvée ? Quelles sources (parties prenantes, documents, autres systèmes) ont été analysées lors de l'élucidation ?

La traçabilité ascendante est aussi bien connue que la traçabilité de la spécification des exigences préalables.

- *Traçabilité aval*: Où cette exigence est-elle utilisée ? Quels sont les produits livrables (modules codés, cas de test, procédures, manuels) qui en dépendent ?

  La traçabilité aval est aussi bien connue que la traçabilité amont des spécifications.

- *Traçabilité entre les exigences*: D'autres exigences dépendent-elles de cette exigence ou vice versa (par exemple, des exigences de qualité liées à une exigence fonctionnelle) ? L'exigence est-elle un raffinement d'une exigence de niveau supérieur (par exemple, une epic raffinée dans un certain nombre de user stories, une user story détaillée avec un certain nombre de critères d'acceptation) ? Quel est le lien entre les deux ?

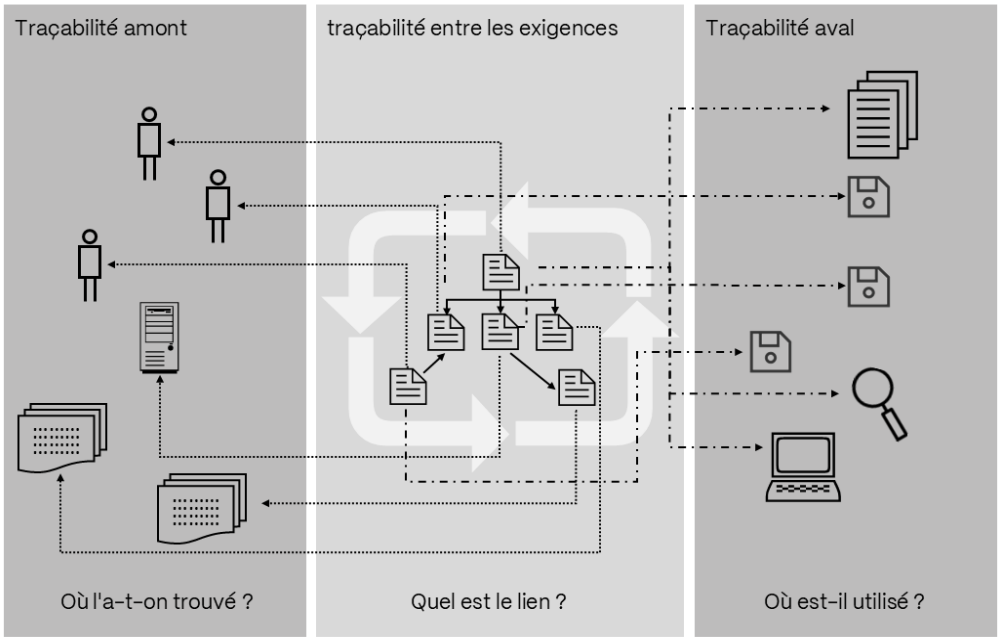


Figure 6.4 Types de traçabilité

Il existe plusieurs façons de documenter la traçabilité. Cela se fait souvent de *manière implicite,* par exemple en appliquant des structures de documents, des modèles standard ou des conventions de nommage. Si vous identifiez toutes vos exigences avec le code *Req-xxx-nnn*, où *xxx* représente le département qui a demandé l'exigence, tout le monde comprendra que Req-sal-012 est une exigence pour le département des ventes (pour la traçabilité en amont). Si vous publiez un document énumérant toutes les exigences qui seront implémentées dans la version du 1er juillet, vous fournissez des informations de traçabilité implicite.

Et si vous rédigez un document comportant une section consacrée, par exemple, au calcul des prix, il pourrait s'agir d'un exemple de traçabilité entre les exigences. Un autre exemple

pourrait être un modèle de haut niveau et une description textuelle des exigences détaillées qui s'y rapportent.

Dans les projets plus complexes, la traçabilité doit (également) être documentée de *manière explicite*. Pour une traçabilité explicite, vous documentez la relation entre les produits d'activités sur la base de leur identification unique. Cela peut se faire sous différentes formes [HuJD2011]:

- L'utilisation d'attributs spécifiques tels que *Source* suggéré par la norme ISO [ISO29148]
- Dans les documents, ajout de références aux documents précédents, à d'autres produits d'activités ou à des exigences individuelles
- Élaboration d'une matrice de traçabilité dans une feuille de calcul ou un tableau de base de données (voir l'exemple à l'adresse Table 6.1 ci-dessous)
- Dans la documentation textuelle, à l'aide d'hyperliens de type Wiki
- Visualisation des relations de traçabilité dans un *graphe de traçabilité* (Figure `6.4` est une forme simplifiée d'un tel graphe)

Dans de nombreux cas, un outil de gestion des exigences ou de gestion de la configuration (voir le chapitre 7) offre une fonctionnalité permettant d'assurer la traçabilité. La gestion de la traçabilité dans un projet d'envergure peut s'avérer compliquée, surtout s'il faut également tenir compte du versioning. Dans ce cas, un bon outillage est indispensable.

`Table 6.1 Exemple d'une matrice de traçabilité`

Source	R1	R2	R3	R4	R5	R6	R7
Entretien avec Mme Smith 06/08	X	X			X		
Questionnaire de synthèse 12 mai	X			X		X	X
Rapport d'observation sur le terrain 07/03			X	X	X		
Règlement d'entreprise version 17.a.02			X			X	X
Documentation API HRM system v3.0.2.a	X			X	X		

6.7 Gestion du changement

"Principe 7: Bienvenue aux exigences changeantes, même tardivement dans le développement. Agile considère le changement comme un avantage concurrentiel pour le client." [BeeA2001]. Les pères fondateurs du mouvement agile ont été très clairs sur ce point : les changements d'exigences se produiront toujours, que vous le vouliez ou non. De nombreuses personnes n'aiment pas du tout les changements, car chaque changement est un risque, une menace pour la stabilité du projet et du système.

Cependant, la modification d'une exigence n'est pas un événement isolé : elle est déclenchée par des changements dans le contexte du système, par de nouvelles idées des parties prenantes, par le comportement des concurrents, etc. ; une loi entre en vigueur, ajoutant une nouvelle contrainte au système ; en raison de la demande croissante du marché, la performance du système doit être améliorée ; un système concurrent est lancé avec des caractéristiques *intéressantes* que votre client souhaite également. Un changement doit donc être considéré comme une chance d'obtenir un meilleur système, d'apporter plus de valeur aux utilisateurs.

Cependant, quelle que soit la situation, tout changement comporte également un risque. Il peut introduire des défauts, entraînant une défaillance du système. Cela peut retarder l'avancement du projet. Cela peut demander plus d'efforts et d'argent que ce qui avait été calculé auparavant. Les utilisateurs peuvent ne pas l'apprécier et refuser de travailler avec. En bref, les choses peuvent mal tourner et perturber un projet ou un système auparavant stable. Mais cela ne signifie pas que les changements sont mauvais et doivent être évités ; cela signifie que tous les changements doivent être gérés avec soin pour obtenir une valeur optimale à des coûts acceptables avec un risque minimal.

