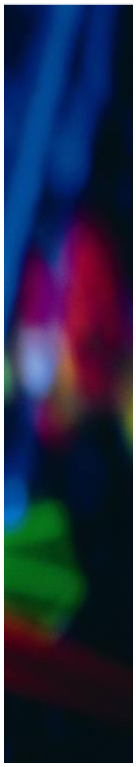
Proposition d'un environnement numérique dédié à la fouille et à la synthèse collaborative d'exigences en ingénierie de produits

Romain Pinquié



2016-ENAM-0030

École doctorale n° 432 : Sciences des Métiers de l'ingénieur

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École Nationale Supérieure d'Arts et Métiers

Spécialité " Conception "

présentée et soutenue publiquement par

Romain PINQUIÉ

07 octobre 2016

Proposition d'un environnement numérique dédié à la fouille et à la synthèse collaborative d'exigences en ingénierie de produits.

Directeur de thèse : **Philippe VÉRON**
Co-encadrement de la thèse : **Frédéric SEGONDS**

Jury

M. Michel TOLLENAERE, Professeur des Universités, G-SCOP, INP Grenoble	Président
Mme. Isabelle MIRBEL, Maître de Conférences HDR, I3S/INRIA, Université de Nice	Rapporteur
M. Abdelaziz BOURAS, Professeur des Universités, College of Engineering, Qatar University	Rapporteur
M. Philippe VÉRON, Professeur des Universités, LSIS, Arts et Métiers ParisTech	Examinateur
M. Frédéric SEGONDS, Maître de Conférences, LCPI, Arts et Métiers ParisTech	Examinateur
M. Nicolas CROUÉ, Ingénieur, Keonys	Invité
M. Bruno PUECHOULTRES, Ingénieur, Keonys	Invité

T H È S E

Arts et Métiers ParisTech - Campus d'Aix-en-Provence
Laboratoire des Sciences de l'Information et des Systèmes – LSIS UMR CNRS 7296

Il suffit d'ouvrir les yeux pour voir que les conquêtes de l'industrie qui ont enrichi tant d'hommes pratiques n'auraient jamais vu le jour si ces hommes pratiques avaient seuls existé, et s'ils n'avaient été devancés par des fous désintéressés qui sont morts pauvres, qui ne pensaient jamais à l'utile, et qui pourtant avaient un autre guide que leur caprice. C'est que, comme l'a dit Mach, ces fous ont économisé à leurs successeurs la peine de penser. Ceux qui auraient travaillé uniquement en vue d'une application immédiate n'auraient rien laissé derrière eux et, en face d'un besoin nouveau, tout aurait été à recommencer.

— Science et méthode, Henri Poincaré

Remerciements

Mes premiers remerciements sont adressés à Michel Tollenaere, Professeur des Universités à l'Institut National Polytechnique de Grenoble, lequel m'a fait l'honneur de présider ce jury de thèse. Michel, j'espère que nous aurons l'occasion de nous revoir très prochainement au sein de la communauté nationale, voire de collaborer autour de problématiques communes.

Je remercie également les deux rapporteurs de ces travaux de thèse : Isabelle Mirbel, Maître de Conférences HDR à l'Université Nice-Sophia-Antipolis, et Abdelaziz Bouras, Professeur des Universités à Qatar University. Isabelle, Aziz, merci à tous les deux d'avoir respectivement accepté de délaisser les plages qui bordent la Côte d'Azur et le golfe Persique pendant quelques jours, et ce, pour vous astreindre à une lecture minutieuse de mon manuscrit. Merci pour vos rapports, vos questions, et vos idées qui nourrissent ma réflexion et éveillent de nouvelles perspectives de recherche. Nul doute que nos chemins se recroiseront !

Je remercie mon directeur de thèse, Philippe Véron, Professeur des Universités aux Arts et Métiers ParisTech d'Aix-en-Provence. Philippe, merci de m'avoir accordé ta confiance que j'espère avoir gagné en m'investissant pleinement dans ce projet. Merci de m'avoir soutenu quand quelques nuages gris ont pointé le bout de leur nez. Je te dis aussi merci de m'avoir toujours permis d'assouvir ma curiosité intellectuelle que ce soit à l'AFIS, au PLM Lab, avec Patrice, ou comme enseignant intérimaire. Merci pour tout, ce fut une très belle expérience à tes côtés.

Je remercie également mon encadrant de thèse, Frédéric Segonds, Maître de Conférences aux Arts et Métiers ParisTech de Paris. Frédéric, merci d'avoir encadré cette thèse avec un regard géographiquement éloigné, mais indéniablement attentionné. Je te remercie de m'avoir fait part de tes précieux conseils fraichement acquis, ainsi que d'avoir relu attentivement tous mes écrits jusqu'au sprint final. Merci aussi de m'avoir fait découvrir de nouveaux horizons : le LCPI, le PLM Lab, etc. Il nous reste encore du papier à gribouiller et j'espère quelques idées à creuser.

Merci à l'entreprise Keonys de m'avoir confié une problématique de recherche et les moyens de penser une solution. Merci de m'avoir placé au-delà de ma zone de confort, cela m'a permis d'explorer de nouvelles théories et technologies qui, j'en suis certain, trouveront leur place dans mes expériences futures. Nicolas, merci pour ta bienveillance, tes encouragements, et ton implication totale dans cette thèse. Merci Bruno pour ta bonne humeur virale, tes nombreuses anecdotes nourrissantes, et tes myriades d'idées pétillantes. Merci aussi à toute l'équipe toulousaine : Vincent, Frédéric, Matthieu, Olivier, Olivier, Sami, etc.

Je voudrais aussi remercier Lionel Roucoules, Professeur des Universités et responsable de l'équipe Ingénierie Numérique des Systèmes Mécaniques. Lionel, merci de m'avoir chaleureusement accueilli pour cette formation à la recherche et par la recherche. Je garde un excellent souvenir de nos échanges informels et de notre périple au Qatar. De manière plus exhaustive, je remercie tous les membres de l'équipe INSM, ainsi que mes camarades thésards pour leur bonne humeur quotidienne. Je vous souhaite à tous de prendre beaucoup de plaisir dans la suite de vos recherches.

Ces trois années de thèse furent aussi ponctuées de rencontres toutes aussi inattendues qu'intellectuellement riches. Patrice, je pense bien entendu à toi particulièrement. Merci sincèrement d'avoir répondu à ma bouteille à la mer. Merci d'avoir levé ce masque qui, jusqu'à notre rencontre, faisait que je ne voyais qu'une face déguisée de l'ingénierie système. Aujourd'hui, grâce à tous nos échanges, oraux et écrits, je crois avoir enfin saisi les fondamentaux, souvent bafoués, d'une méthode passionnante. J'espère que nous continuerons à avoir une réflexion passionnée commune.

Fidèle à mes origines Pyrénéennes, je dois dire que le chemin vers le sommet a été long, sinueux, et ponctué de rencontres qui ont façonné mon goût pour les sciences et les technologies. Conformément à la flèche du temps, je remercie d'abord tous mes professeurs du groupe mathématiques appliquées et informatique de Cranfield University, lesquels m'ont initié à l'ingénierie numérique et à la recherche académique. Merci également à Chris et Jacek Stecki de m'avoir donné l'opportunité de découvrir deux territoires inconnus : le Prognostics & Health Management et *a land of wide open spaces* ..., l'Australie. Merci à tous mes anciens collègues du LISMMA de Supméca Paris, en particulier Mireille, Olivia et Pierre, sans lesquels le moment présent serait certainement différent.

Enfin, je remercie mes parents et mon frère de m'avoir donné tous les moyens matériels, financiers et affectifs pour réaliser un parcours académique international plein d'opportunisme. Merci également à tout le reste de la famille et à mes amis d'enfance.

Romain Pinquié

Table des matières

Table des figures

Première partie

PROBLÉMATIQUE

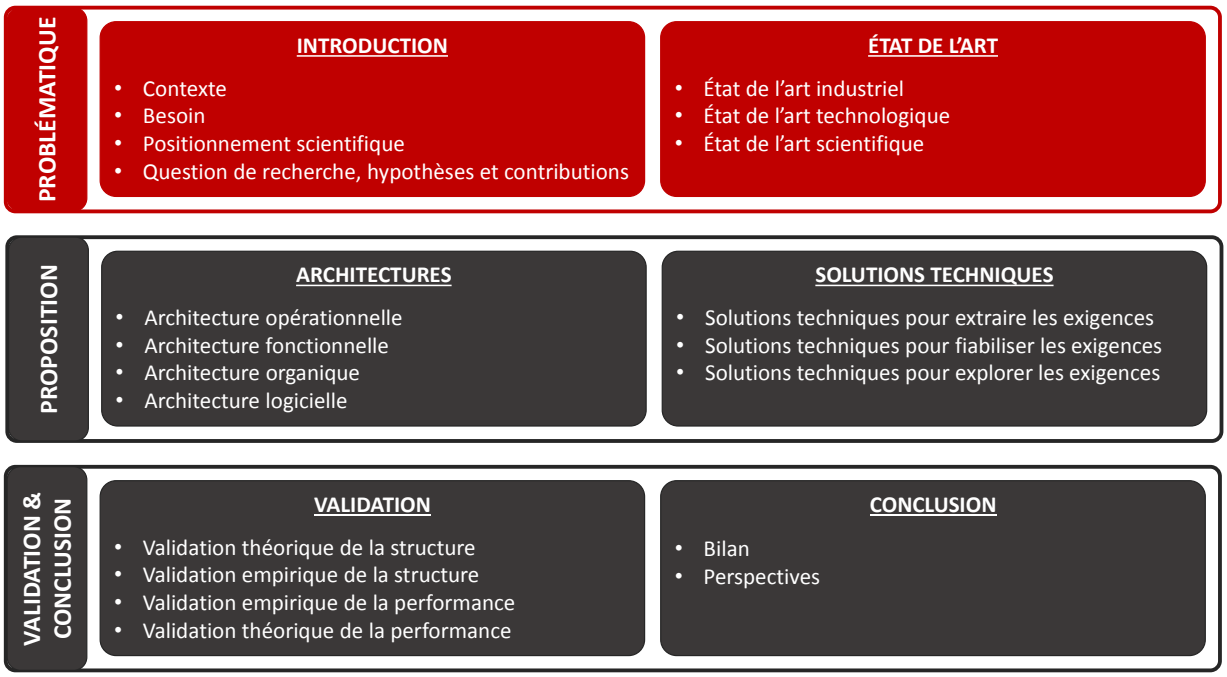


FIGURE 1 – 1ère partie du manuscrit dédiée à la problématique de recherche.

Comme le montre la figure 1, ce manuscrit de thèse est scindé en trois grandes parties : I la problématique, II la proposition, et III la validation et la conclusion.

Dans cette partie I, le chapitre 1 est une introduction au contexte général et au besoin que notre proposition cherche à satisfaire. La place qu'occupent nos travaux au sein de la communauté scientifique y est aussi discutée avant d'introduire notre question de recherche, nos hypothèses et nos contributions.

Le chapitre 2 est un état de l'art général des solutions existantes relatives à notre problématique. Cet état de l'art est réalisé selon trois perspectives : industrielle, technologique et scientifique. L'état de l'art industriel est une revue de l'ingénierie des exigences telle qu'elle est normalisée et pratiquée par les entreprises. L'état de l'art technologique, lui, est un résumé des principaux outils commerciaux qui supportent l'ingénierie des exigences. Pour terminer, l'état de l'art scientifique synthétise et critique les contributions académiques liées à notre problématique.

Chapitre 1

INTRODUCTION

1.1 Le contexte

La révolution numérique

L'informatique, cette techno-science que les américains qualifient de « *General-purposes technology* » – technologie à multi-usages – peut être considérée comme le *primum movens* – premier moteur – d'une grande majorité de nos sciences. L'application de l'informatique aux sciences du vivant, la biologie, a par exemple donné naissance à la bio-informatique.

La théorie de l'information développée par Shannon [1948] est le principal fait qui a conduit à ce changement de paradigme. En effet, avant les années 50, l'information était indissociable de son support physique. Un texte était écrit dans un livre. Un son était gravé sur un disque vinyle. Une photo était imprimée sur un papier argentique. Dans les années 1950, Claude Shannon définissait l'idée de numérisation systématique, laquelle devait nous permettre de s'affranchir du type de support utilisé pour conserver une information. En remplaçant l'atome par le bit, la théorie de l'information a initié une transformation profonde et rapide d'une grande partie de la société. Certains parlent même de « révolution numérique » [Isaacson, 2014].

Cette numérisation systématique de l'information a déjà engendré des bouleversements considérables. Dans l'audio-visuel, le MP3 a remplacé les disques vinyles. Le commerce se fait désormais « en ligne ». Nous naviguons avec l'assistance de cartes interactives et coordonnées GPS. Les moyens de transport se robotisent. Les industries conçoivent, fabriquent, maintiennent et recyclent leurs produits grâce à l'ingénierie assistée par ordinateur. Les sciences cherchent à découvrir les lois fondamentales de la nature via des expérimentations numériques. La médecine exploite des modèles 3D fonctionnels de nos organes vitaux, mais aussi des patients dont la réalité virtuelle, sinon augmentée, les rend aussi vrais que nature.

Une fois numérisés, tous ces objets prennent la forme unique de suites de nombres. La représentation binaire, faite de 0 et de 1, est une représentation pratique quand il s'agit d'exploiter ces numérisations dans les machines à informations qui nous entourent. Emprunté à Berry [2008], le concept de « machine à informations » est préféré au terme « ordinateur » pour désigner l'ensemble des circuits et logiciels qui peuplent les objets connectés avec lesquels nous interagissons quotidiennement.

La numérisation du cycle de vie d'un produit

Notre contexte de recherche, celui de l'ingénierie de produits, n'a pas échappé à la numérisation systématique des informations. Nous sommes passés d'artisans capables de concevoir, fabriquer, maintenir et recycler des produits unitaires simples à une division du travail qui, comme l'a écrit Adam Smith en 1776 [Smith, 1776], a permis une production de masse avec des ressources limitées : « *(...) enfin l'important travail de faire une épingle est divisé en dix-huit opérations distinctes ou environ, lesquelles, dans certaines fabriques, sont remplies par autant de mains différentes, quoique dans d'autres le même ouvrier en remplisse deux ou trois. J'ai vu une petite manufacture de ce genre qui n'employait que dix*

ouvriers, et où par conséquent quelques-uns d'eux étaient chargés de deux ou trois opérations. Mais, quoique la fabrique fût fort pauvre et, par cette raison, mal outillée, cependant, quand ils se mettaient en train, ils venaient à bout de faire entre eux environ douze livres d'épingles par jour : or, chaque livre contient au delà de quatre mille épingles de taille moyenne. Ainsi ces dix ouvriers pouvaient faire entre eux plus de quarante-huit milliers d'épingles dans une journée ; donc chaque ouvrier, faisant une dixième partie de ce produit, peut être considéré comme faisant dans sa journée quatre mille huit cents épingles. Mais s'ils avaient tous travaillé à part et indépendamment les uns des autres, et s'ils n'avaient pas été façonnés à cette besogne particulière, chacun d'eux assurément n'eût pas fait vingt épingles, peut-être pas une seule, dans sa, journée, c'est-à-dire pas, à coup sûr, la deux cent quarantième partie, et pas peut-être la quatre mille huit centième partie de ce qu'ils sont maintenant en état de faire, en conséquence d'une division et d'une combinaison convenables de leurs différentes opérations. ».

La division du travail a, certes, augmenté la productivité, mais elle a aussi cloisonné les échanges d'informations au sein des divisions. C'est à partir de la fin des années 50 que l'essor de la puissance de calcul des machines à information a permis de numériser les produits et les processus industriels. Les maquettes physiques faites de matières premières telles que le bois, la glaise ou l'acier ont ainsi été remplacées par des maquettes numériques faites de 0 et 1. Les tests au cours desquels le produit est soumis à des sollicitations comportementales extrêmes – des tests de collisions, par exemple –, ont été remplacés par des descriptions mathématiques, des modèles, dont la simulation virtualise le comportement du produit. La spécification, la conception, la fabrication, l'assemblage, la maintenance, le démantèlement, le recyclage, toutes les phases du cycle de vie d'un produit ont été numérisées à tel point qu'on puisse parler de « jumeau numérique » (*digital twin*, en anglais) [Boschert and Rosen, 2016]. Cette numérisation systématique des objets techniques a permis de mettre en place une nouvelle démarche de travail outillée : la gestion du cycle de vie du produit ou, en anglais, *Product Lifecycle Management* (PLM) [Garetti and Terzi, 2005, Terzi et al., 2010]. Les disciplines travaillent désormais simultanément, et l'intégration des expertises se fait via une collaboration autour d'un patrimoine informationnel commun, lequel facilite la capitalisation et l'échange des informations numériques tout au long du cycle de vie du produit.

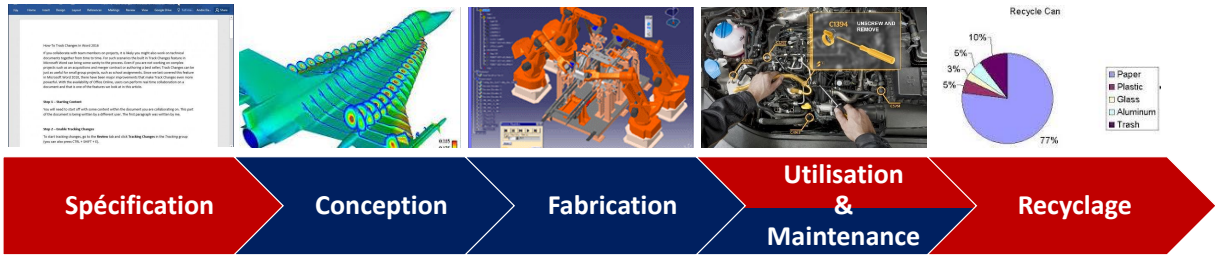


FIGURE 1.1 – Inégalités vis-à-vis de la sophistication (rouge = faible, bleu = élevé) des technologies mises en œuvre tout au long du cycle de vie d'un produit.

Une numérisation inégalitaire de l'ingénierie

La vision idéale, voire rêveuse, d'une numérisation absolue de l'ingénierie que l'on identifie parfois comme l'« industrie 4.0 », laquelle est largement appuyée par les incitatifs économiques des éditeurs de logiciels et des consultants, est toutefois marquée d'inégalités fortes. À ce jour, hormis quelques exceptions, il n'y a que les informations créées et gérées par les génies – mécanique, électrique, civil, informatique, etc. –, historiquement maîtrisés par des ingénieurs, qui exploitent pleinement cette révolution numérique. À l'inverse, la figure 1.1 montre que les phases situées en amont et en aval du cycle de vie, parmi lesquelles l'ingénieur n'a jamais joué un rôle prépondérant, ne sont qu'aux prémisses du numérique. Par exemple, la numérisation des objets techniques manipulés par l'activité d'ingénierie des exigences ne se limite qu'à un simple passage de documents papiers à des documents numériques. À l'inverse, les concepteurs s'appuient sur des représentations numériques multiples et sophistiquées telles

que : les modèles géométriques 3D, les simulations comportementales multi-physiques, les environnements immersifs de réalité virtuelle ou augmentée ou les deux.