Dans la littérature sur la gestion des services informatiques (voir [Axelos2019]), l'*accompagnement au changement* est décrit comme l'une des pratiques fondamentales. Cette pratique garantit que les changements sont mise en œuvre de manière efficace, sûre et opportune afin de répondre aux attentes des parties prenantes. Cette pratique permet d'équilibrer l'efficacité, le rendement, la conformité et le contrôle des risques. Elle se concentre sur trois aspects :

- S'assurer que tous les risques ont été correctement évalués
- Autoriser à procéder à des modifications
- Gérer la mise en œuvre du changement

L'habilitation au changement implique qu'une organisation désigne une autorité chargée de décider des changements et définisse un processus pour les gérer. Voir Figure 6.5 pour un aperçu de ce processus. Ces mesures sont généralement adaptées à l'approche du développement et au moment où un changement se produit.

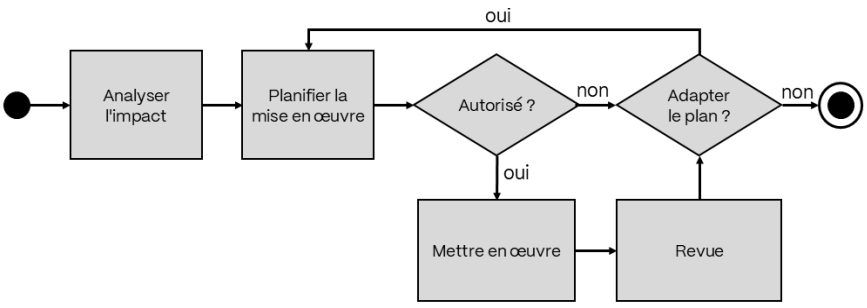


Figure 6.5 Processus d'accompagnement au changement

Tant qu'une exigence est à l'état de projet, l'auteur a le pouvoir de la modifier et aucun processus strict n'est suivi.

Dès qu'une exigence est publiée en vue d'une utilisation ultérieure dans le cadre du projet, l'auteur n'est plus libre de décider, car chaque modification aura un impact sur d'autres produits d'activités basés sur cette exigence. Avant de décider si un changement doit être implémenté, une analyse d'impact doit être réalisée pour clarifier les efforts et les risques liés au changement. C'est là que la traçabilité est indispensable. Dans une approche de développement *linéaire*, l'autorité en matière de changement est souvent confiée à la direction du projet, à un comité *de* pilotage ou à un *comité de contrôle des changements*, et un processus est suivi, avec une décision formelle sur le changement et la planification de sa mise en œuvre. Dans une approche de développement *itératif*, le pouvoir de modification appartient généralement au product owner, qui décide de la modification et ajoute une modification acceptée au product backlog en tant que nouvel élément (produit d'activités). La suite de la mise en œuvre est alors traitée comme n'importe quel autre élément du backlog de produit.

Une fois qu'une exigence est implémentée dans un système opérationnel, un processus encore plus strict doit être suivi, car chaque changement aura désormais une influence sur les utilisateurs et les processus opérationnels.

Dans ce domaine, une distinction est souvent faite entre les changements *standard* (à faible risque, bien compris et autorisé au préalable, par exemple une modification du pourcentage de TVA), les changements *normaux* (basés sur une demande de changement formelle, programmés, évalués et autorisés, par exemple une modification de l'algorithme de calcul des prix) et les changements d'*urgence* (à mettre en œuvre dès que possible, par exemple pour résoudre un incident, mais qui impliquent rarement une modification des exigences). En général, l'autorité en matière de changement relève d'un *comité consultatif des changements* [Math2019]; dans une approche itérative telle que DevOps, un changement peut être autorisé par un responsable de la mise en production.

6.8 Priorisation

Les exigences elles-mêmes ne sont que des concepts dans l'esprit des gens. Elles n'apportent de la valeur que lorsqu'elles sont implémentées dans un système opérationnel. Cette mise en œuvre nécessite des efforts, du temps, de l'argent et de l'attention. Dans la plupart des cas, ces ressources sont limitées, ce qui signifie que toutes les exigences ne peuvent pas être implémentées, du moins pas en même temps. Cela signifie que les parties prenantes doivent décider quelles exigences doivent être satisfaites en premier et lesquelles peuvent être implémentées plus tard (ou pas du tout). En d'autres termes : priorisation [Wieg1999].

La priorité d'une exigence est définie comme le niveau d'importance qui lui est attribué selon certains critères [Glin2025]. Par conséquent, vous devez d'abord déterminer les critères à utiliser pour évaluer les exigences avant de pouvoir les classer par ordre de priorité. Toutefois, avant de déterminer les critères d'évaluation, il faut savoir quel est l'objectif de la

priorisation. Cet objectif n'est généralement pas le vôtre en tant qu'ingénieur des exigences, mais celui de certaines parties prenantes, et vous devez donc déterminer qui sont les parties prenantes pour cette priorisation. Et lorsque vous connaissez leur objectif, il est généralement clair que ce ne sont pas toutes les exigences qui doivent être classées par ordre de priorité, mais seulement un sous-ensemble défini.

En résumant ce qui précède, nous pouvons esquisser une séquence d'étapes à suivre si nous voulons classer les exigences par ordre de priorité :

- **Définir les principaux objectifs et les principales contraintes pour la priorisation**

  Le contexte du projet et du système détermine en grande partie les raisons de la priorisation. Si, par exemple, vous établissez des priorités pour décider quelles fonctionnalités seront implémentées dans la prochaine version, vous pouvez vous concentrer sur la valeur métier; si l'objectif est de sélectionner des user stories pour la prochaine itération, les story points et les dépendances techniques seront plus importants. Des contraintes techniques ou juridiques peuvent limiter les choix à faire.

- **Définir les critères d'évaluation souhaités**

  En principe, les objectifs et les contraintes dictent les critères à utiliser. Les critères couramment utilisés sont la valeur métier pour les parties prenantes, l'urgence perçue par les utilisateurs, l'effort de mise en œuvre, les risques d'utilisation, les dépendances logiques et techniques, la nature juridiquement contraignante d'une exigence, ou simplement la préférence (inter-) subjective des parties prenantes concernées. Parfois, un seul critère est utilisé, mais une sélection équilibrée de plusieurs critères pertinents peut donner un meilleur résultat.

- **Définir les parties prenantes qui doivent être impliquées**

  Les objectifs et les contraintes influencent les parties prenantes que vous devez impliquer dans l'établissement des priorités, mais d'un autre côté, certaines parties prenantes fixent elles-mêmes ces objectifs, et vous devez donc être conscient de l'interdépendance. Par exemple, lorsque vous fixez des priorités pour le lancement d'un nouveau système, vous inviterez probablement des représentants métier et un panel de futurs clients. Lors de l'établissement des priorités du backlog produit pour décider de la prochaine itération, l'équipe de scrum est impliquée.