Des inégalités injustifiées

L'enjeu d'outiller des phases comme l'ingénierie des exigences est d'autant plus important qu'elles ont un impact considérable sur les coûts du produit. La figure 1.2 illustre l'évolution des coûts engagés pour corriger une erreur en fonction des phases du cycle de vie. On lit que le coût de correction d'une erreur en phase de conception est 3 à 6 fois plus important qu'en phase de spécification, et qu'il est respectivement multiplié par 20 à 100 fois et 500 à 1000 fois en phases de développement et de production. Par ailleurs, on peut lire que 70 % des coûts du cycle de vie sont engagés dès la phase de concept.

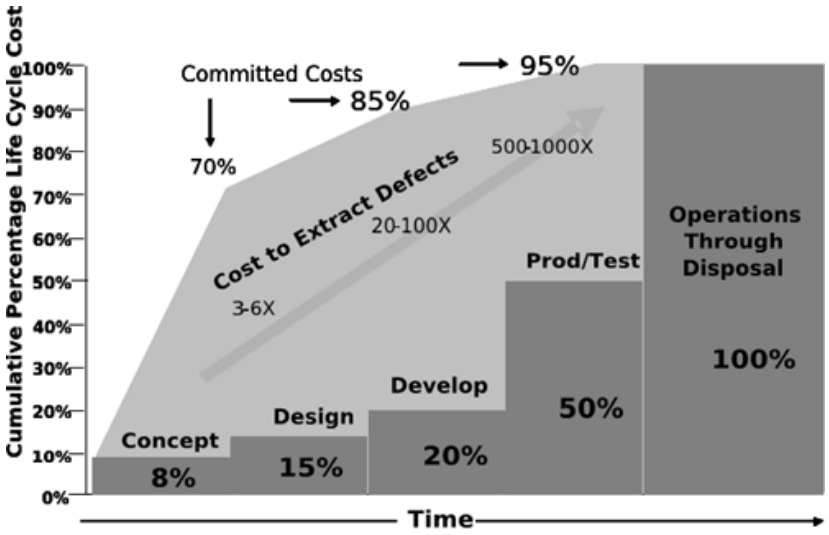


FIGURE 1.2 – Coûts engagés en fonction du temps [INCOSE, 2015b]

L'évaluation de l'efficacité de l'ingénierie système menée par Elm [2008] a aussi montré qu'il y a une forte corrélation entre les exigences et les performances d'un projet. Par ailleurs, Selby and Selby [2007] ont estimé que 49.1% des défauts en génie logiciel sont injectés pendant la phase d'ingénierie des exigences. Enfin, le populaire rapport annuel Chaos Manifesto du The Standish Group [2005] avait déjà identifié que l'ingénierie des exigences est une cause majeure qui engendre le succès ou l'échec d'un projet informatique. Les répartitions illustrées sur la figure 1.3 ont été synthétisées par l'AFIS [2012a]. Dans ces répartitions on distingue les facteurs de succès et les facteurs d'échecs. Le projet est un succès s'il est réalisé en temps et en heure, dans les budgets alloués, avec toutes les fonctionnalités et caractéristiques spécifiées. Inversement, le projet est un échec s'il est annulé avant la fin prévue ou n'est jamais mis en œuvre.

Facteur d'échecs	Causes racines	Facteurs de succès
37%	**Besoins, Exigences**	**40%**
9%	Projet, Ressources	23%
8%	Gestion des données techniques	14%
11%	Technique, Technologie	9%

FIGURE 1.3 – Les facteurs d'échecs et de succès des projet [AFIS, 2012a].

.

1.2 Le besoin

Le besoin émane du retour d'expérience des activités de conseil et d'intégration des solutions technologiques PLM de notre partenaire industriel, Keonys. Premier intégrateur européen des outils de gestion du cycle de vie des produits de Dassault Systèmes, Keonys est implanté sur 10 sites en Europe : 7 en France, 1 en Belgique, 1 en Hollande et 1 en Allemagne. Au cœur de l'entreprise numérique, et depuis plus de vingt ans, Keonys se positionne comme un partenaire qui accompagne ses clients dans la définition et la mise en œuvre d'une stratégie PLM outillée. Le métier de Keonys s'articule autour de trois axes majeurs :

- le conseil et l'intégration de solutions PLM,
- la distribution des logiciels de Dassault Systèmes (et autres partenaires), et de matériels informatiques spécifiques aux environnements de Conception Assistée par Ordinateur (CAO) et de PLM,
- la conception d'applications spécifiques et d'outils de productivité.

Issu du monde du logiciel, le groupe Keonys a créé un département de R&D constitué de dix ingénieurs qui développent de nouveaux produits et services. Les missions de R&D s'articulent autour de deux axes majeurs :

- les projets de développement à forte valeur ajoutée technologique en partenariat avec les principaux leaders industriels. Par exemple, l'application DocDoku avec l'avionneur Airbus, ou de nouvelles passerelles entre plusieurs outils PLM avec Thales Alenia Space.
- le développement de technologies en tant qu'éditeur. Par exemple, l'application Web LISA (Licensing Statistics Analyser) pour superviser le parc de licences logicielles vendues à des donneurs d'ordres de premier plan comme Alstom Transport ou Daher.

Attentifs aux nouvelles technologies open source dédiées à la fouille de données, le *cloud*, la gestion des connaissances, et aux interactions homme-machine, lesquelles sont devenues stratégiques pour la compétitivité des entreprises, Keonys réfléchit à de nouvelles applications numériques pour valoriser le patrimoine informationnel des entreprises. Les travaux de thèse présentés ici se positionnent parmi ces projets de R&D.

Dans leur activité quotidienne de définition et de mise en œuvre d'une stratégie PLM dans des secteurs d'activité divers tels que l'automobile, l'aéronautique, le naval, le bâtiment, l'énergie, la santé ou l'agroalimentaire, Keonys a constaté que de nombreuses entreprises souffrent d'un manque de solutions technologiques dédiées à la première phase du cycle de vie d'un produit manufacturé : la spécification.

Pour mieux comprendre la nature du problème, cette section restitue une analyse du besoin résumée dans le graphe des prestations 1.4 généralement appelé « bête à corne ».

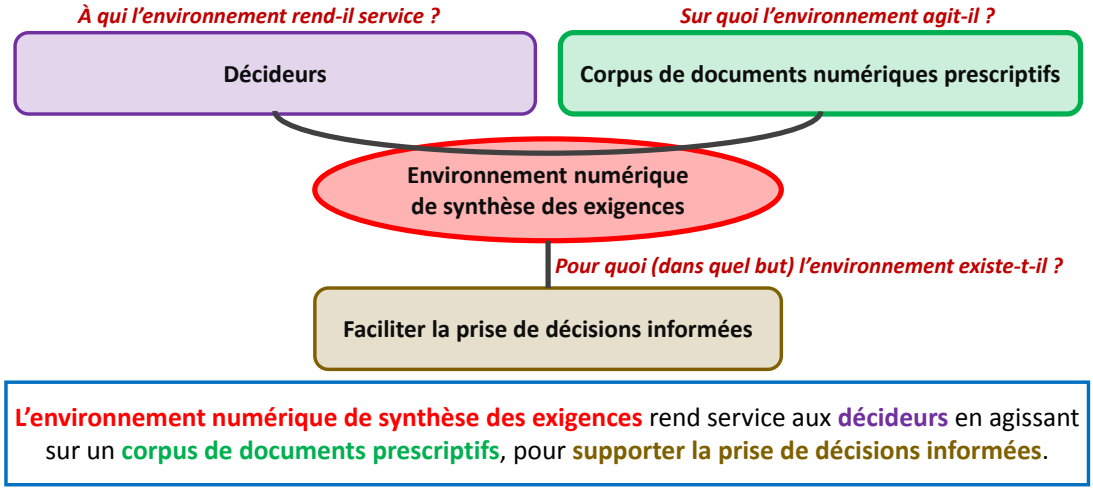


FIGURE 1.4 – Besoin d'un environnement numérique de synthèse des exigences.

À qui le besoin rend-il service ?

Si l'on s'affranchit de toute contrainte, l'environnement numérique est utile à quiconque souhaite obtenir une synthèse d'un corpus de documents numériques prescriptifs. N'importe quelle relation client-fournisseur, intra- ou inter-entreprises, pourrait donc bénéficier d'un tel outil. Néanmoins, ici, l'analyse du besoin se limite aux entreprises qui œuvrent dans l'ingénierie de produits.

À l'échelle de l'entreprise étendue, c'est-à-dire au sein de l'organisation systémique qui intègre un donneur d'ordres et une chaîne logistique, l'outil de synthèse des exigences est mis en œuvre dans chacune des relations contractuelles client-fournisseur. Plus précisément, l'environnement numérique est destiné à être utilisé par chacun des fournisseurs, en particulier par chacun des sous-traitants qui se positionnent au milieu, voire en bas de la chaîne logistique.

Au sein même d'une entreprise sous-traitante, l'environnement numérique rend service à un **décideur**. Un décideur est un individu qui a une fonction décisionnelle, c'est à dire qui a la capacité de prendre des décisions au sein de l'entreprise. Dans cette étude, à chaque fois que nous employons le terme décision, nous faisons référence à la définition donnée par Hazelrigg [2012], dont les caractéristiques sont les suivantes :

- Une décision n'est pas une action. Une décision ne possède aucune propriété matérielle : masse, inertie, vitesse, etc. Elle n'existe que dans la tête d'un décideur et demeure inconnue à quiconque jusqu'à ce que le décideur entreprenne une action basée sur la décision. Une décision est donc une chose mentale. Ce n'est pas l'action elle-même, mais l'engagement envers une action. **La prise d'une décision implique l'engagement de ressources** (argent, temps, énergie, etc.).
- **Une décision est un engagement pris au présent**. Il est impossible de s'engager dans le passé. Le passé est révolu. Par ailleurs, il est impossible de s'engager dans le futur. Le futur n'est pas encore là. Ainsi, chaque décision implique une allocation irrévocable de ressources. L'allocation de ressources est accomplie par l'action qui résulte de la décision.
- **Une décision est une chose qui est prise par un individu**. Une décision est, par essence, une pensée et une pensée n'existe que dans la tête d'un seul individu. Seul un individu peut s'engager envers une action. Certains argumenteront qu'un groupe a la capacité de décider. Or, l'action doit toujours suivre la décision sans aucune chance de dissension. Ainsi, un groupe a des comportements émergents, il ne prend pas de décisions.
- **Une décision est un choix parmi un ensemble d'alternatives**. Pour qu'une décision puisse exister il faut que le décideur ait le choix entre au moins deux alternatives. La décision est alors la pensée qui permet à un décideur de passer du choix à l'action. C'est lors de ce passage que se fait l'allocation irrévocable de ressources.
- **Une décision est prise au présent dans l'objectif d'un meilleur futur tel que le décideur le désire**. Quand un décideur fait un choix parmi un ensemble d'alternatives, son choix n'est pas basé sur ce qu'il pressent être une bonne décision, c'est-à-dire sur ce qu'il semble être bon de choisir, mais se fonde plutôt sur la conséquence de son choix. La décision est prise au présent, tandis que la conséquence arrive dans le futur.
- **Une décision est risquée**. Parce qu'une décision est prise au présent, alors qu'une conséquence arrive dans le futur (au moment de la décision), et parce que personne ne peut prédire le futur avec certitude, toutes les décisions impliquent une part de risque. Même dans les cas extrêmes de quasi-certitude ou quasi-incertitude, le futur n'est pas totalement prédicable.
- **Une décision requiert l'expression de préférences**. Sans préférence, un décideur n'a aucun moyen qui puisse affecter sa décision. Il est donc incapable de faire un choix parmi un ensemble d'alternatives.

Sur quoi le besoin agit-il ?

L'environnement numérique agit sur un corpus de **documents prescriptifs** rédigés en anglais. Un document prescriptif (Fig. 1.5) est un couple $(\mathcal{P}, \mathcal{I})$ dans lequel $\mathcal{P}$ est un ensemble d'énoncés prescriptifs et $\mathcal{I}$ est un ensemble d'énoncés informatifs. Un énoncé prescriptif est un énoncé écrit décrivant une

propriété, comportementale ou structurelle, que le produit doit posséder. Dans notre cas, un énoncé informatif est un énoncé écrit qui permet à un lecteur d'interpréter les énoncés prescriptifs.

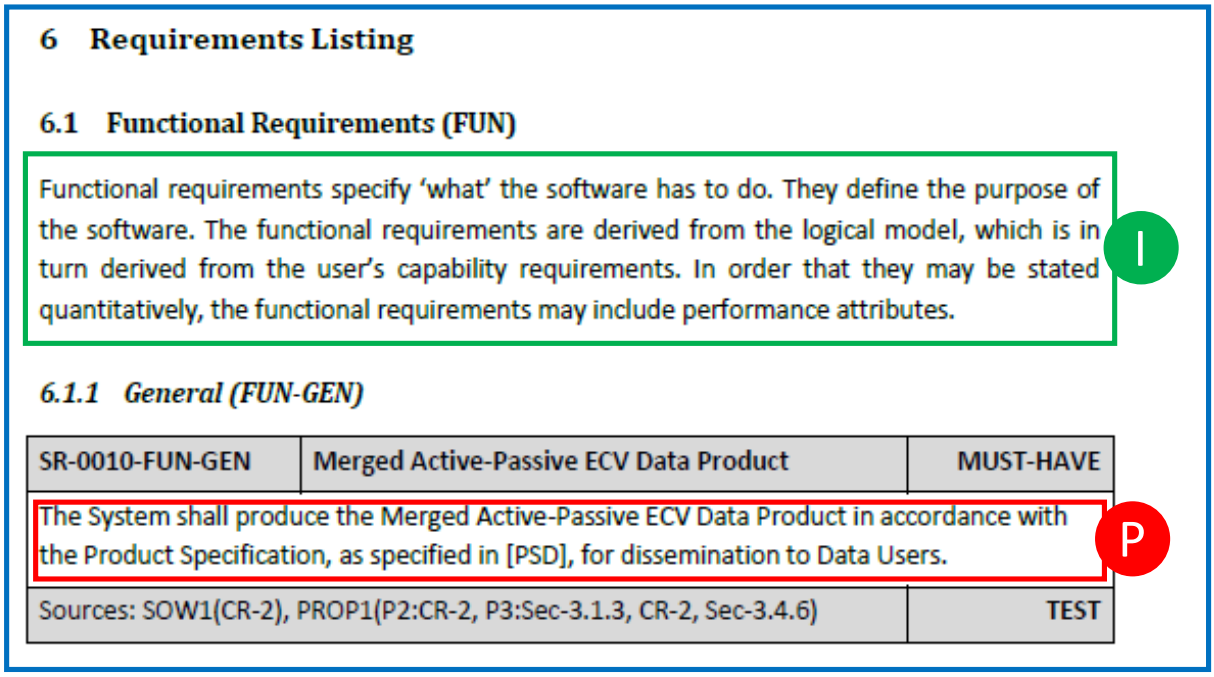


FIGURE 1.5 – Extrait d'un document prescriptif contenant des énoncés informatifs encadrés en vert et des énoncés prescriptifs encadrés en rouge.

Les énoncés prescriptifs sont habituellement de deux types : « besoin » ou « exigence ». Les deux guides [INCOSE, 2015a,b] de l'*International Council on Systems Engineering* (INCOSE) utilisent le mot « formel » à diverses reprises pour qualifier la transformation d'un besoin en exigence. Si la transformation du besoin en exigence était réellement formelle, alors nous devrions, par bijection, être aussi capable de passer de l'exigence au besoin. Or, aucune des deux transformations n'est explicitement détaillée, ce qui nous emmène à conclure que la frontière qui sépare un besoin d'une exigence n'est pas explicitement dessinée. En règle générale, un besoin est un énoncé prescriptif qui ne possède pas toutes les caractéristiques de qualité d'une exigence. Le besoin est la prescription selon la perspective du client – de la maîtrise d'ouvrage –, tandis que l'exigence correspond à la perspective du fournisseur – de la maîtrise d'œuvre. Au cours du passage client-fournisseur il y a, dans un premier temps, un processus de « dérivation » des exigences du client vers le fournisseur et, dans un second temps, un processus de « négociation » des exigences du fournisseur vers le client. La dérivation permet de traduire le besoin en exigence afin que cette dernière respecte des critères de qualité (complète, non ambiguë, faisable, etc.), tandis que la négociation permet d'élargir l'espace prescriptif en cas d'infaisabilité technique ou financière. Dans ces travaux de recherche, nous employons le terme « exigence » pour désigner les deux concepts : besoin et exigence. D'ailleurs, dans la littérature anglaise [ISO/IEC/IEEE 29148, 2011, Ryan, 2013], il est fréquent que le terme « *stakeholder requirement* » soit préféré au terme « *need* ». Le cas échéant, on ne parle pas de « *requirement* », mais de « *systems requirement* » ou « *technical requirements* ».

Les documents prescriptifs définissent et contractualisent les obligations qui régissent la coopération de multiples parties prenantes. Dans un processus contractuel entre un client et un fournisseur tel que celui illustré par la figure 1.6, les documents prescriptifs sont principalement remis par le client au fournisseur. Ils peuvent aussi provenir de sources internes au fournisseur. Par exemple, des différentes fonctions de l'entreprise (bureau d'études et des méthodes, service de qualité, etc.) ou des unités en charge d'un sous-système (cockpit, train d'atterrissage, ailes, moteurs, etc.). Ils peuvent également être remis par des tierces parties telles que des organismes de régulation ou des législations. L'Agence Européenne de la Sécurité Aérienne (AESA), par exemple, qui régit la sécurité aérienne impose des

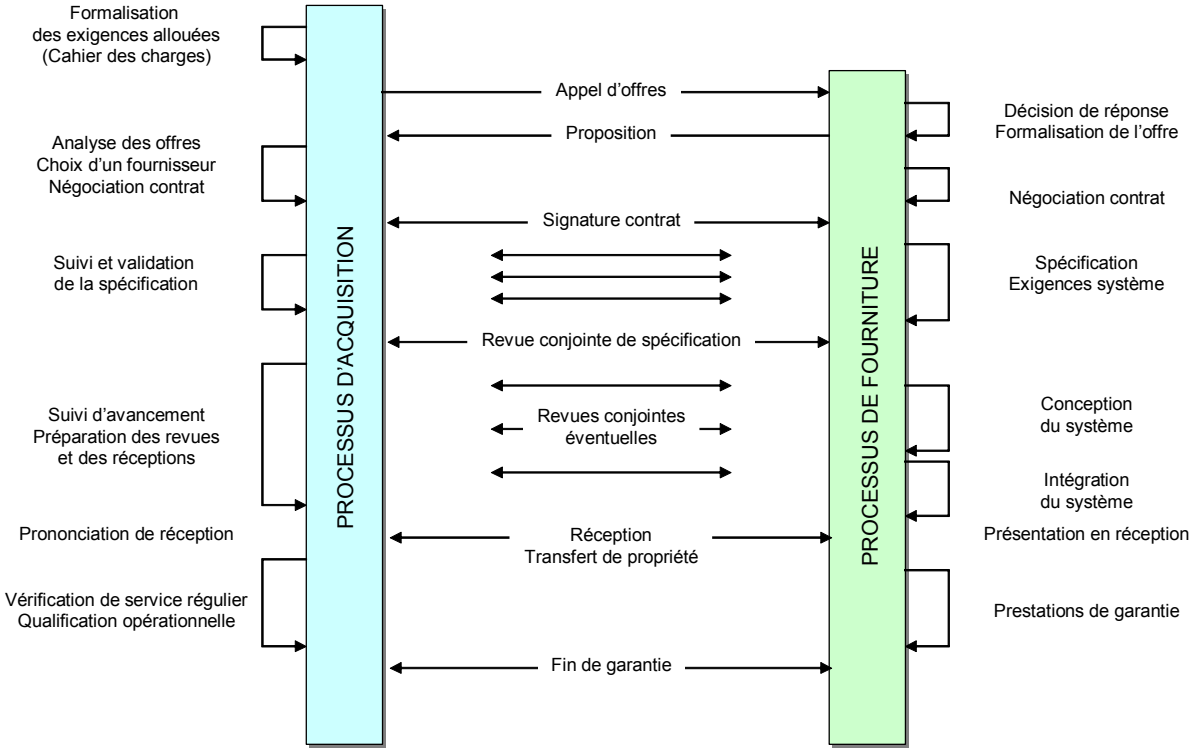


FIGURE 1.6 – Exemple de processus contractuel en ingénierie système [AFIS, 2012b].

quantités pharamineuses d'exigences au sein desquelles chaque constructeur aéronautique doit faire le tri afin d'identifier celles qui sont applicables.

Quand on fait référence aux documents prescriptifs, en premier lieu, on pense au cahier des charges et à la spécification. Le cahier des charges collecte les attentes du client selon le langage de la maîtrise d'ouvrage. La spécification non seulement exprime ce que le système doit faire selon le langage de la maîtrise d'œuvre, mais aussi complète les exigences initiales du client avec les exigences techniques du fournisseur. Par exemple, les contraintes de sécurité, de fabrication et de maintenance sont greffées aux exigences initiales. Cette structure duale des documents prescriptifs est une règle de bonnes pratiques, mais elle n'est en aucun cas universelle. En effet, en pratique, les énoncés prescriptifs sont dispersés dans divers documents [Sannier and Baudry, 2012]. Des rapports et des questionnaires d'entretiens, de brainstorming, de jeux de rôles, de scénarios, d'ateliers de créativité, de veille technologique, de retour d'expérience, etc. sont utilisés pour collecter des besoins. Dans une démarche d'ingénierie système, le concept d'opérations (ConOps) [IEEE 1362, 1998] est une description opérationnelle de ce que l'utilisateur doit être capable de faire avec le système, sa mission, sa raison d'être, ses scénarios opérationnels, etc. C'est donc le point d'entrée pour la définition des exigences du système. En dernier lieu, les textes législatifs et réglementaires sont des documents qui contiennent plusieurs centaines, voire plusieurs milliers d'énoncés prescriptifs qui régissent le cycle de vie d'un produit.

Précédemment, nous avons expliqué que l'environnement numérique de synthèse des exigences doit servir à des individus qui ont une fonction décisionnelle au sein d'une entreprise sous-traitante. Par ailleurs, nous avons vu que l'environnement numérique agit sur un corpus de documents numériques prescriptifs contenant plusieurs centaines ou milliers d'exigences. Comme le décrit l'image 1.7, nous allons maintenant voir dans quel but, puis pourquoi le besoin pour un tel outil se fait sentir.

Pour quoi (dans quel but) le besoin existe-t-il ?

La compétitivité économique qui anime les industriels leur impose des contraintes vis-à-vis du triptyque : coûts, délais et qualité. Or, le constat général est que la prolifération des exigences fait que ces

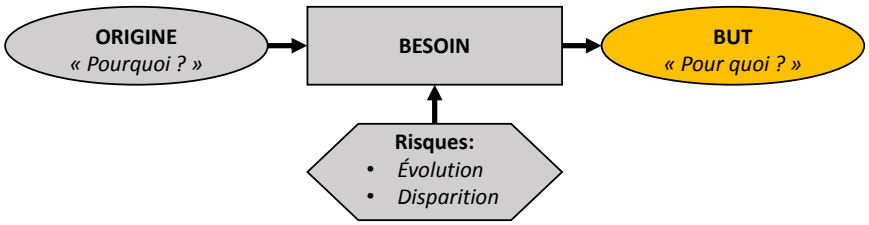


FIGURE 1.7 – Identification du but.

mêmes industriels ne sont plus capables d'appréhender un tel volume de données de manière intelligible, ce qui, *in fine*, met en péril la conception, l'utilisation, la maintenance et le recyclage du produit. Par conséquent, les entreprises ont tendance à adopter la stratégie du Standish Group [Group, 2013], lequel recommande de n'implémenter qu'un sous-ensemble d'exigences clés, car seulement 20 % des fonctions d'un produit supportent 80 % de sa valeur.

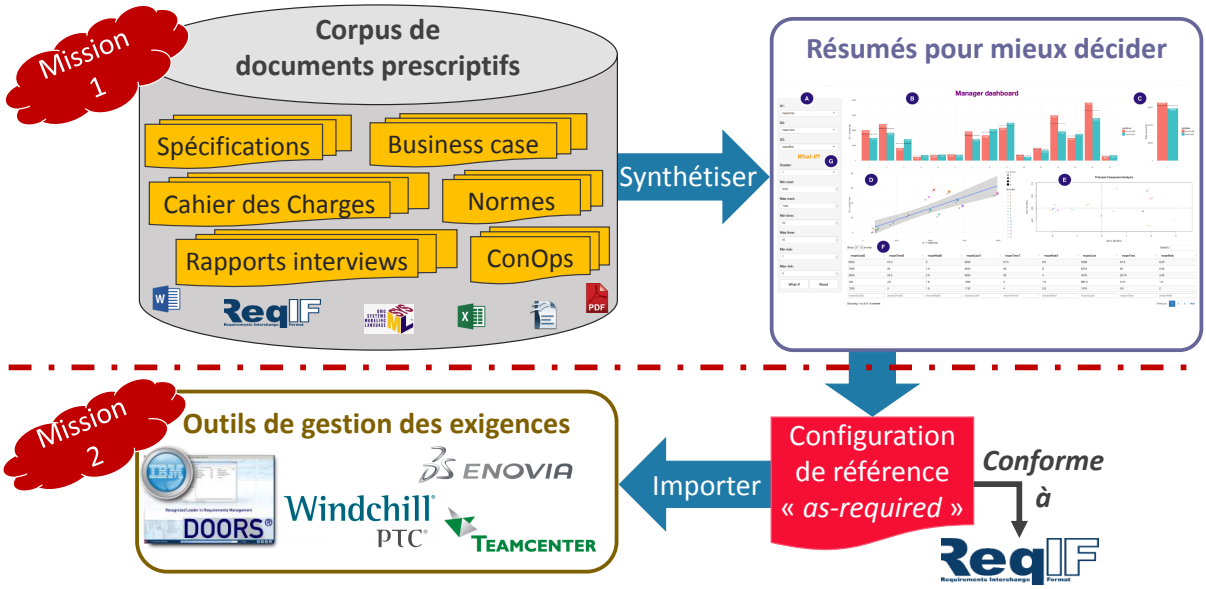


FIGURE 1.8 – Missions de l'environnement numérique.

La mission de l'environnement numérique proposé est alors de supporter une activité de **synthèse des exigences** pour faciliter la prise de décisions informées en phase amont du cycle de vie du produit. La décision informée la plus évidente que cette proposition vise à supporter est la réponse à un appel d'offre qui doit être conclue par une décision de « *GO vs. NO GO* ».

Comme le montre la figure 1.8, au-delà de la mission 1 « à court terme » qui est présentée dans ce manuscrit, « à moyen terme », l'activité de synthèse des exigences outillée peut permettre de trier et prioriser les exigences [Davis, 2003]. Un sous-ensemble d'exigences clés constituerait alors une configuration de référence « *as-required* » optimisée prête à être gérée au moyen d'un outil de gestion des exigences tel que IBM Rational Doors ou Enovia V6 Requirements Manager. La configuration de référence est qualifiée d'optimisée car elle est amenée à ne contenir qu'un sous-ensemble d'exigences clés judicieusement choisies au juste besoin selon une stratégie. En effet, si l'entreprise décide d'adopter une stratégie d'innovation engendrant des investissements et des risques élevés, alors elle sélectionnera quelques exigences dont l'implémentation peut nécessiter le développement de nouvelles compétences qui, en contrepartie, élargiront les parts de marché. À l'inverse, si l'entreprise préfère jouer la carte de la sécurité, alors elle s'engagera sur des exigences pour lesquelles elle possède déjà une solution pré-

existante ou similaire. Parmi les exigences clés on pourrait ainsi retrouver les exigences : (1) **légales** qui sont les exigences non négociables nécessaires à la certification du produit ; (2) **basiques** qui sont les exigences non négociables sans lesquelles personne n'achète le produit ; (3) **à valeur ajoutée** qui sont les exigences négociables prescrivant des facteurs de compétitivités.

Pourquoi (à cause de quoi) le besoin existe-t-il ?

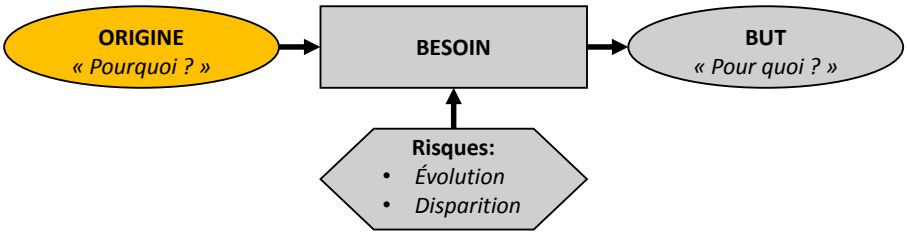


FIGURE 1.9 – Identification de l'origine du besoin.

Après s'être intéressé au but que l'environnement numérique doit permettre d'atteindre, nous discutons l'origine d'un tel besoin (Fig. 1.9). Pourquoi les entreprises ont-elles besoin d'un environnement numérique de synthèse des exigences ?

Abrev.	*Level*	*Order of magnitude*	*Sample empirical evidence*	*Interdependency management conjectures with current RE technology*
SSRE	Small-Scale Requirements Engineering	~10 requirements		Managing a complete set of interdependencies requires small effort.
MSRE	Medium-Scale Requirements Engineering	~ 100 requirements	[3]	Managing a complete set of interdependencies is feasible but requires large effort.
LSRE	Large-Scale Requirements Engineering	~1000 requirements	[8]	Managing a complete set of interdependencies is practically unfeasible, but feasible among small bundles of requirements.
VLSRE	Very Large-Scale Requirements Engineering	~10000 requirements	[6]	Managing a complete set of interdependencies among small bundles of requirements is unfeasible in practice.

FIGURE 1.10 – Les différentes volumétries d'exigences [Regnell et al., 2008].

Un environnement numérique de synthèse est nécessaire parce que le nombre d'exigences tend à croître de manière exponentielle, et ce, au point que l'on puisse aujourd'hui les considérer comme des « *big data* » à partir desquelles il est difficile, voire impossible, de prendre des décisions stratégiques avisées. Par exemple, la spécification du projet *FBI Virtual Case file*, une base de données informatique dont le budget avoisinait les 170 millions de dollars contenait 800 pages d'exigences [Goldstein, 2005]. Comme l'illustre la figure 1.11, chez Mercedes-Benz, une automobile est, en moyenne, composée de 100 sous-systèmes et 400 composants élémentaires [Houdek, 2013, 2012]. La spécification d'un système se matérialise par un document dont la taille avoisine les 200 pages, tandis que celle d'un composant élémentaire requiert 250 pages. Ainsi, la spécification d'un véhicule est équivalente à un corpus de plus de 100 000 pages. Plus généralement, la taille d'une spécification chez Mercedes-Benz varie entre 60 et 2000 pages, 600 pages en moyenne, hébergeant entre 1000 et 50 000 exigences. Par ailleurs, les exigences requièrent une vue contextuelle qui met en évidence les 30 à 300 documents applicables – d'autres spécifications ou des normes, par exemple – référencés par les exigences qui, eux aussi, sont susceptibles de prescrire des exigences applicables. Chez Ericsson Microwave Systems (EMW)

en Suède, un projet requiert entre 3000 et 6000 exigences selon le type de produit avec 700 à 1300 exigences par sous-systèmes [Alenljung and Persson, 2006]. Dans un autre article, Langenfeld et al. [2016] rapportent que chez l'équipementier Bosch, le développement d'un convertisseur DC-DC dédié à un véhicule automobile a nécessité une spécification de 10 000 exigences. Enfin, Regnell et al. [2008] mentionnent que chez le fabriquant de téléphone Sony Ericsson, le nombre d'exigence excède 10 000. Pour résumer, les différents ordres de grandeur sont décrits dans la figure 1.10.

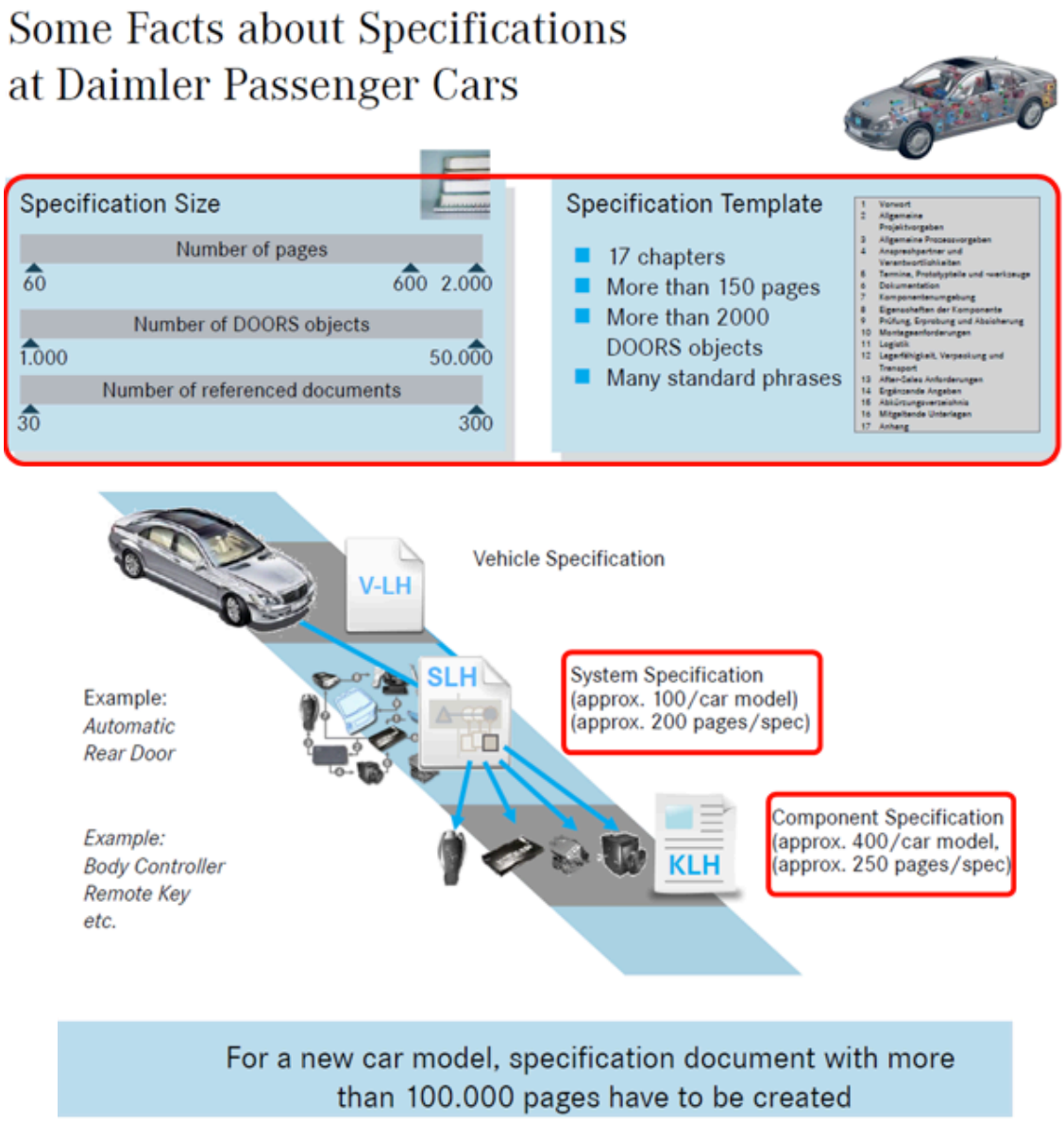


FIGURE 1.11 – Statistiques en ingénierie des exigences chez Mercedes-Benz [Houdek, 2013].

Parmi les raisons qui sont à l'origine de la prolifération des énoncés prescriptifs on peut citer :

• **R1 - La complication et la complexification technique des systèmes.** Les adjectifs compliqués et complexes sont la source de débats bouillonnants en épistémologie et en systémique. Diverses interprétations, souvent influencées par les problématiques inhérentes à chacune des disciplines [Jackson, 2015], ont été proposées. En informatique, la complexité correspond au temps demandé pour l'exécution d'un programme. Dans la théorie de l'information, la complexité est étroitement liée à l'entropie de l'information introduite par Shannon [1948]. En conception, la complexité est une mesure de l'incertitude quant à la faculté à satisfaire une exigence fonctionnelle [Suh, 2005]. En ingénierie système, domaine qui nous intéresse, on distingue la complexité structurelle et la complexité dynamique [AFIS,

2012b]. La complexité structurelle dépend du nombre de constituants, ainsi que de la variété et de l'homogénéité des constituants. La complexité dynamique, elle, dépend du nombre d'interactions entre les constituants, ainsi que de la variété et de l'homogénéité des interactions. La complexité systémique se concrétise par la présence de propriétés émergentes faisant que les propriétés globales du système sont non réductibles aux propriétés de ses constituants, mais émergeant, par synergie, du réseau d'interactions entre ses constituants. Tous ces points de vues sont subjectifs car ce qui est complexe pour l'un ne l'est pas forcément pour l'autre. L'aspect progressiste des sciences et des technologies fait que l'interprétation de la complexité évolue au cours du temps. Ainsi, aux yeux d'Aristote, le système solaire semblait complexe, alors qu'il paraissait beaucoup plus simple aux yeux de Kepler. Néanmoins, si l'on considère les éléments de définition proposés en ingénierie système, quand on observe les systèmes que nous utilisons quotidiennement, on peut admettre que l'évolution monotone croissante de la complexité des systèmes nécessite une spécification toujours plus exhaustive afin de spécifier leurs propriétés structurelles et comportementales. Pour s'en convaincre, il suffit de constater l'évolution technique (nombre de pièces, physiques multiples, couplage du logiciel et du matériel, etc.) des systèmes de transport (automobile, aérien, ferroviaire, etc.).