- **Définir les exigences qui doivent être priorisées**

  Il est peu probable que l'ensemble des exigences doive être classé par ordre de priorité. Une fois de plus, cela dépend principalement des objectifs et des contraintes de l'établissement des priorités. Par exemple, les contraintes peuvent dicter que certaines exigences sont incontournables. En fait, il n'est utile de prioriser que les exigences pour lesquelles vous avez le choix de les inclure ou non dans une prochaine étape du processus de développement. Cela signifie que la phase du projet est également un facteur important. Dans une phase initiale, vous pouvez inclure des versions préliminaires dans l'établissement des priorités ; dans une phase tardive, vous limiterez souvent l'établissement des priorités aux exigences qui se trouvent

dans une version stable. Il faut savoir que les exigences à classer par ordre de priorité doivent se situer à un niveau d'abstraction comparable en fonction des objectifs de classement par ordre de priorité. Dans une phase initiale du projet, par exemple, vous pouvez donner la priorité à des thèmes ou à des fonctionnalités tout en donnant la priorité à des user stories lors de la planification de l'itération.

- **Choisir la technique de priorisation**

  Une technique de priorisation est la manière dont les parties prenantes priorisent les exigences. Comme décrit ci-dessous, il existe plusieurs techniques, qui diffèrent en termes d'effort, d'exhaustivité et de niveau de détail. Ici aussi, les objectifs et les contraintes sont déterminants, mais le facteur le plus important est que les parties prenantes concernées s'accordent sur la technique qu'elles ont l'intention d'utiliser. Dans le cas contraire, elles n'accepteront pas le résultat et votre effort de priorisation sera vain.

- **Établir les priorités**

  Une fois la préparation terminée, la priorisation proprement dite peut être effectuée. Tout d'abord, tous les critères d'évaluation définis sont appliqués à toutes les exigences sélectionnées. En collaboration avec les parties prenantes concernées, vous appliquez ensuite la technique de priorisation choisie aux exigences évaluées. Vous obtenez ainsi une liste d'exigences classées par ordre de priorité. Cependant, il pourrait y avoir un problème. Les différentes parties prenantes peuvent avoir des priorités différentes, même si elles sont d'accord sur les critères évalués. Dans ce cas, il s'agit généralement d'un conflit d'exigences qui doit être résolu comme n'importe quel autre conflit, comme décrit dans la section `4.3` sur la résolution des conflits.

Si l'on examine de plus près les techniques de priorisation, on distingue deux catégories :

- **Les techniques Ad-hoc**

  Avec les techniques ad hoc, les experts attribuent des priorités aux exigences sélectionnées sur la base de leur propre expérience. En principe, cette priorisation est basée sur un seul critère, à savoir la perception subjective de l'expert. Si cette expertise est de haut niveau et acceptable pour les parties prenantes, cette technique peut être un moyen rapide, peu coûteux et facile d'établir des priorités. Une variante consisterait à inviter plusieurs experts et à calculer une sorte de moyenne des priorités. Les techniques ad hoc courantes comprennent le classement par ordre de priorité (Top-10) et la priorisation MoSCoW (Must have, Should have, Could have, Won't have this time). L'analyse de Kano (section `4.2.1`) est également utile : les facteurs d'insatisfaction sont indispensables, les facteurs de performance sont souhaitables et les facteurs d'excitation peuvent être souhaitables ou non. Pour plus d'informations, voir, par exemple, [McIn2016].

- **Les techniques Analytiques**

  Les techniques analytiques utilisent un processus systématique d'attribution des priorités. Dans ces techniques, les experts attribuent une pondération à plusieurs

critères d'évaluation (tels que les avantages, les coûts, les risques, le temps de mise en œuvre, etc.) et, par la suite, les priorités des exigences sont calculées comme des résultats pondérés basés sur ces critères. Ces techniques demandent plus d'efforts et de temps mais ont l'avantage de donner un aperçu clair des facteurs qui déterminent les priorités et du processus par lequel les priorités sont établies. Cela peut stimuler l'acceptation du résultat par les parties prenantes. Toutefois, deux aspects doivent être pris en compte. Tout d'abord, le résultat est fortement influencé par les facteurs de pondération utilisés dans le calcul du résultat. Par conséquent, un accord entre les parties prenantes sur ces facteurs de pondération doit être établi avant de procéder à la hiérarchisation proprement dite. Dans le cas contraire, certains pourraient essayer de modifier les facteurs de pondération afin de manipuler les priorités. Le deuxième aspect à prendre en compte est que les critères évalués sont pour la plupart des estimations et non des faits mesurés. Les estimations sont souvent établies sur une simple échelle ordinale (faible, moyenne, élevée). La qualité des estimations est donc déterminante pour la qualité de la priorisation qui en résulte. Néanmoins, les techniques analytiques sont utiles pour établir un ordre de priorité clairement étayé, compris et donc accepté par les parties prenantes concernées. Pour une explication détaillée des techniques d'analyse, voir [Olso2014].

Il peut être tentant d'appliquer des techniques détaillées et approfondies et de passer beaucoup de temps à produire des estimations parfaitement exactes en termes d'argent, d'heures, de chiffre d'affaires escompté, etc. Ainsi, l'exigence A pourrait avoir une priorité calculée de 22,76, l'exigence B de 23,12 et l'exigence C de 20,29. Vous concluriez alors qu'il est évident que C doit être fait en premier et que A doit être fait avant B. Cependant, vous venez probablement d'introduire une pseudo-précision dans ce calcul, et il serait préférable de conclure que ces trois exigences sont d'égale importance, ce qui aurait pu être votre intuition dès le départ.
Veillez toujours à ce que l'effort que vous consacrez à l'établissement des priorités soit justifié par la valeur d'une priorisation correcte. Une fois de plus, gardez les objectifs à l'esprit et souvenez-vous du principe 1 : l'orientation vers la valeur.

6.9 Pour en savoir plus

Les manuels de Pohl [Pohl2010], Davis [Davi2005], Hull, Jackson et Dick [HuJD2011], van Lamsweerde [vLam2009] et Wiegers et Beatty [WiBe2013] fournissent un aperçu complet de la gestion des exigences. Le manuel CPRE Advanced Level handbook for Requirements Management de Bühne et Herrmann [BuHe2024] contient des informations supplémentaires sur le thème de la gestion des exigences.

Cleland-Huang, Gotel et Zisman [ClGZ2012] traitent en profondeur de la traçabilité.

Olson [Olso2014] et Wiegers [Wieg1999] traitent des techniques de hiérarchisation.