• **R2 - Le foisonnement des textes législatifs et réglementaires.** Quand bien même nous vivons dans une société dans laquelle les sciences et les technologies ont permis d'atteindre un confort et une sécurité qui n'a pas son pareil dans l'histoire, il semblerait que nous soyons entrés dans ce que Beck [2008] appelle : « la société du risque ». Une société où tout est pensé sous l'angle de la menace : OGM, énergie nucléaire, moteur Diesel, pesticides, clonage, intelligence artificielle, etc [Klein, 2013, 2006, Ferry, 2015]. Cette nouvelle passion par laquelle est prise notre société est partiellement due à la banalisation des techno-sciences qui a fait que nous sommes passés d'une société de la connaissance à une société de l'usage des technologies dans laquelle l'homme, dépourvu de curiosité et de sens critique, est devenu perméable à toutes sortes de croyances [Klein, 2013, 2006, de Kervasdoué, 2014]. Pour calmer ces peurs, les législations et les autorités de régulation légifèrent à tour de bras. Les industries qui développent des produits dont la sécurité est à risque – l'aviation et le nucléaire, par exemple – croulent elles aussi sous les textes réglementaires. Les exigences prescrites par ces textes législatifs et réglementaires, lesquels relèvent parfois plus d'une analyse superficielle qui tient beaucoup de l'opinion et peu de la raison, étouffent le tissu industriel. Ces textes sont par ailleurs construits comme un chaînage de références transversales faisant appel les unes aux autres, et dont la prise de connaissance frivole met en péril les performances et la qualité du produit.

• **R3 - La diversité des produits.** Pour être compétitive ou, plus radicalement, pour ne pas mourir, une entreprise doit, quoi qu'il advienne, innover pour innover, c'est-à-dire diversifier ses produits afin qu'ils deviennent peu à peu obsolètes. Tous les ans, une entreprise commercialise une « nouvelle génération » de ses produits. Ce type de moteur de croissance, fervent opposant au modèle économique keynésien, est ce que Schumpeter [1942] a théorisé sous le nom de « destruction créatrice », plus communément connue sous le nom d'innovation, ou pour reprendre le titre du livre de Luc Ferry [2015] : « l'innovation destructrice ». Pour innover, beaucoup d'entreprises diversifient leur produit phare. Par ailleurs, l'internationalisation des marchés et la personnalisation de masse nécessitent de créer des produits personnalisés dont les variantes et les options sont stratégiquement spécifiées pour chacun des marchés visés. Par exemple, selon David [2000], aux États-Unis, entre les années 1970 et 1990, le nombre de modèles automobiles est passé de 140 à 260 et le nombre de modèles de véhicules à utilité sportive (SUV) a été multiplié par 5 en passant de 8 à 38. Cette explosion de la diversité des produits ne s'est pas limitée à l'industrie automobile. Le nombre de modèles de chaussures de sports était de 5 dans les années 1970 et a atteint 285 dans les années 1990, soit une multiplication par 57 ! Le nombre de variantes de produits céréaliers consommés au petit déjeuner est passé de 160 à 340, alors que celui des lentilles de contact a fait un incroyable bond de 1 à 36. Dans bien des cas, les instances déclinées satisfont aux mêmes besoins fonctionnels, mais jouent sur des critères de performance et de qualité. Bien que les entreprises s'emploient à définir des méthodes de définition de lignes de produits, il est inévitable que le nombre d'exigences augmente [Regnell et al., 2008].

• **R4 - L'ingénierie concourante dans une entreprise étendue.** De nos jours, le développement

FIGURE 1.12 – Hiérarchie de relations contractuelles client-fournisseur.

d'un système se fait au sein d'une entreprise étendue. Une entreprise étendue peut être définie comme un ensemble de partenariats entre des clients et des fournisseurs. La structure organisationnelle peut être représentée sous la forme d'un réseau, ou, et c'est plus fréquemment le cas, sous la forme d'une hiérarchie (Fig. 1.12) au sein de laquelle, de manière récursive, un donneur d'ordres domine des fournisseurs. Chaque fournisseur joue alors le rôle d'un sous-traitant disposant d'une expertise. Une telle stratégie a de nombreux avantages. Elle permet de partager les risques financiers, mais aussi de mutualiser les connaissances et les ressources matérielles, logicielles ou humaines. Par ailleurs, au sein d'une entreprise, une démarche d'ingénierie concourante [Solhenius, 1992] est adoptée pour diminuer les délais de mise sur le marché et les itérations de *rework*. Cela se fait grâce à la conjonction, d'une part, de l'ingénierie simultanée qui fait que les expertises évoluent en parallèle et, d'autre part, de l'ingénierie intégrée qui incite les disciplines à collaborer au plus tôt afin de considérer les contraintes inhérentes à chacun des métiers. Toutefois, travailler dans un tel contexte collaboratif et pluridisciplinaire nécessite de spécifier les interfaces techniques et organisationnelles entre les parties prenantes. Par exemple, dans une industrie dopée au numérique, il faut spécifier les échanges d'informations d'une organisation à l'autre, les droits d'accès alloués à chacun des utilisateurs, les processus de modification des objets techniques, etc. Ainsi, les entreprises définissent des règles dans des polices qui doivent être respectées à l'intérieur et aux interfaces de l'entreprise.

• **R5 - Le langage naturel.** L'usage intensif du langage naturel comme moyen de spécification est une énième cause qui engendre la pullulation d'exigences. En effet, quand une entreprise décide d'améliorer ou de diversifier un produit existant, ou quand elle souhaite réutiliser des composants sur étagère, il n'est pas rare que les spécifications soient partiellement ou totalement réutilisées sans aucune, ou avec très peu, de précaution d'un projet à l'autre. Des exigences inutiles à la spécification du système d'intérêt se greffent alors aux besoins initiaux du client et mettent en péril le projet.

Pour toutes les raisons énoncées précédemment, habituellement, lorsqu'un fournisseur se voit attribuer un lot de documents prescriptifs, il préfère mettre en œuvre des moyens pragmatiques qui limitent les risques et poussent à modifier des solutions de conception pré-existantes, plutôt que d'adopter une démarche causale d'ingénierie : spécification et conception. Certes, la démarche scientifique qui consiste à bien définir le problème avant de rechercher des solutions nécessite un investissement pour satisfaire

de nouvelles exigences, mais les conclusions qui demeuraient jusque-là inconnues sont capitalisées pour les projets à venir et, dans bien des cas, assurent de nouvelles parts de marché.

Le besoin identifié peut évoluer ou disparaître au cours du temps selon les innovations concurrentes, il faut donc valider sa pérennité en s'interrogeant sur les risques d'évolution ou de disparition.

Qu'est-ce qui pourrait faire disparaître ou évoluer le besoin ?

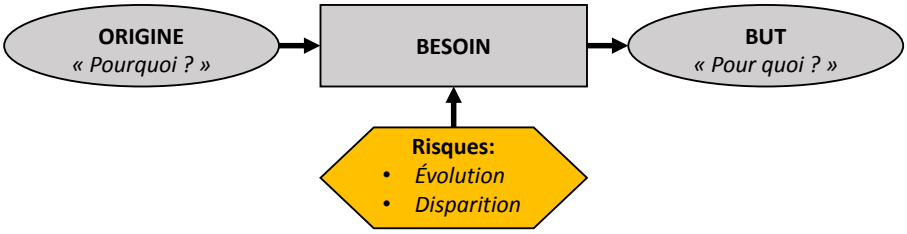


FIGURE 1.13 – Identification des raisons qui pourraient faire évoluer ou disparaître le besoin d'un environnement numérique de synthèse des exigences.

Le besoin est sujet à des risques d'évolution, voire de disparition (Fig. 1.13). Pour notre partenaire industriel Keonys, il est indispensable de les anticiper le plus tôt possible. Pour cela, nous cherchons des réponses à la question : qu'est-ce qui pourrait faire disparaître ou évoluer le besoin ?

1. Si le nombre d'énoncés prescriptifs diminue. [C'est plutôt l'inverse]

Si la complication et la complexité des produits diminuent significativement, si les produits personnalisés s'uniformisent, si la législation décide de ne plus légiférer, et si les produits en venaient à être développés comme ils l'étaient par les artisans d'antan, alors les produits ne nécessiteraient plus de spécifications exhaustives. L'analyse des raisons qui font que les exigences prolifèrent a montré que c'est plutôt la tendance inverse que l'on observe. Comme le dit le dicton : « On n'arrête pas le progrès », notamment le progrès technique.

2. Si les énoncés prescriptifs dans les documents disparaissent. [Peu probable]

Quel que soit leur format, les documents pourraient être substitués par des technologies qui ne permettraient pas, ou ne nécessiteraient pas, d'exporter les données sous forme de documents textuels. Toute la chaîne de valeur pourrait, par exemple, collaborer autour d'une, ou synchroniser plusieurs, bases de données telles que IBM Rational Doors ou ENOVIA Requirements V6. Une seconde « menace » potentielle correspond aux approches centrées sur les modèles « *Model-Based Systems Engineering - MBSE* » [Wymore, 1993], lesquelles sont inscrites dans la vision 2020 [1] de l'INCOSE [2007]. Quelques adeptes de l'ingénierie système argumenteront que l'utilisation de modèles – FFBD, IDEF0, FAST, SADT, BPMN, SysML, etc. – n'est pas une révolution méthodologique, alors que d'autres voient venir le salut de l'ingénierie système. La nouveauté ne correspond à aucune de ces extrêmes. Ce qui est nouveau, c'est la volonté d'éradiquer, autant que faire se peut, les documents. Enfin, il y a les approches comme la théorie de conception pilotée par la valeur – *Value-Driven Design* (VDD), en anglais – développée par le comité du programme VDD de l'*American Institute of Aeronautics and Astronautics* (AIAA). La VDD propose de modéliser les exigences comme un problème d'optimisation sous

1. « *Model-based systems engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases. MBSE is part of a long-term trend toward model-centric approaches adopted by other engineering disciplines, including mechanical, electrical and software. In particular, MBSE is expected to replace the document-centric approach that has been practiced by systems engineers in the past and to influence the future practice of systems engineering by being fully integrated into the definition of systems engineering processes* » [INCOSE, 2007].

contraintes. Il ne s'agit alors plus de proposer l'une des solutions qui appartiennent à l'espace prescriptif contraint par les exigences, mais de chercher la solution optimale qui maximise ou minimise une fonction objective de valeur. L'ingénierie système basée sur les modèles est une initiative louable. Cependant, bien que ces technologies puissent, et c'est la volonté de beaucoup d'adeptes de l'ingénierie système, remplacer toute ou partie des documents, nous demeurons convaincus que les documents seront toujours utiles à la collection des besoins et des exigences, notamment dans les petites et moyennes entreprises, et qu'ils cohabiteront dans un environnement technologique hétérogène fait de documents, de modèles (semi-)formels, de bases de données, etc. Il est également peu probable que les exigences prescrites par les textes législatifs et réglementaires, lesquels sont certainement le premier réservoir d'énoncés prescriptifs, prennent la forme de modèles (semi-)formels. Par ailleurs, parmi les systèmes sémiotiques, le langage naturel fut, est, et demeurera, le seul moyen de communication universel qui satisfait à la diversité des acteurs impliqués dans le cycle de vie d'un produit [INCOSE, 2015a]. D'ailleurs, hormis dans quelques travaux de recherche académique, même le *System Modeling Language* (SysML), spécialement créé pour supporter une démarche de MBSE, s'appuie sur des exigences textuelles telles qu'on peut les lire dans des documents. Les descriptions graphiques et logiques, elles, continueront donc de profiter à un cercle limité d'experts.

1.3 Positionnement scientifique

Pour mieux situer nos travaux de recherche, nous étudions la place qu'ils prennent au sein de l'ingénierie de produit d'une part, et au sein de la recherche académique en ingénierie des exigences d'autre part.

Positionnement au sein de l'ingénierie

Dans sa matrice de créativité (Fig. 1.14), Gold [2007] distingue l'art, la science, le design et l'ingénierie. Selon lui, la science cherche à découvrir et comprendre les lois de la nature et à les exprimer sous la forme d'équations mathématiques. Cette définition unanime de la science remonte au passage de la physique aristotélicienne à la physique moderne de Galilée [2]. Alors que l'art et le design sont souvent confondus, l'auteur insiste sur le fait que pour l'artiste les tests sont sans intérêts, tandis qu'ils sont fondamentaux pour le designer : « *If an artist looks inward as a way of seeing the world, the designer looks outward toward others. An artist paints a painting, stares at it, and says,"isn't it beautiful, it expresses my inner vision perfectly". The designer paints a painting, stares at, then turns it around to the audience and asks "Do you like it ? No ? Then I'll change it."* ». Le designer est donc centré sur la satisfaction du client. Le quatrième et dernier chapeau de la créativité est l'ingénierie. Selon l'auteur, l'ingénierie c'est la résolution de problèmes régis par les contraintes naturelles et législatives. L'ingénierie est la source de l'innovation.

D'après John Maeda, chacun des quadrants a sa propre mission : l'exploration pour la science ; l'innovation pour l'ingénierie ; la communication pour le design ; l'expression pour l'art. Par ailleurs, dans sa matrice (Fig. 1.15), Maeda distingue les disciplines universelles – la science et l'art – de celles qui sont spécifiques – le design et l'ingénierie. D'autre part, il oppose celles qui sont relatives à la matière – l'ingénierie et les sciences – de celles qui sont relatives à l'esprit – l'art et le design. Ces représentations matricielles de la créativité sont radicales et peuvent donc soulever quelques critiques, mais elles illustrent bien les grandes approches de création.

Le *Krebs Cycle of Creativity* (KCC) (Fig. 1.16) est une carte proposée par Oxman [2016] qui révise les relations entre la science, l'ingénierie, le design et l'art. Le rôle de la science est d'expliquer et

2. « La philosophie est écrite dans cet immense livre qui est constamment ouvert sous nos yeux, je veux dire l'univers, mais on ne peut le comprendre si l'on ne s'applique d'abord à en comprendre la langue et à connaître les caractères avec lesquels il est écrit. Il est écrit en langue mathématique et ses caractères sont des triangles, cercles et autres figures de géométrie, sans le moyen desquels il est humainement impossible d'en comprendre un mot. Sans eux, c'est une errance vaine dans un labyrinthe obscur. »

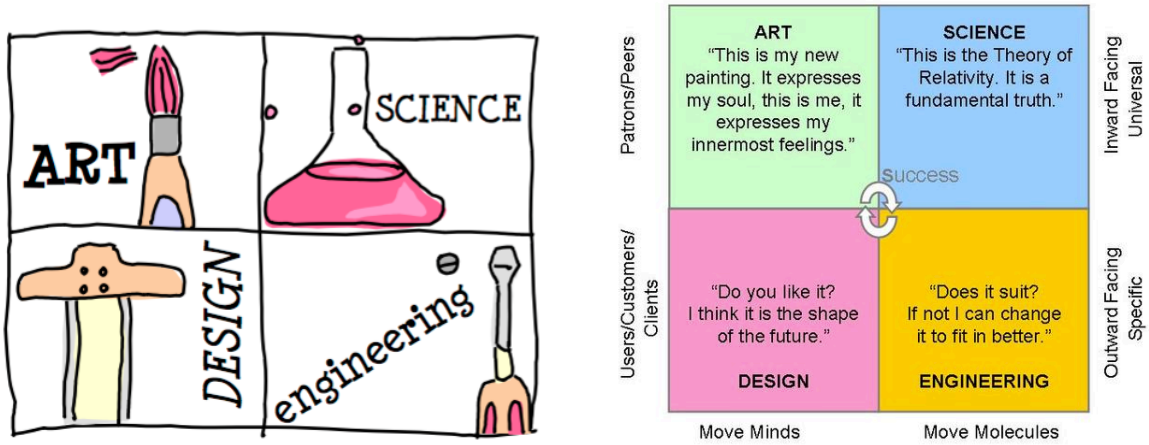


FIGURE 1.14 – Matrice de créativité selon Gold [2007].

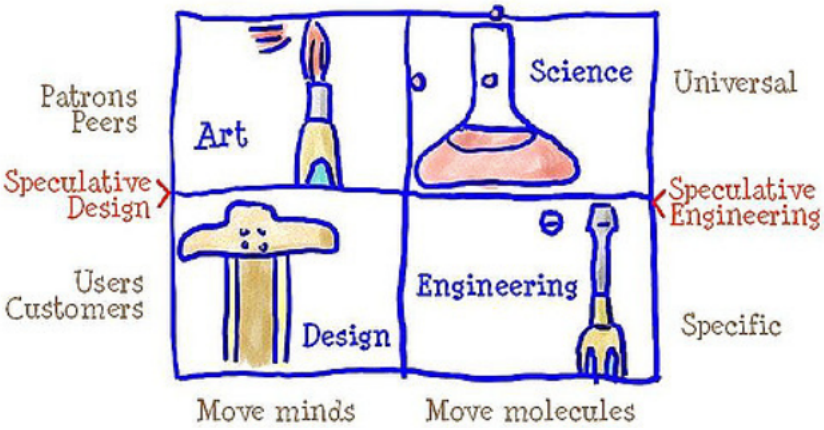


FIGURE 1.15 – Matrice de créativité selon Maeda.

de prédire le monde qui nous entoure ; elle convertie l'information en connaissance. L'ingénierie, elle, développe des solutions pour des problèmes empiriques ; elle convertie la connaissance en utilité [3]. Le troisième cadrant, le design, vise à produire des solutions qui maximisent la fonction et augmentent l'expérience humaine ; il convertit l'utilité en comportement. Enfin, l'art a pour rôle de questionner le comportement humain et d'établir une prise de conscience du monde qui nous entoure ; il convertit le comportement en de nouvelles perceptions de l'information. Les quatre modalités prennent place dans un cycle. La science produit de la connaissance qui est utilisée par les ingénieurs. L'ingénierie produit des changements comportementaux qui sont perçus par les artistes. L'art produit de nouvelles perceptions du monde qui alimentent le questionnement scientifique.

Nos travaux se positionnent clairement dans le domaine de l'ingénierie, mais laquelle ? Bachimont [2007] distingue l'ingénierie artisanale, industrielle, et complexe. L'ingénierie artisanale correspond à l'artisan qui, dans son atelier, invente, improvise en fonction des ressources naturelles – bois, acier, terre, etc. – dont il dispose pour satisfaire une commande unique à chaque client. L'ingénierie industrielle c'est l'usine. Un système de production dans lequel les opérations cadrées et séquencées transforment des ressources qui ne sont plus brutes mais façonnées pour la production intensive. Le passage de l'ingénierie artisanale à l'ingénierie industrielle c'est le remplacement de l'improvisation par la répétabilité et de la personnalisation par la standardisation. Enfin, la troisième ingénierie, l'ingénierie complexe, s'applique à des systèmes complexes qui nécessitent de s'adapter sans cesse au contexte. Par ailleurs, l'auteur

3. L'utilité telle qu'elle est définie en économie, c'est-à-dire : « une mesure du bien-être ou de la satisfaction obtenue par la consommation, ou du moins l'obtention d'un bien ou d'un service » (Wikipédia).

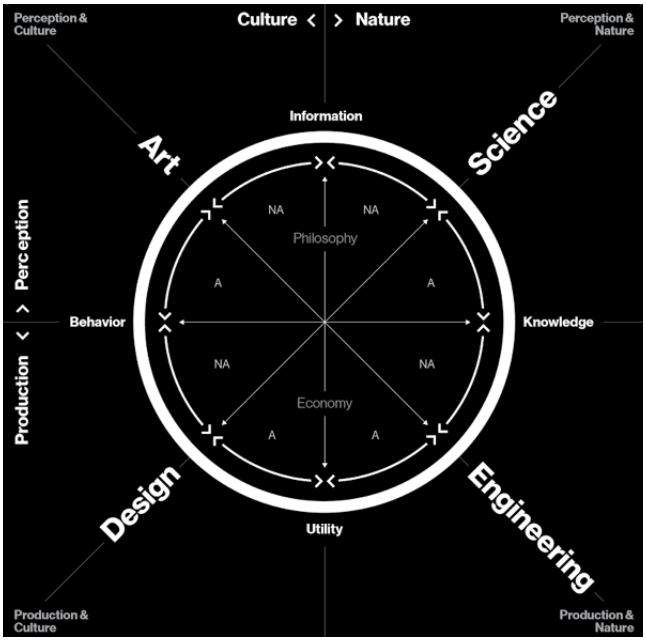


FIGURE 1.16 – Matrice de créativité selon Oxman [2016].

nous dit qu'elle n'est plus menée dans une usine fermée au monde extérieur, elle est désormais ouverte aux particularités et contingences du dehors. L'ingénierie complexe est finalement caractérisée par la nécessité permanente d'innover et d'inventer, car chaque solution inventée déplace les frontières du problème auquel elle répond. L'ingénierie est donc condamnée à créer en permanence pour réaliser les possibles qu'elle invente. Nos travaux de recherche s'appliquent à cette dernière ingénierie que l'on peut résumer grâce à trois principes : la complexité, l'entreprise étendue, et l'innovation destructrice.

de Weck et al. [2011] distingue aussi des époques clés qui ponctuent l'évolution de l'ingénierie (Fig. 1.17). La première époque de l'ingénierie n'apparaît pas dans les grandes évolutions car elle remonte à l'origine de l'homme qui, depuis toujours, pratique l'ingénierie, en particulier la construction d'abris et la fabrication d'armes de chasse, pour assouvir ses besoins de survie. Cette ingénierie a duré jusqu'à la fin du 18ème et début du 19ème siècle, période durant laquelle a eu lieu la révolution industrielle et de grandes inventions telles que l'automobile, le téléphone ou l'ampoule. Bien qu'elles soient honorables, ces grandes inventions appartiennent à l'ingénierie artisanale, laquelle se concentre sur la résolution d'un problème exclusivement technique. La standardisation de l'automobile par Henri Ford, laquelle est représentative de l'ingénierie industrielle, conjuguée à une croissance importante de la population est représentative d'un nouveau mouvement : l'ingénierie des systèmes complexes. Cette dernière se caractérise par la mise en réseau des grandes inventions. La mise en réseau des automobiles, du téléphone et des ampoules a en quelque sorte donné naissance à trois systèmes complexes : le réseau autoroutier, le réseau téléphonique, et le réseau électrique. Cette mise en réseau des objets standardisés a fait apparaître les limites de l'ingénierie industrielle et ainsi naître la nécessité d'établir une ingénierie des systèmes complexes. En effet, plusieurs effets indésirables et inattendus ont été rapidement constatés. Par exemple, la croissance de l'infrastructure autoroutière a produit des embouteillages, des accidents de la route, une pollution de l'atmosphère, une surconsommation des ressources énergétiques, etc. Ces propriétés émergentes causaient alors de nouveaux problèmes qui n'étaient plus exclusivement techniques, ils avaient pris une dimension sociale. Aujourd'hui, l'ingénierie des systèmes connait alors une nouvelle crise [Suh, 2016, Jackson, 2010] comme celle qu'a connu l'informatique dans les années 90. Les entreprises n'arrivent plus à tenir les contraintes de coûts et délais, et les premières versions du système souffrent d'un manque de maturité considérable. C'est pour cela que de Weck identifie le besoin d'une nouvelle ingénierie des systèmes que nous baptisons : ingénierie des systèmes de systèmes socio-techniques complexes. Systèmes de systèmes complexes, car les systèmes que nous développons aujourd'hui résultent de l'intégration de systèmes techniques complexes. Par exemple, l'intégration du

système complexe de transport et du système complexe de communication a donné naissance au système de systèmes complexes GPS. L'intégration du système complexe de transport et du système complexe électrique a donné naissance au système de systèmes complexes des véhicules électriques. Par ailleurs, ces systèmes ne sont plus purement techniques. Cette nouvelle ingénierie doit permettre aux ingénieurs de concevoir des solutions qui tiennent compte des effets sociaux à long terme. Par exemple, le système de systèmes correspondant au réseau des véhicules électriques répond à la problématique du développement durable. Dans cette nouvelle ingénierie système, les systèmes socio-techniques et les régulations doivent être co-conçus simultanément au risque de constater une régulation qui a toujours un temps de retard sur les inventions techniques. L'ingénierie contemporaine s'applique donc à des systèmes dont la complexité n'est plus exclusivement technique, elle est également sociale et ne vise plus à satisfaire un besoin personnel ou standard, mais celui d'une société dynamique. Ces travaux de thèse sont appliqués à des systèmes complexes qui, pour nous, sont devenus la norme. Néanmoins, on peut distinguer deux systèmes : le système à faire – le produit – et le système pour faire – le projet – [AFIS, 2012b]. C'est au système à faire que nous nous intéressons. Pour éviter les confusions nous préférons donc le terme « système produit » écourté à celui de « produit ».

FIGURE 1.17 – L'évolution de l'ingénierie à travers le temps.

Parmi toutes les méthodes d'ingénierie les plus populaires [Asimow, 1962, Suh, 1990, Pugh, 1991, VDI - Association of German Engineers, 1993, Hubka and Eder, 1995, Pahl et al., 1996, Cross, 2008, Ullman, 2015, Klaus Pohl, 2015] qui doivent permettre la conception de systèmes complexes, on retrouve un processus qui décrit une succession d'étapes qu'un concepteur doit réaliser pour obtenir une solution à un problème (socio-)technique. Toutes ces méthodes de conception de systèmes complexes contiennent une première étape dédiée à la définition du problème, à l'analyse du besoin. Même si aucun processus ne tient compte d'une étape de synthèse des exigences qui idéalement n'est pas nécessaire, nos travaux s'inscrivent dans cette première étape d'analyse du besoin, en particulier dans la phase de « *task clarification* » telle qu'elle est définie par Klaus Pohl [2015] : « *... it is necessary to clarify the given task in more detail before starting product development. The purpose of this task clarification is to collect information about the requirements that have to be fulfilled by the product, and also about the existing constraints and their importance. This activity results in the specification of information in the form of a requirements list that focuses on, and is tuned to, the interests of the design process and subsequent working steps. The conceptual design phase and subsequent phases should be based on this document, which must be updated continuously.* ».

Positionnement au sein de la recherche académique

Bien que l'ingénierie des exigences soit une discipline dédiée à l'industrie, de nombreux instituts académiques mènent un travail théorique mais appliqué pour résoudre les problèmes pratiques de l'industrie. Ici, nous souhaitons positionner nos travaux parmi les principaux acteurs nationaux et internationaux de la recherche universitaire en ingénierie des exigences.

À l'échelle nationale, l'ingénierie des exigences n'est pas un sujet de premier ordre, mais plutôt l'expertise des partisans historiques de la méthode de l'ingénierie système. Le Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) de Toulouse contribue depuis très longtemps à l'ingénierie des exigences via des approches complémentaires telles que la logique mathématique [Garion, 2002], le traitement du langage naturel [Kang and Saint-Dizier, 2015a] ou les méthodes d'aide à la décision et d'analyse de la valeur [Zhang et al., 2013a]. L'INRIA de Rennes et de Sophia-Antipolis participent aussi à la création de connaissances en ingénierie des exigences, en particulier en exploitant les méthodes d'ingénierie dirigée par les modèles et les techniques de recherche d'informations [Sannier, 2013], les méthodes de conception par contrat [Benveniste et al., 2015b], les techniques du Web sémantique [Azmeh et al., 2013, 2014, Mirbel and Crescenzo, 2010], ou les méthodes orientées vers les buts [Mirbel and Villata, 2014, Wautelet et al., 2014, Mirbel and Villata, 2012]. Les travaux menés à l'Université Paris 1 Panthéon-Sorbonne par Colette Roland et Camille Salinesi sont plutôt orientés vers les systèmes logiciels et traitent des problématiques diverses et variées telles que la réutilisation de connaissances en ingénierie des exigences [Souag et al., 2016], ou la gestion des exigences de lignes de produits [Djebbi et al., 2008]. Parmi ces manières d'aborder l'ingénierie des exigences, le problème que nous étudions dans ces travaux se rapproche des études menées au LAAS à Toulouse [Kang and Saint-Dizier, 2015a] et à l'INRIA de rennes [Sannier, 2013].

Si à l'échelle nationale l'ingénierie des exigences n'est pas au premier plan, à l'échelle internationale il y a un peu plus d'organismes de recherche qui contribuent à son évolution. Parmi les plus prolifiques, on retrouve encore les adeptes de l'ingénierie système. La school of engineering and information technology de l'untiversité de Nouvelle-Galles du Sud à Canberra dirigée par Prof. M.J. Ryan ont, par exemple, récemment révisé le vocabulaire et les concepts du domaine [Ryan, 2013], lesquels sont aujourd'hui standardisés dans la dernière version de la norme d'ingénierie système ISO/IEC/IEEE 15288 [2015]. Le groupe de recherche Early Development Process and Systems de l'université de Aalto, Finlande, mène un projet de recherche qui vise à définir un outil permettant d'extraire, analyser et mettre en relation des exigences [Coatanéa et al., 2013]. L'équipe de Prof. J. Llorens de l'université Carlos III de Madrid étudie l'utilité des ontologies [Fraga et al., 2015] pour améliorer la qualité des exigences lors de la rédaction. Les informaticiens s'intéressent aussi à l'ingénierie des exigences. L'université catholique de Louvain a par exemple donné naissance à l'ingénierie des exigences orientées vers les buts [Dardenne et al., 1993]. Mylopoulos et al. [1992] et Yu [1996] qui exercent au sein du département informatique de l'Université de Toronto ont joué et continuent de jouer un rôle clé dans les approches orientées buts. Les travaux menés par Lockerbie et al. [2012], professeur à la City University London, lesquels se concentrent sur la spécification de systèmes numériques complexes sont aussi reconnus à l'échelle internationale. Les membres du Center for Systems and Requirements Engineering de DePaul University, lequel est mené par Dr. J. Cleland-Huang, eux, se concentrent sur la mise en œuvre des technologies de recherche d'informations et de fouille de données pour tracer, trier et prioriser les exigences. Enfin, le dernier acteur principal de la recherche académique en ingénierie des exigences est le laboratoire d'informatique CIT-IRST de l'université de Trento en Italie qui est particulièrement actif dans l'utilisation des techniques d'intelligence artificielle et d'optimisation pour prioriser les exigences [Perini et al., 2013]. D'un point de vue de la recherche internationale, notre environnement numérique fédère un ensemble de techniques qui sont étroitement liées à tous ces organismes de recherche académique mais dont la mission est originale.

1.4 Question de recherche, hypothèse et contributions

La question de recherche à laquelle ces travaux cherchent à répondre, l'hypothèse principale et les contributions sont résumées dans cette section.

Notre question de recherche

Dans cette introduction, nous avons souligné que ces travaux sont dédiés à l'ingénierie des systèmes dont la conception oblige à surmonter quatre grands challenges : la complexification technique, la complexité organisationnelle, la diversité des produits, la foisonnement des textes législatifs et réglementaires. Ce nouveau contexte d'ingénierie engendre une prolifération des exigences que les entreprises, en particulier les sous-traitants, ne sont plus capables d'appréhender dans les coûts et délais impartis. Cela fait naître un besoin théorique, méthodologique et technologique de synthèse des exigences, ce qui nous emmène à la question de recherche : comment supporter les entreprises, en particulier les fournisseurs jouant le rôle de sous-traitants, à faire la synthèse d'un corpus de documents prescriptifs rédigés en anglais dans lesquels sont spécifiés des centaines, voire des milliers d'exigences que la solution doit satisfaire ?

Notre hypothèse

La naissance du Web et la démocratisation des machines à information ont donné naissance à de nouveaux problèmes dont celui qui consiste à faire le tri entre les informations utiles et inutiles. Pour réaliser ce tri, il faut collectionner, stocker, analyser, et synthétiser les informations. Pour cela, on utilise un ensemble de connaissances informatiques : les sciences des données. Certains parlent d'intelligence artificielle ; cependant, dans ces travaux, nous préférons utiliser le terme « sciences des données » qui ne laisse aucune place au débat « métaphysique » – une machine peut-elle être intelligente ? – qui anime passionnément notre société. Ce terme est presque devenu la norme dans le jargon technique et c'est aussi le titre de la leçon inaugurale de Serge Abiteboul [2012] au Collège de France.

Parmi ces informations, la plupart sont échangées sous la forme de données textuelles dont l'analyse se fait au moyen de techniques informatiques de fouille de données textuelles (*text mining*, en anglais). Cette sous-discipline des sciences des données est, elle même, le regroupement de techniques dédiées à l'analyse de textes. Parmi elles, il y a par exemple le traitement du langage naturel, la classification et le partitionnement sémantique de textes, ainsi que la recherche, l'extraction et la visualisation de textes. Attention, il ne faut pas confondre la fouille de données (*data mining*) avec la fouille de textes (*text mining*) [Witten, 2005]. Dans la fouille de données, l'analyste cherche des patterns [4] d'informations qui sont implicites, originellement inconnus, difficilement identifiables sans moyens techniques, et potentiellement utiles. Au contraire, dans la fouille de textes, les patterns d'informations à identifier sont clairement et explicitement énoncés. Les auteurs s'efforcent à s'exprimer clairement, mais ces informations demeurent inconnues, car les restrictions cognitives ou temporelles font que l'être humain ne peut pas lire un tel volume de textes.

Les sciences des données, en particulier les techniques de fouille de données, semblent être un moyen de synthétiser un gros volume d'exigences textuelles. C'est la raison pour laquelle nous faisons l'hypothèse qu'un environnement numérique constitué de briques théoriques et technologiques issues des sciences des données permet de supporter les entreprises, en particulier les sous-traitants, à faire la synthèse de centaines ou milliers d'exigences.

À la fin de ce manuscrit, dans le chapitre 7 de validation et de discussion, nous reviendrons sur la question de recherche en apportant des éléments de réponse qui réfuteront ou corroboreront, partiellement ou totalement, cette hypothèse.

4. Des organisations possédant des propriétés caractéristiques que l'on peut observer de façon répétée.

Nos contributions

La contribution principale de ces travaux de recherche est la définition d'un cadre opérationnel, fonctionnel, organique et logiciel permettant de supporter la synthèse collaborative de plusieurs centaines ou milliers d'exigences. Nous insistons sur le fait que c'est cette définition d'une méthode outillée et intégrée répondant au problème de synthèse d'exigences qui se démarque des contributions esseulées de l'état de l'art.

Au-delà de ce cadre de synthèse des exigences, diverses contributions techniques via des solutions, nouvelles ou différentes, mériteront sans doute d'être reprises et améliorées par d'autres acteurs de la recherche pour des objectifs communs ou différents du notre. Parmi ces contributions techniques on retrouve :

- Extraire les exigences :
  - L'intégration de différents parseurs avec un chaînage d'opérations de traitement du langage naturel pour extraire les exigences de documents non structurés et semi-structurés (Sec. 4.3).
  - Un modèle de classification binaire basé sur l'apprentissage machine pour identifier les exigences faisant référence à des documents externes applicables (Sec. 4.3).
- Fiabiliser les exigences :
  - L'application de techniques de traitement du langage naturel pour identifier les défauts de qualité intra- et inter-exigences afin de réduire le champ des interprétations d'une exigence (Sec. 5.3).
  - Des représentations graphiques interactives pour analyser et nettoyer les défauts de qualité intra- et inter-exigences (Sec. 5.3).
  - La définition de la notion de contexte d'une exigence incluant le contexte du document, le contexte sémantique et conceptuel, ainsi que le contexte des références transversales (Sec. 6.3).
- Explorer les exigences :
  - Un modèle de classification automatique multi-classes basé sur l'apprentissage machine pour catégoriser les exigences au sein de métiers (Sec. 6.3).
  - La comparaison d'une technique de détection des communautés basée sur la théorie des graphes avec la technique de partitionnement sémantique LinLog pour segmenter les exigences (Sec. 6.3).
  - Un espace collaboratif et multi-disciplinaire pour estimer des critères d'aide à la décision, ainsi qu'un tableau de bord constitué de résumés statistiques graphiques pour prendre des décisions stratégiques informées à partir d'une synthèse des estimations (Sec. 6.3).

Somme toute, notre proposition contribue aussi au passage du chaos non structuré quasi-inexploitable vers un modèle orienté graphe exploitable dans lequel les exigences, leurs propriétés, et certaines de leurs interdépendances sont formellement modélisées.

1.5 Structure du manuscrit

La suite du manuscrit se divise en huit chapitres dont le contenu de chacun est synthétisé ci-après. La séquence des chapitres suit la logique du processus outillé de synthèse des exigences. Ainsi, la lecture doit être linaire sous peine de compromettre la fluidité et la compréhension de la présentation des travaux de recherche.

Le chapitre 2 est un état de l'art macroscopique qui synthétise et critique les grandes approches qui pourraient résoudre le problème de la prolifération des exigences. L'état de l'art est abordé selon trois perspectives. La première perspective industrielle expose les différents processus d'ingénierie des exigences tels qu'ils sont décrits par les normes d'ingénierie système et les polices de grandes entreprises. La deuxième perspective technologique est un état de l'art des outils d'ingénierie des exigences. Enfin, la troisième partie est une revue de la littérature académique relative à notre problème.

Le chapitre 3 est la définition de la proposition selon quatre perspectives. La définition opérationnelle présente, d'une part, l'environnement numérique de synthèse des exigences comme une « boîte

noire » en relation avec les éléments de l'environnement et, d'autre part, le scénario opérationnel d'utilisation. C'est une description des services que l'environnement doit fournir aux utilisateurs pour réaliser la mission de synthèse des exigences : Qu'est-ce que l'utilisateur doit être capable de faire avec l'environnement numérique ? La deuxième perspective est une définition fonctionnelle décrivant l'agencement logique des fonctions techniques que l'environnement numérique doit implémenter pour fournir les fonctions de service : Qu'est-ce que l'environnement numérique doit être capable de faire pour rendre les services nécessaires à la synthèse des exigences ? La troisième perspective est la définition organique de l'environnement numérique. L'architecture organique est une décomposition structurée décrivant l'agencement des organes logiciels qui implémentent les fonctions techniques de la perspective fonctionnelle, ainsi que les liens d'interfaces entre les organes. Pour conclure, l'architecture logicielle multi-couches du prototype et le modèle de données orienté graphe sont présentés.

Les chapitres 4, 5 et 6 correspondent aux trois fonctions de service nécessaire à la mission de synthèse des exigences. Chacun de ces chapitres est systématiquement fondé sur la même structure : introduction, état de l'art, proposition, expérimentation et synthèse. Une première section d'introduction non seulement rappelle la fonction de service et les fonctions techniques à réaliser, mais aussi synthétise la structure du chapitre. Une deuxième section est un état de l'art spécifique des solutions pré-existantes. La troisième section correspond à la proposition d'une solution. La quatrième section présente une ou plusieurs expérimentations utiles à la vérification de la proposition. Enfin, pour conclure, la quatrième section résume les contributions nouvelles ou différentes avant d'exposer les limites et les perspectives d'évolution de la proposition.