7 Support des outils

Un ingénieur des exigences a besoin d'outils pour exercer correctement son métier, tout comme un menuisier a besoin de ses outils, d'un crayon, d'un marteau, d'une scie et d'une perceuse pour concevoir et réaliser un meuble. Sans outils, il est difficile, voire impossible, d'enregistrer les exigences, de travailler ensemble sur les exigences et de contrôler les exigences.

Ce chapitre examine les différents types d'outils d'ingénierie des exigences disponibles et les aspects qui doivent être pris en compte pour introduire des outils d'ingénierie des exigences dans une organisation.

7.1 Outils dans l'ingénierie des exigences

L'ingénierie des exigences est une tâche difficile sans l'aide d'outils. Des outils sont nécessaires pour soutenir les tâches et les activités d'ingénierie des exigences. Les outils existants se concentrent sur le soutien de tâches spécifiques, telles que la documentation des exigences ou le soutien du processus d'IE, et rarement sur l'ensemble des tâches et activités du processus d'ingénierie des exigences. Il n'est donc pas surprenant que l'ingénieur des exigences doive disposer d'un ensemble d'outils pour soutenir les différents composants du processus d'ingénierie des exigences – tout comme le menuisier a besoin de plusieurs outils (par exemple, la conception assistée par ordinateur (CAO)) pour concevoir un meuble et a besoin d'outils tels qu'une scie, un grattoir et du papier de verre pour le réaliser.

Les outils ne sont qu'une aide au processus d'ingénierie des exigences et à l'ingénieur des exigences, et ces outils sont appelés outils CASE (computer-aided software engineering, ingénierie logicielle assistée par ordinateur). Les outils CASE soutiennent une tâche spécifique dans le processus de production de logiciels [Fugg1993].

Nous distinguons différents types d'outils qui prennent en charge les aspects suivants de l'ingénierie des exigences :

- **La gestion des exigences**

  Les outils de cette catégorie possèdent les propriétés nécessaires pour soutenir les activités et les thèmes décrits au chapitre 6. Ces outils permettent de mieux contrôler le processus d'ingénierie des exigences. Les exigences sont susceptibles de changer et, dans un environnement où cela se produit fréquemment, il est indispensable de disposer d'un outil doté des propriétés appropriées. Les outils de cette catégorie prennent en charge :

  - Définition et stockage des attributs des exigences afin d'identifier et de collecter des données sur les produits d'activités et les exigences, comme décrit dans la section 6.5
  - Facilitation et documentation de la hiérarchisation des besoins (Section 6.8)
  - Gestion du cycle de vie, contrôle des versions, configurations et baselines, tels que décrits dans les sections 6.2, 6.3, et 0

- Suivi et traçabilité des exigences, ainsi que des défauts dans les exigences et les produits d'activités (Section 6.6)
- Gestion des changements pour les exigences ; comme nous l'avons appris dans la section 6.7, les changements sont inévitables et doivent être gérés avec soin

- **Gestion du processus d'IE**

  Pour soutenir le processus d'ingénierie des exigences, des informations sont nécessaires pour permettre d'ajuster ou d'améliorer le processus. Ce type d'outil peut :

  - Mesurer et rendre compte du processus d'ingénierie des exigences et du flux de travail
  - Ces informations permettent d'améliorer le processus d'ingénierie des exigences et de réduire les déchets.
  - Mesurer la qualité du produit et en rendre compte
  - Ces informations permettent de détecter les défauts et les imperfections, qui peuvent à leur tour être utilisés pour améliorer la qualité du produit.

- **Documentation des connaissances sur les exigences**

  La quantité de connaissances (et d'exigences) accumulées dans le cadre d'un projet peut être énorme. En outre, une grande quantité de connaissances est accumulée sur un produit au cours de son cycle de vie. Toutes les informations pertinentes doivent être soigneusement documentées pour permettre ce qui suit :

  - Partage et création d'une compréhension commune des besoins
  - Sécuriser les exigences en tant qu'obligation légale
  - Une vue d'ensemble et un aperçu des exigences

- **Modélisation des exigences**

  Comme nous l'avons appris à la section 3.4.1.6, l'expression des exigences à la fois dans des diagrammes et en langage naturel utilise les points forts des deux formes de notation. Un outil capable de modéliser les exigences vous permet de :

  - Structurer ses propres pensées ; il peut être utilisé comme une aide à la réflexion
  - Spécifier les exigences dans un langage plus formel que les exigences textuelles, avec tous les avantages que cela comporte

- **Collaboration dans l'ingénierie des exigences**

  Lorsque plusieurs personnes et disciplines travaillent sur un même projet, un outil peut soutenir et permettre cette collaboration, en particulier dans le monde dans lequel nous vivons aujourd'hui, où de plus en plus d'activités sont réalisées localement (à la maison). Ce type d'outil permet d'obtenir, de documenter et de gérer les exigences.

- **Test et/ou simulation des exigences**

  Les outils sont de plus en plus sophistiqués. De plus en plus d'options sont développées pour tester et/ou simuler les exigences à l'avance. Cela permet de mieux prévoir si les exigences proposées auront l'effet escompté.

Les outils disponibles sont souvent un mélange des deux précédents. Comme indiqué précédemment, il peut être nécessaire de combiner différents outils pour soutenir de manière adéquate l'ingénierie des exigences. Si différents outils sont utilisés, il est important de veiller à l'intégration entre eux et à l'interaction avec d'autres applications et systèmes afin de garantir un fonctionnement sans heurts.

Parfois, d'autres types d'outils (par exemple, des outils bureautiques ou de suivi des incidents) sont utilisés, ou plutôt mal utilisés, pour documenter ou gérer les exigences. Cependant, ces outils ont leurs limites et ne doivent être utilisés que lorsque les ingénieurs des exigences et les parties prenantes maîtrisent le processus d'ingénierie des exigences et que les exigences sont harmonisées. Dans le cas contraire, il s'agit d'un risque majeur dans le processus d'IE, car ces outils ne soutiennent aucune activité de gestion des exigences.

7.2 Mise en place des Outils

La sélection d'un outil d'IE n'est pas différente de la sélection de n'importe quel autre outil . Vous devez décrire les objectifs, le contexte et les exigences avant de sélectionner et de mettre en œuvre le(s) outil(s) approprié(s).

Les outils ne sont qu'une aide au processus d'ingénierie des exigences et à l'ingénieur des exigences. Ils ne résolvent pas les problèmes organisationnels ou humains. Imaginez que vous souhaitiez, avec vos collègues, documenter les exigences de manière uniforme. Des outils peuvent faciliter cette tâche, par exemple un modèle dans un outil de traitement de texte ou une page wiki. Cela ne garantit pas que tous vos collègues adoptent cette méthode de travail, ni qu'ils aient la discipline nécessaire pour enregistrer et gérer leurs besoins de cette manière. Ce qui peut aider, c'est de conclure des accords entre eux, de vérifier si les accords sont respectés et de pouvoir communiquer entre eux si les accords ne sont pas respectés. Ce n'est pas un outil qui va vous aider. L'introduction d'un outil d'ingénierie des exigences nécessite des responsabilités et des procédures claires en matière d'ingénierie des exigences.