Le chapitre 4 correspond à la fonction de service qui doit permettre à un analyste d'extraire les exigences spécifiées dans des documents, et d'identifier les exigences qui font références à des documents externes. Nous proposons de chaîner des parseurs avec des opérations de traitement du langage naturel pour extraire les exigences, ainsi qu'une fonction de classification basée sur l'apprentissage automatique pour identifier les références transversales.

Le chapitre 5 correspond à la fonction de service qui doit permettre à un analyste de fiabiliser l'interprétation des exigences. Nous proposons des techniques pour identifier, signaler et supprimer les défauts intra- et inter-exigences qui augment l'ambiguïté.

Le chapitre 6 correspond à la fonction de service qui doit permettre au manageur d'examiner l'espace prescriptif engendré par les exigences afin qu'il puisse prendre des décisions informées. Nous proposons de segmenter les exigences en communautés afin d'assurer le passage à l'échelle sur un gros volume d'exigences. Un environnement de vote collaboratif permet à des experts d'estimer les critères d'aide à la décision associés à chacune des communautés. Enfin, un tableau de bord de visualisation analytique basé sur la statistique descriptive synthétise les estimations des experts et permet de simuler des scénarios du type « What-if ? ».

Le chapitre 7 est une validation de l'environnement numérique de synthèse des exigences. Tandis que les sections d'expérimentation de chaque chapitre servent à vérifier chacune des unités logicielles, la section de validation consiste à s'assurer que l'environnement numérique est conforme au besoin pour l'usage prévu.

Le chapitre 8 est une conclusion générale des travaux de thèse et un ensemble de perspectives de recherche.

Chapitre 2

ÉTAT DE L'ART GÉNÉRAL

2.1 Introduction

Avant d'exposer notre proposition, il est indispensable de faire un tour d'horizons des pratiques industrielles, des outils commercialisés et des dernières avancées académiques en ingénierie des exigences. Ce chapitre est un état de l'art général selon trois perspectives : industrielle, technologique et académique. L'état de l'art industriel présente l'ingénierie des exigences, en particulier ses processus, telle qu'elle est pratiquée par de grands groupes industriels et préconisée par les normes. La mise en œuvre de l'ingénierie des exigences nécessite des outils numériques de développement et de gestion des exigences. Les principales solutions commerciales sont introduites dans l'état de l'art technologique. Enfin, l'état de l'art académique, lui, est un aperçu général des propositions académiques relatives à l'ingénierie des exigences. Les solutions proposées se répartissent en deux catégories : les approches (semi-)formelles et les approches de fouille de données.

Cet état de l'art est volontairement synthétique. En effet, dans ce chapitre, nous souhaitons mettre en exergue trois constats clés : (1) en pratique, les solutions pragmatiques jouent un rôle tout aussi important que les processus systématiques d'ingénierie des exigences ; (2) les solutions technologiques ne proposent aucun moyen de synthèse des exigences ; (3) la recherche académique en ingénierie des exigences a donné naissance à de nombreuses contributions parmi lesquelles très peu, voire aucune d'entre elles vise à synthétiser un gros volume d'exigences, mais l'intégration de certaines propositions peut nous aider à répondre à notre question de recherche.

2.2 État de l'art industriel

L'ingénierie système, laquelle inclut l'ingénierie des exigences, est une méthode scientifique collaborative et interdisciplinaire destinée à l'industrie. En effet, c'est pendant la période transitoire entre la 2^{nde} guerre mondiale et le début de la guerre froide que l'armée, l'industrie et la recherche académique américaine ont uni leurs efforts contre l'Union Soviétique dans une course à l'armement, notamment dans le programme balistique intercontinental Atlas. Les organisations gouvernementales du ministère de la défense américain et des agences spatiales – la NASA aux U.S.A et l'ESA en Europe – ont été et demeurent des acteurs majeurs de la définition et de la mise en œuvre de l'ingénierie système.

L'ingénierie des exigences étant vouée à l'industrie, nous commençons par présenter un état de l'art industriel qui décrit le processus d'ingénierie des exigences tel qu'il est mis en œuvre par certains grands groupes. Cette synthèse résulte de quelques manuscrits de thèses supportées par des industriels et consensus standardisés dans les normes et les manuels.

L'ingénierie des exigences chez Airbus

Dès le début du développement de l'avion de ligne civil A380, le constructeur aéronautique Airbus Opérations a organisé des ateliers de travail avec les compagnies aériennes « *Customer Focus Group*

(CFG) » [de Chazelles et al., 2004]. Ces ateliers durent pendant 2 à 3 jours et servent à élucider les besoins des clients potentiels. Les besoins sont de deux types : génériques ou spécifiques. Les besoins génériques sont communs à toutes les compagnies aériennes : fiabilité opérationnelle élevée, coûts de maintenance minimes, etc. Les besoins spécifiques à chacune d'entre elles sont des facteurs de compétitivité : motorisation, confort de la cabine, etc. Les besoins des parties prenantes utilisatrices du système avion sont ensuite formalisés afin de devenir les exigences de haut niveau qui pilotent le développement de l'avion.

D'une manière plus détaillée le processus d'ingénierie des exigences débute par l'accueil des compagnies aériennes qui apportent leurs besoins personnels tels qu'ils les perçoivent. Airbus, ses partenaires de premier rang, et les compagnies aériennes font, chacun à leur tour, des présentations utiles à la définition d'une vision partagée des enjeux communs. Les réactions des clients potentiels sont écrites dans une feuille de capture « *capture sheet* » telle que celle présentée figure 2.1. Airbus est ensuite responsable de sélectionner une partie des besoins selon le point de vue des parties prenantes utilisatrices de l'avion, puis de les transformer en exigences selon le point de vue des parties prenantes réalisatrices de l'avion. Les besoins non retenus servent de référence pour la définition des options. Après avoir collecté les besoins, Airbus développe des solutions de conception qui sont proposées au CFG afin de les valider, c'est-à-dire de s'assurer qu'elles correspondent au besoin des compagnies aériennes. Cette énième rencontre client-fournisseur est aussi l'opportunité d'identifier des manques éventuels. Le référentiel des exigences spécifiées et des solutions de conception préférées pilotent la phase fabrication. Enfin, les CFGs permettent de constater les déviations entre le produit réalisé « *as-built* » et le besoin exprimé par le client « *as-required* ».

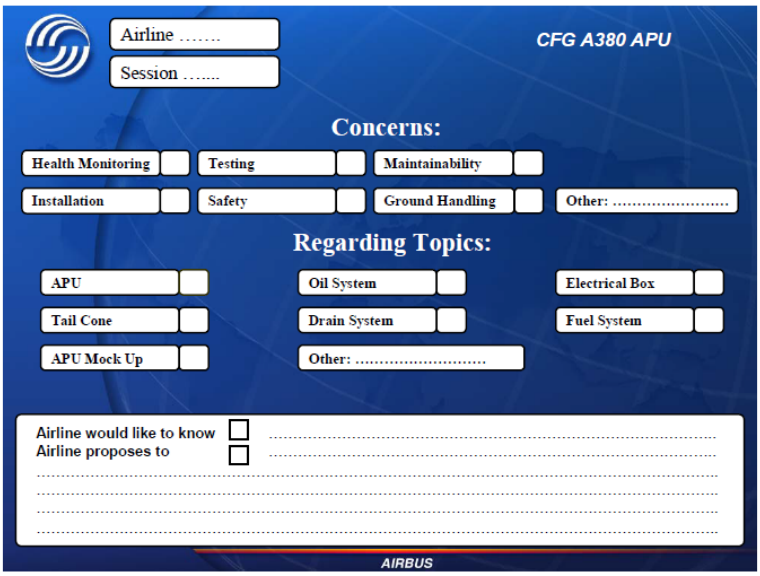


FIGURE 2.1 – « *Capture sheet* » utilisée par Airbus pour collecter les besoins des compagnies aériennes [de Chazelles et al., 2004].

L'ingénierie des exigences chez Rolls-Royce

Rolls-Royce s'appuie sur un ensemble de bonnes pratiques accumulées pendant leurs derniers projets tels que le « *Trent XWB* » et sont exposées par Zhang [2012]. Le processus débute par l'élucidation des exigences des clients, lesquelles complètent les exigences techniques développées dans les projets antérieurs. Les exigences élucidées sont alors analysées afin d'identifier des défauts de qualité (incomplétude, ambigüité lexicale, etc.) et de les catégoriser (exigence fonctionnelle *vs.* exigence non-fonctionnelle). Une seconde phase d'analyse permet d'identifier les exigences dites « basiques ». Les exigences basiques sont des exigences qui prescrivent des facteurs de base correspondant à des attentes

qui n'apportent pas de satisfaction particulière au client, mais qui sont indispensables [Badreau and Boulanger, 2014]. Par exemple, la ponctualité des moyens de transports publics (bus, trains, avions) est une exigence basique, car, à première vue, elle semble évidente mais le client est extrêmement déçu dès qu'elle n'est pas satisfaite. Rolls-Royce conduit ensuite des analyses fonctionnelles du système pour identifier les interfaces, les dépendances inter-exigences, et potentiellement des exigences initialement omises. Comme pour l'identification des exigences basiques, des analyses de sensibilité sont menées pour identifier les exigences qui prescrivent des facteurs d'enthousiasme. Un facteur d'enthousiasme est un facteur dont l'implémentation procure une satisfaction supplémentaire non négligeable, voire l'émerveillement, et dont l'absence n'engendre pas de frustration particulière [Badreau and Boulanger, 2014]. Le fournisseur doit néanmoins s'assurer que le client est prêt à payer pour chacun de ces facteurs d'enthousiasme. Enfin, pour dériver les exigences du système aux sous-systèmes, Rolls-Royce réalise une analyse d'aide à la décision grâce à la méthode matricielle QFD « *Quality Function Deployment* ». Enfin, des cas d'utilisation et des scénarios sont développés pour formaliser les exigences.

L'ingénierie des exigences chez Volvo Aéro

Comme chez Rolls-Royce, Volvo possède ses propres pratiques d'ingénierie des exigences, lesquelles sont aussi exposées par Zhang [2012]. Volvo Aéro commence par prospecter les besoins et les insatisfactions des clients en organisant des ateliers collaboratifs de réflexion. Au cours de ces ateliers, les acteurs échangent autour d'artefacts graphiques (maquettes, schémas, etc.) utiles pour se remémorer des situations analogues passées ou imaginer des situations futures. L'identification des exigences du système passe alors par une décomposition des objectifs stratégiques définis par l'*Advisory Council for Aeronautic Research* et des organisations gouvernementales telles que *International Civil Aviation Organization*, la *Federal Aviation Administration* et la *European Aviation Safety Strategy Agency*. Les exigences systèmes sont alors décomposées au niveau des sous-systèmes grâce aux connaissances métiers de quelques concepteurs impliqués dans chacun des sous-systèmes. Les exigences sont ensuite gérées tout au long de la phase de conception. La gestion consiste à tracer les modifications et vérifier les nouvelles versions. Une enquête bi-annuelle de satisfaction est menée auprès des clients afin d'opposer Volvo Aéro à ses concurrents. Ces enquêtes servent à évaluer des aspects précis sous la forme de questions telles que : « Que pensez-vous des performances de Volvo Aéro vis-à-vis de cet aspect en particulier ? ». En plus d'une enquête bi-annuelle de satisfaction, Volvo Aéro réalise une enquête bi-annuelle de marque. Cette dernière se déroule sous forme d'entretiens avec des manageurs d'entreprises clientes et servent à identifier les caractéristiques de la marque Volvo Aero et celles qu'ils souhaiteraient trouver dans le futur.

L'ingénierie des exigences à la NASA

La NASA qui fut l'une des organisations à l'origine de l'ingénierie système décrit trois processus d'ingénierie des exigences : (1) le processus de définition des attentes des parties prenantes ; (2) le processus de définition des exigences techniques ; et (3) le processus de gestion des exigences [NASA, 2007].

Le processus de définition des attentes des parties prenantes (Fig. 2.2) démarre par l'identification des parties prenantes. Ces dernières sont ensuite concertées afin de définir la mission, les objectifs et les critères de vérification du système. Ce processus doit permettre d'aboutir à une définition opérationnelle du système répondant à la question : Qu'est-ce que les utilisateurs doivent être capables de faire avec le système ?

Le processus de définition des exigences techniques transforme les attentes des parties prenantes en un ensemble d'exigences que le système doit satisfaire. Ces exigences sont ensuite dérivées aux sous-systèmes, et ce, de manière récursive, jusqu'aux constituants élémentaires. À la différence des attentes, les exigences satisfont divers critères de qualité qui font qu'elles doivent être complètes, non ambiguës, réalisables, singulières, etc. Pour chaque bloc constitutif, les exigences techniques doivent spécifier les entrées, les sorties et les relations entrée-sortie.

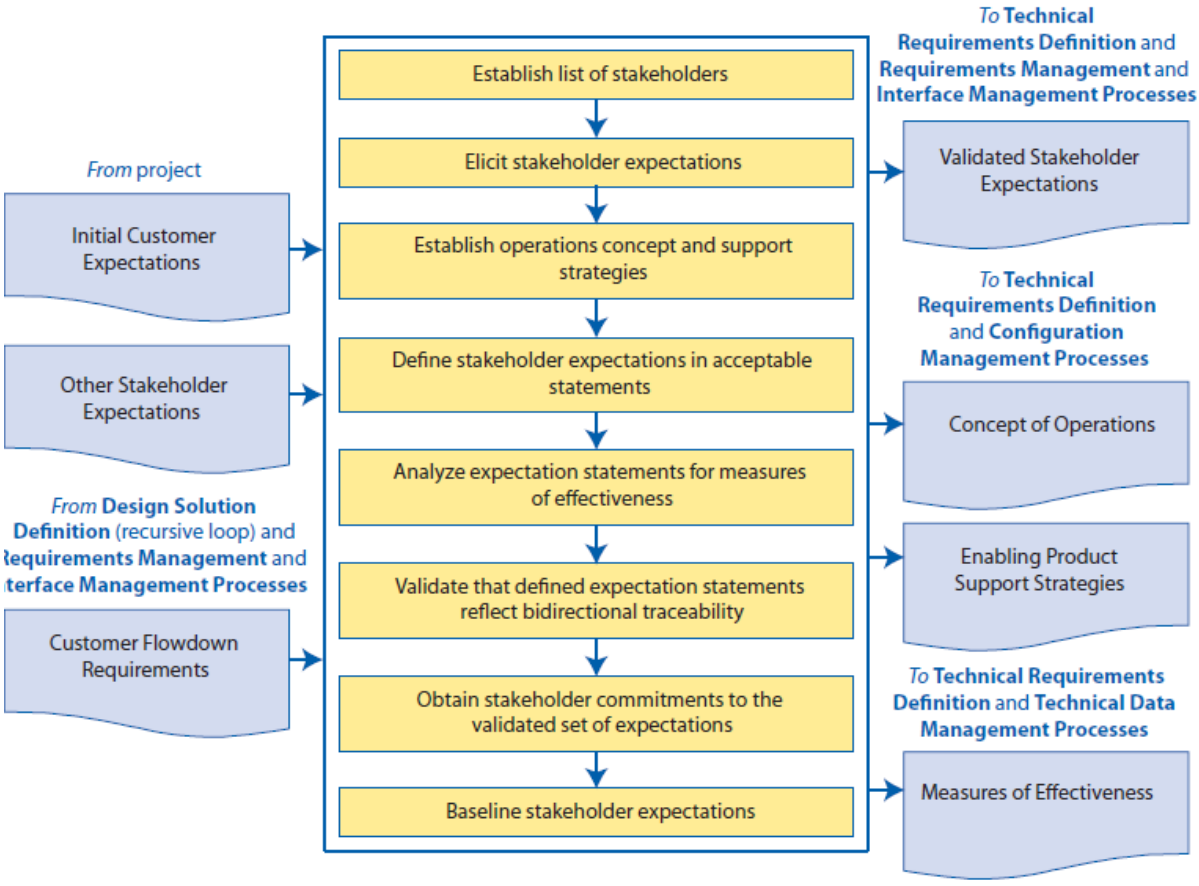


FIGURE 2.2 – Processus de définition des besoins des parties prenantes à la NASA [NASA, 2007]

.

Le processus de gestion des exigences est utile pour non seulement tracer les exigences, mais aussi gérer les modifications appliquées au référentiel d'exigences. Un plan d'exécution de gestion des exigences est initialement créé, après quoi les exigences sont organisées dans une structure arborescente. Chacune des exigences est alors tracée vers une exigence appartenant au niveau systémique supérieur. Pendant le processus de gestion des exigences il est important de s'assurer que chacune des exigences est validée, c'est-à-dire qu'elle résulte directement ou indirectement d'une attente d'une partie prenante. Par ailleurs, chacune des exigences doit être vérifiable, c'est-à-dire qu'une méthode de vérification (simulation, inspection, etc.) puisse vérifier que la solution de conception satisfait aux exigences. Enfin, quand les exigences sont à un niveau de maturité acceptable, elles doivent être figées dans une configuration de référence, laquelle sert de référentiel pour évaluer et exécuter les demandes de modification. Une activité de maintien en consistance entre les scénarios opérationnels, les exigences et les architectures garantit le bon déroulement du processus.

L'ingénierie des exigences selon l'INCOSE

Parmi les processus techniques d'ingénierie système, l'INCOSE recommande aussi deux processus. Le premier est dédié à la définition des besoins des parties prenantes à partir du concept d'opérations (ConOps). Les besoins, souvent ambigus et incomplets, sont ensuite transformés en exigences que l'on stocke dans la « Spécification des Exigences des Parties Prenantes » (*Stakeholder Requirements Specification* - StRS, en anglais). Le second processus décrit les activités, entrées et sorties nécessaires à la transformation des exigences des parties prenantes en exigences système prescrites dans une « Spécification des Exigences Système » (*System Requirements Specification* - SyRS, en anglais). Le processus est non seulement itératif entre les besoins et les exigences, mais aussi récursif du niveau système jusqu'aux

constituants élémentaires (Fig. 2.3).

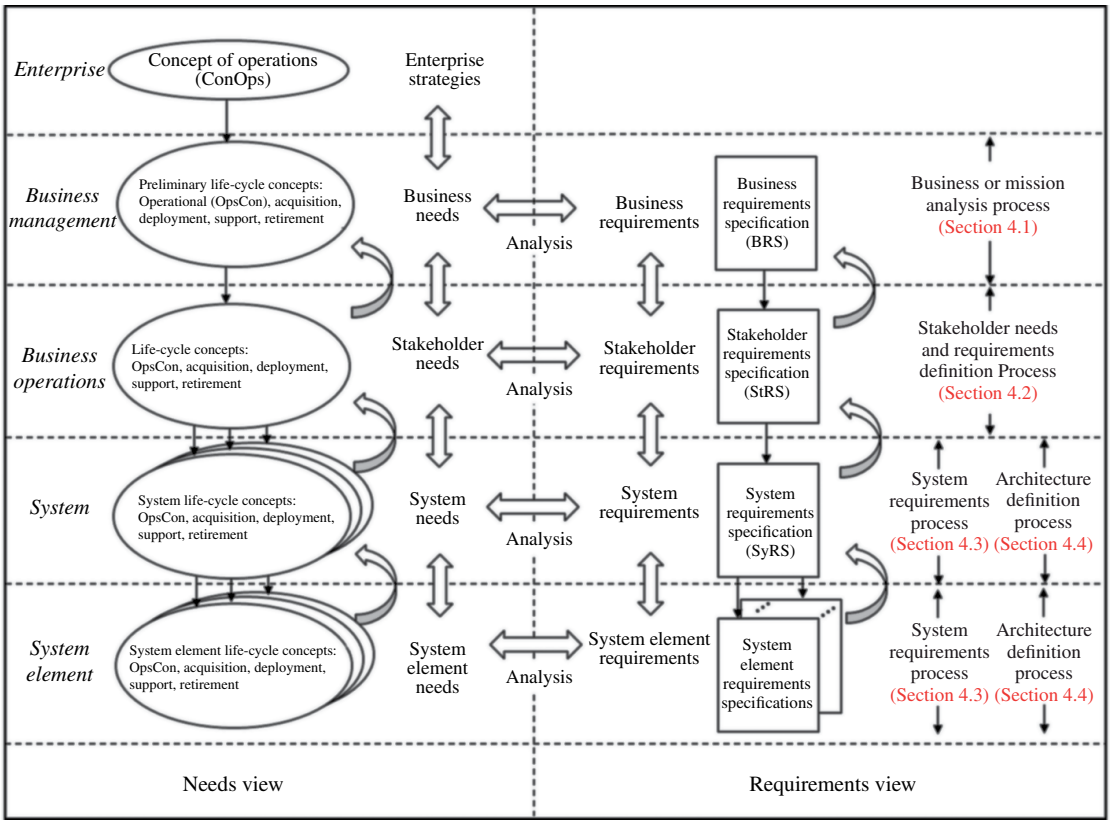


FIGURE 2.3 – Transformation des besoins en exigences [INCOSE, 2015b] selon Ryan [2013]

.

L'ingénierie des exigences selon l'ISO 29148

La norme international ISO/IEC/IEEE 29148 [2011] « *Systems and software engineering – System life cycle processes* » définit un ensemble de processus qui décrivent le cycle de vie d'un système. La figure 2.4 montre que le premier processus est celui qui permet la définition des besoins des parties prenantes. C'est la description opérationnelle de ce que les utilisateurs doivent être capables de faire avec le système. Pour cela, la norme recommande de : (1) identifier les parties prenantes qui interagissent, directement ou indirectement, avec le système dans une ou plusieurs phases du cycle de vie ; (2) élucider les besoins des parties prenantes ; (3) formaliser les besoins en exigences ; (4) analyser, c'est-à-dire identifier et prioriser les exigences conflictuelles, manquantes, incomplètes, ambiguës, inconsistantes, incongrues ou invérifiables. Un référentiel d'exigences selon le point de vue des clients matérialise le résultat du processus de définition des besoins des parties prenantes.

Le deuxième processus consiste à analyser les exigences. L'analyse des exigences doit permettre de s'assurer de la qualité des exigences et de leur faisabilité technique. En effet, au cours du processus précédent, les exigences peuvent être incomplètes, car les parties prenantes sont incapables de toutes les recenser. Par exemple, les exigences de fabrication et de sureté sont méconnues du client. Ainsi, les parties prenantes réalisatrices du système sont là pour s'assurer que la spécification est complète et les exigences réalisables.

Dès que les exigences du système sont définies, il faut concevoir des architectures du système et allouer les exigences aux blocs constitutifs. Le processus est récursif. Selon les solutions de conception, de nouvelles exigences sont dérivées au niveau systémique inférieur à la manière de ce que Suh [2001] appelle le *zigzagging* dans sa théorie de la conception axiomatique.

Enfin, après avoir vérifié et validé les exigences, il est indispensable de les gérer tout au long du cycle de vie du produit.

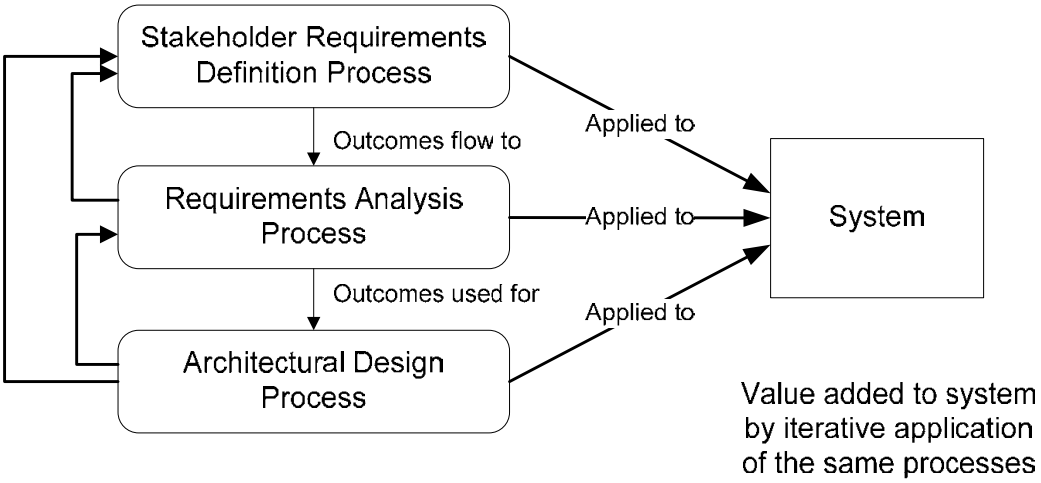


FIGURE 2.4 – Processus d'ingénierie des exigences selon l'ISO 29148 [ISO/IEC/IEEE 29148, 2011].

L'ingénierie des exigences selon l'EIA 632

La norme américaine ANSI/EIA 632 [2003] « *Processes for Engineering a System* » fournit un ensemble de processus pour aider les entreprises qui souhaitent faire l'ingénierie d'un système à partir des meilleures pratiques développées pendant la seconde moitié du 20ème siècle. En plus des processus classiques de définition des exigences avec une perspective client et une perspective fournisseur, une partie de cette norme normalise le processus d'acquisition et livraison.

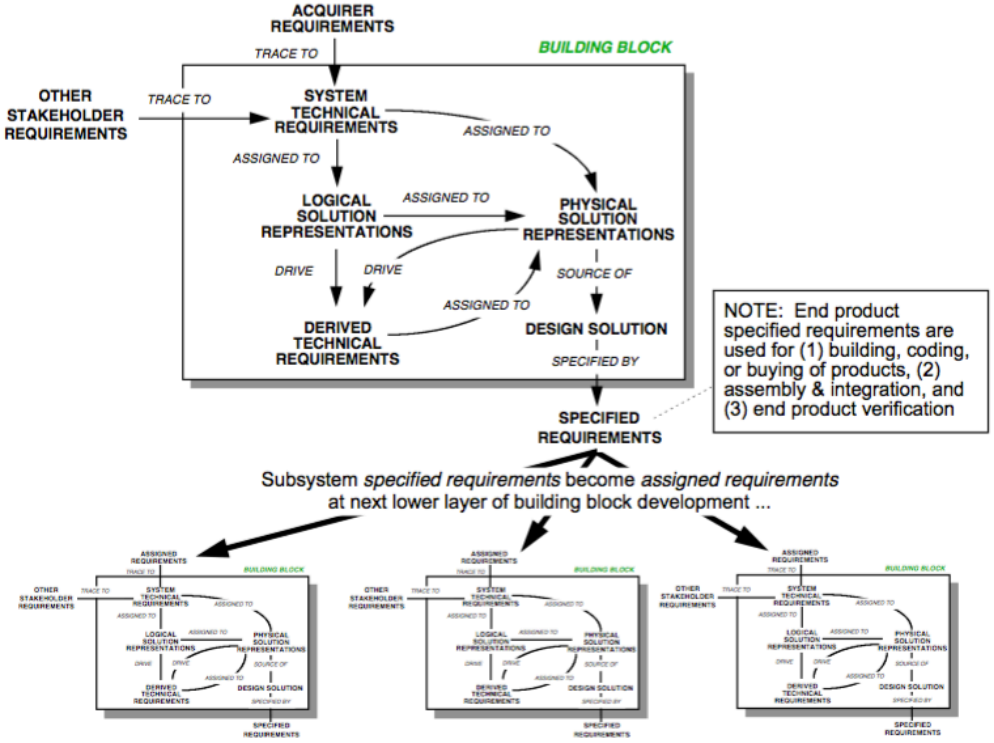


FIGURE 2.5 – Le processus d'ingénierie système selon l'ANSI/EIA 632 [2003].

Le processus de livraison est appliqué par le fournisseur afin de définir et satisfaire les accords établis avec le client. Ainsi, le fournisseur doit commencer par étudier l'appel d'offre afin de savoir si son organisation est capable de satisfaire les exigences du client. Dès que les exigences du client ont été validées, il faut contractualiser selon les contraintes applicables, puis enregistrer le contrat avant d'appliquer le processus d'ingénierie système défini dans la norme. Enfin, les produits sont livrés tels qu'ils ont été spécifiés dans le contrat. Le processus d'acquisition décrit dans l'EIA 632 est très similaire au processus de livraison, mais il est mis en œuvre par le client.

La première activité qui correspond à « étudier l'appel d'offre », brièvement mentionnée dans le processus de livraison, est un cas, sinon le meilleur, qui puisse donner sens à notre problématique, car c'est à ce moment précis que le fournisseur hérite d'un amas d'exigences. Certes, la norme mentionne l'existence d'un passage de témoins client-fournisseur, mais ne donne aucune préconisation méthodologique.

Concernant le processus d'ingénierie des exigences lui-même, il est plus ou moins identique aux précédents. Dans tous les processus on retrouve, d'un côté, les exigences initiales du client et, de l'autre, les exigences techniques du fournisseur (Fig. 2.5). On remarquera que les termes utilisés varient d'une référence à l'autre. Les exigences du client sont appelées : « besoins », « exigences des parties prenantes », « exigences des parties prenantes utilisatrices » « exigences de l'acquéreur », etc. Les exigences du fournisseur sont appelées : « exigences système », « exigences techniques », « exigences des parties prenantes réalisatrices », etc. Néanmoins, on retrouve toujours une opposition entre deux perspectives : le client et le fournisseur. Par ailleurs, on retrouve le processus de dérivation des exigences tout au long de la décomposition systémique du système.

Synthèse de l'état de l'art industriel

Les processus présentés ci-dessus laissent transparaître que l'ingénierie des exigences, sous-discipline de l'ingénierie système, repose sur deux processus : le développement des exigences et la gestion des exigences [Badreau and Boulanger, 2014, Wiegers and Beatty, 2013, CMMI Product Team, 2010]. Le développement des exigences permet de définir un référentiel d'exigences, tandis que la gestion des exigences consiste à suivre l'évolution du référentiel tout au long du cycle de vie du produit.

Cette description de l'ingénierie des exigences est idéaliste. Attention, nous ne remettons pas en cause les fondamentaux de l'ingénierie des exigences ou, plus généralement, de l'ingénierie système [Hall, 1962], mais la manière dont ceux-ci sont mis en œuvre et outillés dans certaines entreprises. La méconnaissance ou une application négligée de ces fondamentaux est l'une des causes de la prolifération des exigences. La présentation de l'ingénierie des exigences est alors idéaliste pour deux grandes raisons.

D'une part, leur présentation séquentielle laisse parfois penser à tort que l'on peut, dans une première phase, spécifier un produit de A à Z puis, dans une seconde phase, concevoir une solution et gérer les exigences. Le processus de conception d'un produit, en particulier celui de l'ingénierie système, est itératif entre le domaine du problème et celui de la solution. En effet, à la manière de la Méthode Scientifique, l'ingénierie système exige de : (1) bien poser le problème (spécification et validation) ; (2) élaborer des hypothèses de solutions (conception) ; et (3) évaluer les hypothèses préférées (vérification). En plus d'être itératif à un niveau systémique donné, le processus est récursif. Comme le préconise le deuxième précepte du réductionnisme cartésien [1], l'ingénierie système applique une décomposition systémique – du niveau système vers les constituants élémentaires – qui permet de simplifier un problème compliqué.

D'autre part, l'ingénierie des exigences telle que nous pouvons la lire est idéaliste, car, comme toujours, la théorie est idéale et la pratique est pragmatique. Il est malheureusement évident que le droit de réserve fait que les entreprises ne vont pas témoigner sur la place publique des écueils à éviter qu'ils n'ont pas assez considérés consciemment ou inconsciemment. En effet, en pratique, dans une entreprise, étendue ou non, les processus formels laissent souvent place au pragmatisme des « inter-subjectivistes » ; ces amateurs des méthodes agiles, dont le slogan pourrait être « *people rather than processes* ». Par

1. « Le second, de diviser chacune des difficultés que j'examinerais, en autant de parcelles qu'il se pourrait, et qu'il serait requis pour les mieux résoudre. » [Descartes, 1637]

exemple, alors que les processus d'ingénierie système en général, et d'ingénierie des exigences en particulier, sont appliqués avec plus ou moins de succès chez les intégrateurs de systèmes, les sous-traitants du 2^e, 3^e,... , et N^e rang sont rarement invités à prendre part au jeu, mais héritent d'un référentiel d'exigences qui, après un long voyage, s'est irrationnellement enrichi. Ainsi, les difficultés rencontrées par les sous-traitants ne sont pas traitées par les processus d'ingénierie système, mais sont surpassées, tant bien que mal, par des moyens *ad-hoc*. Laudan et al. [2009] avaient déjà conclu que la phase d'analyse des exigences est pilotée par la connaissance et la collaboration plutôt que des processus rigoureusement séquencés. Les solutions pragmatiques, nourries par les connaissances tacites, consistent à mettre en œuvre des actions pratiques afin de s'adapter à toute situation. La plupart des sous-traitants du type TPE/PME adoptent une ingénierie routinière et, quand le besoin s'avère vraiment nécessaire, elles appliquent un minimum d'actions correctives pour obtenir une solution de conception qui satisfait les exigences. Pour cela, ils mettent en œuvre des méthodes agiles. Par exemple, des processus très itératifs ponctués de réunions. Ils s'appuient aussi sur des guides de bonnes pratiques qui résultent d'une capitalisation du savoir-faire interne à une organisation ou d'un consensus entre des tierces organisations. En ingénierie système, des associations savantes comme l'INCOSE ou l'Association Française d'Ingénierie Système (AFIS) ont publié des guides de bonnes pratiques en ingénierie des exigences [AFIS, 2012a, INCOSE, 2015a] et en expression du besoin [AFIS, 2014]. En ingénierie des exigences, l'action pragmatique est coutumière. Les besoins sont collectés au cours d'entretiens informels client-fournisseur ou de lectures minutieuses de documents applicables. Des analystes diagnostiquent manuellement les défauts de qualité dans les exigences et les spécifications. Des réunions en comité restreint servent à prioriser des exigences. Une telle approche traite, dans l'urgence ponctuelle, les symptômes de la prolifération des exigences, mais n'apporte que très peu, voire aucun remède aux causes de cette prolifération. Les entreprises sont néanmoins conscientes des limites du pragmatisme et se tournent vers des approches plus académiques. L'approche (semi-)formelle (Sec. 2.4) notamment.

Si d'un point de vue des processus industriels l'activité de synthèse des exigences est absente, nous pouvons nous interroger sur l'existence de technologies qui pourraient potentiellement supporter les sous-traitants en difficultés.

2.3 État de l'art technologique

Comme les processus d'ingénierie des exigences, les technologies d'ingénierie des exigences peuvent être catégorisées selon deux classes. La première correspond aux technologies de développement des exigences, tandis que la seconde regroupe les outils de gestion des exigences.

Les technologies qui supportent le développement des exigences

En haut de l'échelle de popularité des technologies qui supportent le développement des exigences on retrouve les outils de bureautique dédiés au **traitement de texte** tels que MS Word et OpenOffice. La popularité de ces outils fait qu'ils ne nécessitent pas de formation du personnel. Leur déploiement peut également être réalisé sans la mise en place au-préalable d'une infrastructure informatique sophistiquée et onéreuse. Leur mise en œuvre intuitive repose sur l'absence d'un schéma structuré laissant l'utilisateur libre de toute contrainte. Les analystes rédigent les spécifications comme ils le faisaient sur des documents papiers. Cette liberté est aussi avantageuse que périlleuse. En effet, le développement collaboratif des exigences peut être menacé par les initiatives personnelles et irrationnelles. Par ailleurs, la diversité des formats de spécification nécessite une interprétation nouvelle pour chacune des lectures et une rigueur chirurgicale pour la réutilisation d'exigences d'un projet à l'autre.

Les **tableurs** tels que MS Excel ou OpenOffice Calc sont aussi des outils de bureautique utilisés pour rédiger et stocker des exigences. Une spécification prend alors la forme d'un tableau dans lequel chacune des lignes est un objet « Exigence » et chacune des colonnes est un attribut – l'auteur, l'énoncé, la priorité, ou la version d'une exigence, par exemple. La définition d'un schéma d'attributs permet de rationaliser l'expression du besoin. Tous les analystes savent ce qu'ils doivent spécifier et ne pas

spécifier. Néanmoins, il est difficile de contextualiser les exigences, c'est-à-dire de les positionner au centre d'énoncés informatifs indispensables à l'interprétation.

Sur ces outils de bureautique peuvent se greffer des **outils d'analyse des exigences**. Les plugins Requirements Authoring Tool (RAT) [Fraga et al., 2015] et Requirements Quality Analyzer (RQA) commercialisés par The Reuse Company et l'outil Semios développé par Prometil, permettent de diagnostiquer et de signaler les défauts de qualité présents dans un énoncé d'exigences. RAT et RQA s'appuient sur un glossaire qui doit être personnalisé par chacune des entreprises. La technologie Semios ne nécessite pas l'existence d'un glossaire car elle exploite un moteur sémantique implémentant des techniques de traitement du langage naturel.

La volonté des pratiquants de l'ingénierie système de remplacer, autant que faire se peut, les documents par des modèles a permis, de manière mineure, de modéliser les exigences à l'aide de **diagrammes d'exigences SysML**. Dans les digrammes d'exigences, les exigences ont toujours une forme textuelle laissant place à l'ambiguïté, mais elles sont inter-connectées par des liens de composition, dérivation, raffinement, satisfaction, vérification, copie et traçabilité. Les éditeurs SysML qui gouvernent le marché sont Cameo Systems Modeler, Enterprise Architect, Artisan Studio, et Rational Rhapsody Designer, lesquels cohabitent avec des outils open-source tels que Papyrus ou Topcased.

Enfin, on retrouve les outils de **modélisation et de simulation de systèmes**, lesquels permettent de modéliser (semi-)formellement les exigences pour les simuler numériquement. Par exemple, le module d'ingénierie système RFLP de CATIA V6 permet de vérifier qu'une solution de conception est conforme ou non aux exigences [Pinquié et al., 2015a]. Les exigences stockées dans une base de données comme IBM Rational Doors peuvent être tracées et vérifiées avec Simulink Verfication and Validation. Enfin, on retrouve l'outil Stimulus [Jeannet and Gaucher, 2016] développé par l'entreprise Argosim qui permet de modéliser les exigences fonctionnelles avec un langage spécifique afin de les valider au moyen de simulations.

Les technologies qui supportent la gestion des exigences

Les technologies qui supportent la gestion des exigences sont, pour la plupart d'entre elles, des applications collaboratives fondées sur une **base de données** relationnelle. Ces outils peuvent être autonomes ou intégrés dans un environnement numérique PLM.