Un outil peut vous aider à configurer votre processus d'ingénierie des exigences de manière efficace et efficiente. Les outils fournissent souvent un cadre basé sur l'expérience des meilleures pratiques. Ces cadres peuvent ensuite être adaptés à la situation.

Comme nous l'avons appris dans les chapitres précédents, les activités principales d'ingénierie des exigences ne sont pas des processus autonomes.

La sélection des outils d'IE appropriés commence par la définition des objectifs et/ou des problèmes que vous souhaitez résoudre dans le cadre du processus d'IE. L'étape suivante consiste à déterminer le contexte du système (dans ce cas, l'ensemble d'outils). Prenez en compte les aspects du contexte (parties prenantes, processus, événements, etc.) et

appliquez vos compétences en matière d'ingénierie des exigences pour spécifier les exigences relatives aux outils d'IE. Mettez en pratique ce que vous prêchez.

Les sections suivantes décrivent certains des aspects qui doivent être pris en compte lors de l'introduction d'un (nouvel) outil d'ingénierie des exigences dans votre organisation.

7.2.1 Prendre en compte tous les coûts du cycle de vie au-delà des coûts de licence

Les coûts les plus évidents, tels que les coûts d'achat ou de licence, sont généralement pris en compte. En outre, des coûts moins visibles doivent également être pris en compte, tels que l'utilisation de ressources pour l'implémentation, le fonctionnement et la maintenance de l'outil.

7.2.2 Considérer les ressources nécessaires

La spécification des exigences et la supervision du processus de sélection requièrent les ressources nécessaires, en plus des coûts mentionnés dans la section précédente. Il ne faut pas négliger les personnes nécessaires pour guider le processus de sélection, les ingénieurs des exigences, les ressources matérielles et les autres ressources. Après la mise en service de l'outil, des ressources peuvent également être nécessaires pour la maintenance et l'assistance aux utilisateurs.

7.2.3 Réduire les risques en menant des projets pilotes

L'introduction d'un nouvel outil peut menacer le contrôle de la base d'exigences actuelle. Un chaos des exigences peut survenir lors de la transition de l'ancienne méthode de travail et/ou des anciens outils vers la nouvelle méthode de travail et les nouveaux outils. L'introduction d'un nouvel outil au cours d'un projet existant conduira irrémédiablement à un retard dans la livraison des exigences et même du projet.

L'introduction d'un nouvel outil, éventuellement assorti d'une méthode de travail différente, doit être testée à petite échelle, là où les risques et l'impact restent gérables. Il y a plusieurs façons de procéder :

- Appliquer l'outil à un projet/système non critique
- Utiliser l'outil de manière redondante avec un projet existant
- Appliquer l'outil à une situation/un projet fictif
- Importer/copier les exigences d'un projet déjà réalisé

Lorsque l'outil répond aux objectifs et aux exigences fixés, il peut être déployé à plus grande échelle au sein de l'organisation ou dans le cadre d'autres projets.

7.2.4 Evaluer l'outil selon des critères définis

Le choix de l'outil approprié peut s'avérer difficile. La vérification approfondie du respect des objectifs et des exigences est une approche standard de l'ingénierie des exigences. Une approche systématique qui évalue l'outil sous différents angles contribue également à faire le bon choix. Les perspectives suivantes peuvent être envisagées :

- **Perspective du projet**

  Ce point de vue met l'accent sur les aspects de gestion de projet. L'outil soutient-il le projet et les informations requises dans le cadre du projet ?

- **La perspective du processus**

  Cette perspective permet de vérifier le soutien du processus d'ingénierie des exigences. L'outil soutient-il suffisamment le processus d'IE ? Peut-il être suffisamment adapté au processus d'IE et à la méthode de travail existants ?

- **Perspective utilisateur**

  Cette perspective permet de vérifier le degré d'application par les utilisateurs de l'outil. Il s'agit d'un point de vue important, car si les utilisateurs ne sont pas satisfaits de l'outil, le risque que l'outil ne soit pas accepté augmente. L'outil prend-il suffisamment en charge l'autorisation des utilisateurs et des groupes ? Est-il suffisamment convivial et intuitif ?

- **Perspective du produit**

  Les fonctionnalités offertes par l'outil sont vérifiées sous cet angle. Les exigences sont-elles suffisamment couvertes par l'outil ? Où sont stockées les données ? Existe-t-il une feuille de route avec les extensions fonctionnelles de l'outil ? L'outil est-il encore pris en charge par le fournisseur pour le moment ?

- **Le point de vue des fournisseurs**

  Dans cette perspective, l'accent est mis sur le service et la fiabilité du fournisseur. Où se trouve le fournisseur ? Comment le support à cet outil est-il organisé ?

- **Perspective économique**

  Cette perspective s'intéresse à l'analyse de rentabilité : l'outil apporte-t-il des avantages suffisants par rapport aux coûts ? Quels sont les coûts (de gestion) pour l'achat et la maintenance ? Qu'apporte l'outil au processus d'IE ? Un contrat de maintenance (séparé) est-il nécessaire ?

- **La perspective de l'architecture**

  Cette perspective évalue la manière dont l'outil s'intègre dans l'organisation (informatique). La technologie utilisée est-elle adaptée à l'organisation ? L'outil peut-il

être suffisamment relié à d'autres systèmes ? L'outil s'intègre-t-il dans le paysage informatique et respecte-t-il les contraintes architecturales ?

7.2.5 Former les employés à l'utilisation de l'outil

Une fois l'outil sélectionné, les utilisateurs doivent se familiariser avec les possibilités que l'outil peut apporter au processus d'ingénierie des exigences. Les utilisateurs – c'est-à-dire les ingénieurs des exigences – doivent être formés à l'utilisation de l'outil dans le cadre du processus d'ingénierie des exigences existant. Si les utilisateurs ne sont pas suffisamment formés, cela peut signifier que tous les avantages de l'outil ne sont pas utilisés. En effet, il est possible que l'outil soit mal utilisé, avec toutes les conséquences que cela implique.

Le processus d'ingénierie des exigences peut également être modifié en fonction de l'outil choisi. Des aspects du processus d'ingénierie des exigences qui n'étaient pas possibles auparavant peuvent être rendus possibles grâce à un nouvel outil : par exemple, une gestion adéquate des versions, la modélisation des exigences, etc. Cela peut signifier que de nouvelles procédures sont convenues, que des modèles sont adaptés ou appliqués, que des changements sont apportés à la méthode de travail, etc. L'implication de l'ingénieur des exigences dans ce changement contribue au succès de l'acceptation de l'outil.

7.3 Pour en savoir plus

La littérature suivante peut être consultée pour une vue d'ensemble des outils disponibles et des évaluations d'outils. Juan M. Carrillo de Gea et. al. donnent un aperçu complet du rôle des outils d'ingénierie des exigences [dGeA2011].

L'article de Barbara Kitchenham, Stephen Linkman, David Law [KiLL1997] décrit et valide une méthode d'évaluation systématique des outils. Si vous êtes à la recherche d'un outil d'ingénierie des exigences, vous trouverez une liste complète de ces outils sur le site Web de Volere [Vole2026] ou à l'adresse [BiHe2020].

8 Références

[Alex2005] Ian F. Alexander: A Taxonomy of Stakeholders – Human Roles in System Development. International Journal of Technology and Human Interaction 2005, 1(1), 23–59.

[AnPC1994] Annie I. Antón, W. Michael McCracken, Colin Potts: Goal Decomposition and Scenario Analysis in Business Process Reengineering. CAiSE (Conference on Advanced Information Systems Engineering), 1994, 94–104.

[Armo2004] Philip G. Armour. Les lois du processus logiciel : Un nouveau modèle pour la production et la gestion des logiciels. Boca Raton, Fl.: CRC Press, 2004.

[Axelos2019] Axelos : ITIL Foundation : ITIL 4 Edition. Axelos Ltd, 2019.

[BaBo2014] Stéphane Badreau, Jean-Louis Boulanger : Ingénierie des Exigences. Paris : Dunod, 2014.

[BeeA2001] Kent Beck et al : Principles behind the Agile Manifesto. http://agilemanifesto.org/principles.html, 2001. Dernière visite en novembre 2025.

[BiHe2020] Andreas Birk, Gerald Heller : Liste des outils de gestion des exigences. https://makingofsoftware.com/resources/list-of-rm-tools/ 2020, Dernière visite en novembre 2025.

[Boeh1981] Barry W. Boehm : Software Engineering Economics, Englewood Cliffs, New Jersey : Prentice Hall, 1981.

[BoRJ2005] Grady Booch, James Rumbaugh, Ivar Jacobson : The Unified Modeling Language User Guide, 2nd edition. Reading, MA: Addison-Wesley, 2005.

[Bour2009] Lynda Bourne: Stakeholder Relationship Management – A Maturity Model for Organisational Implementation. Farnham : Gower, 2009.

[Bowe2009] Glenn A. Bowen: Document Analysis as a Qualitative Research Method. Qualitative Research Journal, 2009, vol. 9, no. 2, pp. 27–40.

[BuHe2024] Stan Bühne, Andrea Herrmann : Manuel de gestion des exigences selon la norme IREB – Éducation et formation pour la qualification IREB Certified Professional for Requirements Engineering Niveau avancé Gestion des exigences. Version 2.1. Karlsruhe: IREB. https://cpre.ireb.org/downloads-and-resources/downloads#cpre-requirements-management-handbook. Dernière visite en février 2025.

[CaDJ2014] Dante Carrizo, Oscar Dieste, Natalia Juristo: Systématisation de la sélection des techniques d'identification des exigences. Information and Software Technology 2014, 56(6), 644–669.

[Chen1976] Peter P.-S. Chen : The Entity-Relationship Model : Toward a Unified View of Data, ACM Transactions on Database Systems 1976, 1(1), 9–36.

[ClGZ2012] Jane Cleland-Huang, Olly Gotel, Andrea Zisman (eds.) : Software and Systems Traceability. Londres : Springer, 2012.

[CLSZ2021] Jan Jaap Cannegieter, Hans van Loenhoud, Stefan Staal, Johan Zandhuis: Basiskennis requirements: IREB CPRE foundation examenstof uitgelegd en praktisch gemaakt. Utrecht : EBURON, 2021 (en néerlandais).

[Cock2001] Alistair Cockburn: Writing Effective Use Cases. Boston: Addison-Wesley, 2001.

[Cohn2004] Mike Cohn: User Stories Applied: For Agile Software Development. Boston: Addison-Wesley, 2004.

[Cohn2010] Mike Cohn : Réussir avec Agile : Développement de logiciels avec Scrum. Upper Saddle River, NJ : Addison-Wesley, 2010.

[CoWe1998] Reidar Conradi, Bernhard Westfechtel : Version Models for Software Configuration Management (Modèles de version pour la gestion de la configuration des logiciels). ACM Computing Surveys 1998, 30(2), 232-282.

[DaTW2012] Marian Daun, Bastian Tenbergen, Thorsten Weyer : Point de vue sur les exigences. En : K. Pohl, H. Hönninger, R. Achatz, M. Broy : Model-Based Engineering of Embedded Systems, Heidelberg : Springer, 2012.

[Davi1993] Alan M. Davis: Software Requirements – Objects, Functions, and States. 2e édition, Englewood Cliffs, New Jersey : Prentice Hall, 1993.

[Davi1995] Alan M. Davis : 201 Principes du développement de logiciels. New York : McGraw-Hill, 1995.

[Davi2005] Alan M. Davis: Just Enough Requirements Management – Where Software Development Meets Marketing. New York: Dorset House, 2005.

[DeBo2005] Edward De Bono : De Bono's Thinking Course (édition révisée), Barnes & Noble Books, 2005.

[DeCo2007] Conseil du design : 11 leçons : Une étude du processus de conception. https://www.designcouncil.org.uk/resources/report/11-lessons-managing-design-global-brands, 2007. Dernière visite en novembre 2025.

[dGeA2011] Juan M. Carrillo de Gea, Joaquín Nicolás, José L. Fernandez-Alemán, Ambrosio Toval, Christof Ebert, Aurora Vizcaíno : Outils d'ingénierie des exigences. IEEE Software 2011, 28(4), 86-91.

[DeMa1978] Tom DeMarco: Structured Analysis and System Specification. New York: Yourdon Press, 1978.

[DIN66001] DIN 66001:1983-12 : Traitement de l'information ; symboles graphiques et leur application. Deutsches Institut für Normung e.V., Berlin, 1983 (en allemand).

[Eber2022] Christof Ebert: Systematisches Requirements Engineering: Anforderungen ermitteln, dokumentieren, analysieren und verwalten, 7. Auflage. Heidelberg : dpunkt 2022 (en allemand).

[Fowl1996] Martin Fowler: Analysis Patterns: Reusable Object Models. Reading, MA: Addison-Wesley, 1996.

[FLCC2016] Xavier Franch, Lidia Lopez, Carlos Cares, Daniel Colomer. (2016). Le cadre i* pour la modélisation orientée vers les objectifs. Dans Domain Specific Conceptual Modeling, Springer, 485-506.

[Fugg1993] Alfonso Fuggetta: A Classification of CASE Technology. IEEE Computer 1993, 26(12), 25-38.

[GaWe1989] Donald C. Gause et Gerald M. Weinberg : Exploration des exigences : La qualité avant la conception. New York: Dorset House, 1989.

[GFPK2010] Tony Gorschek, Samuel Fricker, Kenneth Palm, and Steven A. Kunsman : Un processus d'innovation léger pour le développement de produits à forte intensité logicielle. IEEE Software 2010, 27(1), 37-45.

[GGJZ2000] Carl A. Gunter, Elsa L. Gunter, Michael Jackson, Pamela Zave : Un modèle de référence pour les exigences et les spécifications. IEEE Software 2000, 17(3), 37-43.

[GHJV1994] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides : Design Pattern – Elements of Reusable Object-Oriented Software. Reading, Mass : Addison-Wesley, 1994.

[Gilb1988] Tom Gilb : Principes de gestion du génie logiciel. Reading, Mass : Addison Wesley, 1988.

[Glas1999] Friedrich Glasl: Confronting Conflict – A First-Aid Kit for Handling Conflict. Stroud, Gloucestershire: Hawthorn Press, 1999.

[GlFr2015] Martin Glinz et Samuel A. Fricker : Sur la compréhension partagée dans le génie logiciel : An Essay. Computer Science – Research and Development 2015, 30(3-4), 363-376.

[Glin2007] Martin Glinz : Sur les exigences non fonctionnelles. 15th IEEE International Requirements Engineering Conference, Delhi, India, 2007, 21-26.

[Glin2008] Martin Glinz : Une approche des exigences de qualité basée sur le risque et orientée vers la valeur. IEEE Software 2008, 25(2), 34-41.

[Glin2016] Martin Glinz : De quelle quantité d'ingénierie des exigences avons-nous besoin ? Softwaretechnik-Trends 2016, 36(3), 19-21.

[Glin2019] Martin Glinz: Ingénierie des exigences I. Course Notes, University of Zurich, 2019. https://www.ifi.uzh.ch/en/rerg/courses/archives/hs19/re-i.html#resources. Dernière visite en novembre 2025.

[Glin2025] Martin Glinz: A Glossary of Requirements Engineering Terminology. Version 2.2. https://cpre.ireb.org/downloads-and-resources/downloads#cpre-glossary, Dernière visite en février 2026.

[GlWi2007] Martin Glinz et Roel Wieringa : Stakeholders in Requirements Engineering (Introduction des éditeurs invités). IEEE Software 2007, 24(2), 18–20.

[GoFi1994] Orlena Gotel, Anthony Finkelstein: An Analysis of the Requirements Traceability Problem. 1st International Conference on Requirements Engineering, Colorado Springs, 1994, 94–101.

[Good1961] Leo A. Goodman: "Snowball sampling". Annals of Mathematical Statistics. 1961, 32 (1): 148–170.

[GoRu2003] Rolf Goetz, Chris Rupp: Psychotherapy for System Requirements. 2nd IEEE International Conference on Cognitive Informatics (ICCI'03), London, 2003, 75–80.

[Gott2002] Ellen Gottesdiener: Requirements by Collaboration: Workshops for Defining Needs, Boston: Addison-Wesley Professional, 2002.

[GreA2017] Eduard C. Groen, Norbert Seyff, Raian Ali, Fabiano Dalpiaz, Joerg Doerr, Emitzá Guzmán, Mahmood Hosseini, Jordi Marco, Marc Oriol, Anna Perini, Melanie Stade : The Crowd in Requirements Engineering – The Landscape and Challenges (La foule dans l'ingénierie des exigences – le paysage et les défis). IEEE Software 2017, 34(2), 44–52.

[Greg2016] Sarah Gregory : "It Depends" : Heuristique pour les exigences "suffisamment communes". Discours liminaire au REFSQ 2016, Essen, Allemagne, 2016.

[GRL2020] Goal oriented Requirement Language. University of Toronto, Canada https://www.cs.toronto.edu/km/GRL. Dernière visite en novembre 2025.

[GrSe2005] Paul Grünbacher, Norbert Seyff : Négociation des exigences. En A. Aurum, C. Wohlin (eds.) : Engineering and Managing Software Requirements. Berlin: Springer, 2005, 143–162.

[Hare1988] David Harel. On Visual Formalisms. Communications of the ACM 1988, 31(5), 514–530.

[HoSch2020] Stefan Hofer, Henning Schwentner: Domain Storytelling — A Collaborative Modeling Method. Disponible auprès de Leanpub, http://leanpub.com/domainstorytelling. Dernière visite en novembre 2025.

[HuJD2011] Elizabeth Hull, Ken Jackson, Jeremy Dick : L'ingénierie des exigences. 3rd ed., Berlin: Springer: 2011.

[Hump2017] Aaron Humphrey : Personas d'utilisateurs et profils de médias sociaux. Persona Studies 2017, 3(2), 13–20.

[IEEE830] Pratique recommandée de l'IEEE pour les spécifications des exigences logicielles. IEEE Std 830-1998, 1998.

[ISO19650] ISO 19650. Organization and Digitization of Information about Buildings and Civil Engineering Works, including Building Information Modelling (BIM)–

Information Management Using Building Information Modelling – Part 1 and 2, 2018.

[ISO5807] ISO/IEC/IEEE 1985-02 : Information processing; Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts. International Organization for Standardization, Geneva, 1985.

[ISO25010] ISO/IEC/IEEE 25010:2023: Systems and software Quality Requirements and Evaluation (SQuaRE) – Product quality model. International Organization for Standardization, Geneva, 2023.

[ISO29148] ISO/IEC/IEEE 29148: Systems and Software Engineering – Life Cycle Processes – Requirements Engineering. International Organization for Standardization, Geneva, 2018.

[Jack1995] Michael Jackson: Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices. New York : ACM Press, 1995.

[Jack1995b] Michael Jackson: The World and the Machine. 17th International Conference on Software Engineering 1995 (ICSE 1995), 287–292.

[Jaco1992] Ivar Jacobson : Ingénierie logicielle orientée objet : une approche basée sur les cas d'utilisation. New York : ACM Press, 1992.

[JaSB2011] Ivar Jacobson, Ian Spence, Kurt Bittner: Use Case 2.0: The Guide to Succeeding with Use Cases. Ivar Jacobson International SA, 2011.

[KiLL1997] Barbara Kitchenham, Stephen Linkman, David Law : DESMET : Une méthodologie pour l'évaluation des méthodes et des outils de génie logiciel. Computing & Control Engineering Journal 1997, 8(3), 120–126.

[KSTT1984] Noriaki Kano, Nobuhiku Seraku, Fumio Takahashi, Shinichi Tsuji : Qualité attrayante et qualité indispensable. Hinshitsu (Quality – Journal of the Japanese Society for Quality Control) 1984, 14(2), 39–48 (in Japanese).

[Laue2002] Søren Lauesen : Exigences logicielles : Styles et techniques. London: Addison-Wesley, 2002.

[LaWE2001] Brian Lawrence, Karl Wiegers, et Christof Ebert : The Top Risks of Requirements Engineering. IEEE Software 2001, 18(6), 62–63.

[LiOg2011] Jeanne Liedtka, Tim Ogilvie: Designing for Growth: A Design Thinking Tool Kit for Managers. New York: Columbia University Press, 2011.

[LiSS1994] Odd I. Lindland, Guttorm Sindre, Arne Sølverg : Comprendre la qualité dans la modélisation conceptuelle. IEEE Software 1994, 11(2), 42–49.

[LiSZ1994] Horst Lichter, Matthias Schneider-Hufschmidt, Heinz Züllighoven : Le prototypage dans les projets logiciels industriels – Combler le fossé entre la théorie et la pratique. IEEE Transactions on Software Engineering 1994, 20 (11) : 825–832.

[LiQF2010] Soo Ling Lim, Daniele Quercia, Anthony Finkelstein : StakeNet : Utilisation des réseaux sociaux pour analyser les parties prenantes des projets logiciels à grande échelle. 32nd International Conference on Software Engineering (ICSE 2010), 2010, 295–304.

[MaGR2004] Neil Maiden, Alexis Gizikis, Suzanne Robertson : Provoquer la créativité : Imaginez ce que vos exigences pourraient être. IEEE Software 2004, 21(5), 68–75.

[Math2019] Joseph Mathenge : Change Control Board vs Change Advisory Board : Quelle est la différence ? https://www.bmc.com/blogs/change-control-board-vs-change-advisory-board, Nov. 22, 2019. Dernière visite en novembre 2025.

[McIn2016] John McIntyre : Modèles MoSCoW ou Kano – Comment prioriser ? https://www.hotpmo.com/management-models/moscow-kano-prioritize, Oct. 20, 2016. Dernière visite en novembre 2025.

[MNJR2016] Walid Maalej, Maleknaz Nayebi, Timo Johann, et Guenther Ruhe : Vers une ingénierie des exigences guidée par les données. IEEE Software 2016, 33(1), 48–54.

[Moor2014] Christopher W. Moore : The Mediation Process – Practical Strategies for Resolving Conflicts, 4e édition. Hoboken, NJ: John Wiley & Sons, 2014.

[MWHN2009] Alistair Mavin, Philip Wilkinson, Adrian Harwood, and Mark Novak : Easy Approach to Requirements Syntax (EARS). 17th IEEE International Requirements Engineering Conference (RE'09), Atlanta, Georgia, 2009, 317–322.

[NuKF2003] Bashar Nuseibeh, Jeff Kramer, Anthony Finkelstein : Points de vue : Les relations sérieuses sont difficiles ! 25th International Conference on Software Engineering (ICSE'03), Portland, Oregon, 2003, 676–681.

[OleA2018] K. Olsen et al : Certified Tester, Foundation Level Syllabus – Version 2018. International Software Testing Qualifications Board, 2018.

[Olso2014] David Olson : Matrice de priorisation. http://www.bawiki.com/wiki/Matrix-Prioritization.html, 2014. Dernière visite en novembre 2025.

[OMG2013] Object Management Group: Business Process Model and Notation (BPMN), version 2.0.2. OMG document, formal/2013-12-09. https://www.omg.org/spec/BPMN/. Dernière visite en novembre 2025.

[OMG2017] Object Management Group: OMG Unified Modeling Language (OMG UML), version 2.5.1. OMG document, formal/2017-12-05. https://www.omg.org/spec/UML/About-UML/. Dernière visite en novembre 2025.

[OMG2018] Object Management Group: OMG Systems Modeling Language (OMG SysML™), version 1.6. OMG document, ptc/2018-12-08.

https://www.omg.org/spec/SysML/About-SysML/. Dernière visite en novembre 2025.

[Osbo1948] Alex F. Osborn: Your Creative Power: How to Use Imagination. C. Scribner's Sons, 1948. (Accessible en tant que réimpression numérique : Read Books Ltd. (epub eBook), April 2013).

[Pich2010] Roman Pichler : Agile Product Management with Scrum – Creating Products that Customers Love, Boston : Addison-Wesley, 2010.

[Pohl2010] Klaus Pohl: Requirements Engineering: Fundamentals, Principles, and Techniques. Berlin-Heidelberg: Springer, 2010.

[PoRu2015] Klaus Pohl, Chris Rupp: Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam, (2nd ed). Rocky Nook, Santa Barbara, 2015.

[PoRu2021] Klaus Pohl, Chris Rupp: Basiswissen Requirements Engineering, 5. Auflage. Heidelberg: dpunkt.verlag, 2021 (in German).

[Rein1997] Donald G. Reinertsen: Managing the Design Factory – A Product Developer's Toolkit. The Free Press, 1997.

[Rein2009] Donald G. Reinertsen : Les principes du flux de développement de produits : le développement de produits allégés de deuxième génération. Redondo Beach, Californie : Celeritas Publishing, 2009.

[Ries2011] Eric Ries: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. New York : Crown Business, 2011.

[Robe2001] S. Ian Robertson: Problem Solving. Hove, East Sussex : Psychology Press, 2001.

[RoRo2012] Suzanne Robertson and James Robertson: Mastering the Requirements Process: Getting Requirements Right. 3rd edition. Boston: Addison-Wesley, 2012.

[RuJB2004] James Rumbaugh, Ivar Jacobson, Grady Booch : The Unified Modeling Language Reference Manual, 2e édition. Reading, MA: Addison Wesley, 2004.

[Rupp2020] Chris Rupp: Requirements-Engineering und Management, 7. Auflage. München: Hanser, 2020 (en allemand).

[SoSa1998] Ian Sommerville et Pete Sawyer : Requirements Engineering : A Good Practice Guide. Chichester: John Wiley & Sons, 1997.

[SwBa1982] William Swartout et Robert Balzer : On the Inevitable Intertwining of Specification and Implementation. Communications of the ACM 1982, 25(7), 438-440.

[Verd2014] Dave Verduyn : Discovering the Kano Model, in : Kano model, https://www.kanomodel.com/discovering-the-kano-model, 2014. Dernière visite en novembre 2025.

[vLam2009] Axel van Lamsweerde: Requirements Engineering: From System Goals to UML Models to Software Specifications. Chichester: John Wiley & Sons, 2009.

[Vole2026] Volere Requirements Resources: https://www.volere.org. Dernière visite en février 2026.

[WiBe2013] Karl Wiegers et Joy Beatty : Exigences logicielles. 3rd edition. Redmond, Wa.: Microsoft Press, 2013.

[Wieg1999] Karl E. Wiegers : Chaque chose en son temps : Priorité aux exigences. https://www.processimpact.com/articles/prioritizing.pdf, 1999. Dernière visite en novembre 2025.

[ZoCo2005] Didar Zowghi, Chad Coulin : Requirements Elicitation : A Survey of Techniques, Approaches, and Tools. En A. Aurum, C. Wohlin (eds.) Engineering and Managing Software Requirements. Berlin: Springer, 2005, 19–46.