Les trois principaux environnements PLM intégrés sont : Enovia de Dassault Systèmes, Windchill de PTC et Teamcenter de Siemens. Pour chacun de ces environnements, le module de gestion des exigences est respectivement dénommé Enovia Requirements Manager, Windchill Requirements Management et Teamcenter Requirements Management. Ces environnements sont livrés au client avec un modèle de données standard défini par l'éditeur. Le modèle de données est alors quasi-systématiquement personnalisé afin d'être en harmonie avec les processus du client, lequel peut ainsi implémenter le schéma d'attributs qu'il désire.

Parmi les applications autonomes, c'est-à-dire indépendantes des autres outils d'ingénierie assistée par ordinateur, on retrouve le leader du marché des technologies de gestion des exigences : IBM Rational DOORS. Dans la même catégorie, OneDesk est une plateforme collaborative dans laquelle les utilisateurs peuvent, comme dans un forum social, développer des exigences à partir de *feedbacks*, planifier de nouvelles révisions d'un produit, etc. En plus des solutions de stockage et de gestion des évolutions, il y a les outils de traçabilité pour analyser l'impact qu'a la modification d'un objet technique (exigence, solution de conception, cas de test, etc.) sur d'autres objets. Avec Reqtify, un utilisateur peut non seulement tracer des objets techniques, en particulier des exigences, en analysant des tags explicitement ajoutés par les analystes, mais aussi quantifier le niveau d'implémentation de chaque exigence.

Les technologies qui reposent sur une base de données facilitent la gestion des modifications. Les opérations de demande de modification « *Engineering Change Request (ECR)* », d'ordre de modification « *Engineering Change Order (ECO)* », d'évolution des versions et d'allocation des droits d'accès sont présentes par défaut. Grâce aux relations d'implémentation entre une exigence et un objet technique (fonction, maillage, modèle B-REP, simulation FAO, gamme d'assemblage, etc.), les environnements

intégrés ont aussi l'avantage de proposer des analyses d'impacts permettant de répertorier tous les objets techniques affectés par la modification d'une exigence.

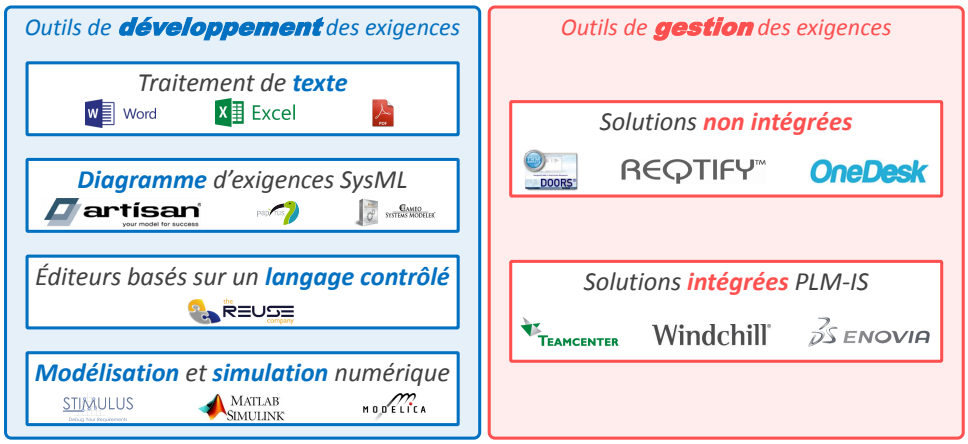


FIGURE 2.6 – Aperçu de l'état de l'art technologique dont les solutions se limitent au développement et à la gestion des exigences.

Comme on pouvait l'imaginer dès le début de cet état de l'art, les solutions technologiques d'ingénierie des exigences outillent les deux processus de développement et de gestion des exigences (Fig. 2.6) tels qu'ils sont normalisés en ingénierie système. Ils ne proposent donc aucune fonctionnalité de synthèse des exigences.

Synthèse de l'état de l'art technologique

Les outils d'ingénierie des exigences sont principalement dédiées à la gestion des exigences grâce à des bases de données relationnelles dont l'interface avec l'utilisateur se présente sous la forme d'un tableur dans lequel chaque ligne est une exigence et chaque colonne est un attribut. On constate aussi que les outils de gestion du cycle de vie du produit se sont appropriés le monopole commercial et tentent aujourd'hui de les intégrer avec les outils de conception fonctionnelle, comportementale et physique. Les technologies de développement des exigences, elles, sont encore centrées sur les documents. Cependant, quelques nouvelles innovations issues de la recherche académique exploitent de nouvelles techniques telles que les réseaux sociaux et le traitement du langage naturel pour faciliter l'activité de spécification. Malgré tous ces honorables efforts, aucune solution commerciale ne s'intéresse de près ou de loin au problème de la prolifération des exigences et cela conforte l'initiative de notre partenaire industriel.

2.4 État de l'art académique

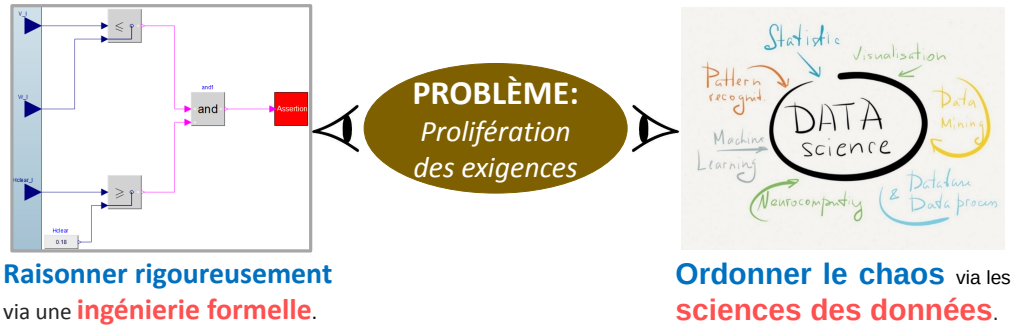


FIGURE 2.7 – Les deux grandes perspectives pour traiter le problème de la prolifération des exigences.

D'un point de vue académique, le problème de la prolifération des exigences peut être traité selon deux grandes approches (Fig. 2.7). La première consiste à appliquer les sciences des données pour essayer d'ordonner le chaos hérité du donneur d'ordres. Alternativement, une entreprise peut adopter une ingénierie formelle qui impose une certaine rigueur permettant de spécifier au juste besoin.

Les solutions (semi-)formelles

Les approches (semi-)formelles d'ingénierie des exigences s'appuient sur des modèles. Dans son sens le plus générique, un modèle peut être défini comme tel : « Pour un observateur B, un objet A* est un modèle d'un objet A s'il permet à B de répondre à une question qu'il se pose à propos de A. » [Minsky, 1965].

Exemple 1 *Soit un agent X qui désire acheter une maison. X voudrait que la maison ne soit pas proche d'une station de métro (à cause de la gêne provoquée par le bruit) ou qu'elle soit bien insonorisée ; il voudrait également que si la maison n'est pas proche du centre ville, alors elle soit proche d'une station de métro. La première exigence est prioritaire pour X, par rapport à la deuxième. On considère un langage propositionnel contenant les variables I (la maison est insonorisée), M (la maison est proche d'une station de métro) et C (la maison est proche du centre ville), alors on peut exprimer l'ensemble des exigences de l'agent par la position :* $\Gamma = [\neg M \vee I, \neg C \rightarrow M]$. *Ceci signifie que X voudrait que sa maison vérifie* $\neg M \vee I$ *et* $\neg C \rightarrow M$. *Mais si ce n'est pas possible (à cause des contraintes du domaine ou de la réglementation), il préférerait que la maison vérifie* $\neg M \vee I$ *et ne vérifie pas* $\neg C \rightarrow M$. *Si ce n'est toujours pas possible, il accepte que la maison satisfasse* $\neg C \rightarrow M$ *et ne satisfasse pas* $\neg M \vee I$. *Enfin, dans le pire des cas, il accepte que la maison ne satisfasse ni* $\neg C \rightarrow M$ *ni* $\neg M \vee I$. *Cette position est associée au pré-ordre sur les mondes possibles suivant :*

$||(\neg M \vee I) \wedge (C \vee M)|| \leq ||(\neg M \vee I) \wedge (\neg C \wedge \neg M)|| \leq ||(M \wedge \neg I) \wedge (C \vee M)|| \leq ||(M \wedge \neg I) \wedge (\neg C \wedge \neg M)||$

FIGURE 2.8 – Exigence formelle basée sur la logique mathématique [Cholvy and Garion, 2003].

Les exigences les plus formelles sont des descriptions basées sur la logique mathématique. Dans ses travaux de thèse, Garion [2002] étudie les apports de la logique mathématique en ingénierie des exigences. Ces travaux sont ensuite repris par Cholvy and Garion [2003] qui proposent de modéliser les exigences au moyen d'une logique de préférences conditionnelles. Des langages de programmation basés sur le calcul de prédicats du premier ordre comme Prolog permettent d'implémenter ces descriptions formelles. Les exigences formelles sont réservées aux logiciens qui disposent de l'expertise nécessaire à leur lecture et écriture.

Library = = [shelved:ℙ Book:readers:ℙReader:
stock:ℙ Book: issued:ℙBook]

Issue

Library
b? : Book
r? : Reader

b? ∈ shelved; r? ∈ readers
issued′ = issued ⊕ {b? - r?}
shelved′ = shelved\{b?}
stock′ = stock: readers′ = readers

FIGURE 2.9 – Exigence formelle écrite en langage Z [Hull et al., 2011].

Les langages formels tels que Vienna Definition Language (VDM) [Jones, 1990], Language of Temporal Ordering Specification (LOTOS) [ISO, 1989], Z [Spivey, 1989], Requirements State Machine Language (RSML) [Leveson et al., 1994] et B-Method [Abrial, 1996] sont aussi utilisés pour spécifier des exigences dans des domaines où la sureté de fonctionnement est critique, comme le logiciel embarqué. Parmi ces langages formels, certains sont plus utilisés que d'autres. Le langage Z et la B-Method sont les plus populaires. La figure 2.9 illustre une exigence formelle exprimée avec le langage Z, lequel est fondé sur la logique des prédicats du premier ordre et la théorie des ensembles. L'équipe projet Hycomes de l'INRIA de Rennes a récemment entamé un travail de recherche qui unifie des méthodes dites de « conception par contrat » [Benveniste et al., 2012, 2015a,b]. La conception par contrat doit faciliter la transformation du cahier des charges d'un système en une série d'exigences sous-système, lesquelles sont ensuite confiées à des sous-traitants. Une exigence spécifiée avec un tel langage graphique ou symbolique est moins formelle qu'une description logique mathématique, mais sa lecture n'est guère plus simple et ne convient qu'aux personnes formées à ces formalismes.

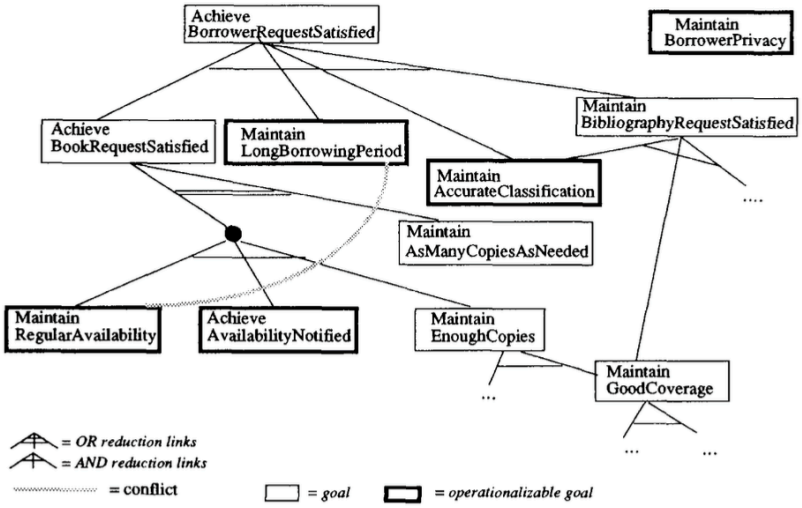


FIGURE 2.10 – Diagramme d'exigences orientées vers les buts [Dardenne et al., 1993].

L'interprétation de certaines exigences que nous qualifions de semi-formelles est plus intuitive. Les méthodes d'ingénierie des exigences orientées par les buts (*Goal Oriented Requirements Engineering* - GORE, en anglais) [van Lamsweerde, 2001] telles que KAOS [Dardenne et al., 1993, van Lamsweerde, 2004], I* [Yu, 1996], TROPOS [Bresciani et al., 2004] et le Non-Functional Requirements (NFR) Framework [Chung and do Prado Leite, 2009, Mylopoulos et al., 1992] sont des approches semi-formelles. Ces techniques semi-formelles ont une représentation graphique : un graphe constitué de buts et de portes logiques (Fig. 2.10). Cette représentation graphique peut être complétée par des descriptions littérales selon un formalisme logique plus formel.

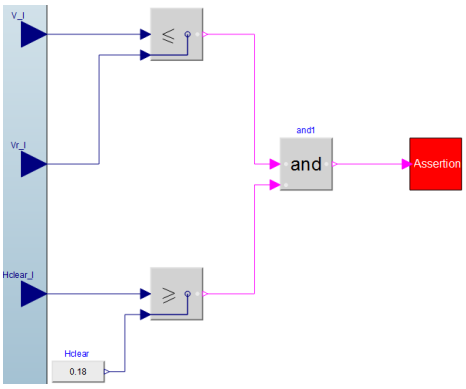


FIGURE 2.11 – PBR graphique modélisée en Modelica [Pinquié et al., 2016].

Une approche qui s'est inspirée des méthodes GORE est la théorie des exigences basées sur les propriétés (Property-Based Requirements - PBR) [Micouin, 2008]. Les PBRs modélisent les exigences d'un système selon une structure de semi-treillis. Les PBRs sont stockées dans un modèle de spécification qui est co-simulé avec un ou plusieurs modèles de conception grâce à un langage de simulation, comme VHDL-AMS [Micouin, 2014], Modelica [Pinquié et al., 2016] ou Simulink. Il est ainsi possible de valider et vérifier les exigences et les solutions de conception dans un même modèle. Si le langage de modélisation le permet, les PBRs sont multi-représentées par une représentation textuelle selon le langage de programmation utilisé et le schéma graphique associé (Fig. 2.11). La modélisation et simulation de PBRs est une solution candidate pour réduire le nombre d'exigences, mais un sous-traitant ne peut pas imposer un tel formalisme à un donneur d'ordres, c'est ce dernier qui décide.

Parmi les modèles graphiques on retrouve également les diagrammes d'exigences SysML [Object Modeling Group, 2015]. SysML est un langage de modélisation des systèmes qui est dérivé du langage de modélisation des logiciels « Unified Modeling Language (UML) ». Un diagramme d'exigences SysML (Fig. 2.12) est à la frontière des descriptions formelles. En effet, les exigences sont textuelles mais structurées au sein de diagrammes inter-connectés via des relations typées. On peut par exemple créer une relation de dérivation entre deux exigences pour formaliser que l'une dérive de l'autre.

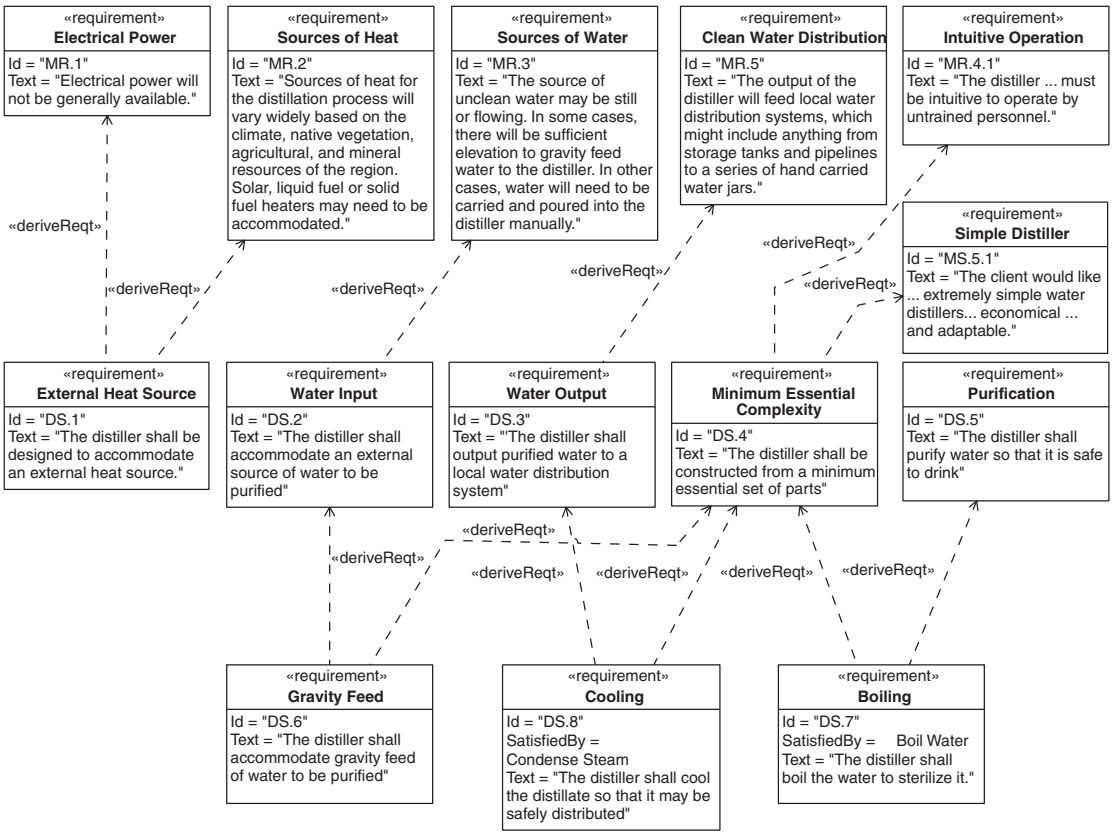


FIGURE 2.12 – Diagramme d'exigences SysML [Friedenthal et al., 2011].

Bien qu'elles soient originellement académiques, certaines de ces approches ont été mises à l'épreuve sur des projets industriels. Les PBRs, par exemple, ont été utilisées à travers la méthode d'ingénierie système Property Model Methodology (PMM) sur plusieurs projets, notamment au Centre National d'Études Spatiales (CNES) [Poupart et al., 2016] et chez Airbus Helicopters [Micouin et al., 2016]. Sans méthodologie associée, le langage SysML trouve aussi sa place chez les plus gros fournisseurs de systèmes aéronautiques [Bernard, 2012], spatiaux [Rabelo and Clark, 2015] ou automobiles [Andrianarison and Piques, 2010]. Il a même franchi la porte d'entrée des écoles d'ingénieurs, des universités et, plus récemment, des lycées techniques.

Certaines approches (semi-)formelles sont prometteuses, en particulier pour les secteurs d'activités

dans lesquels la sécurité du produit est une dimension critique, comme l'aéronautique et le spatial. À ce jour, elles restent néanmoins beaucoup moins populaires que les représentations textuelles, et ce, pour diverses raisons. D'une part, elles sont réservées à des experts qui maîtrisent les fondamentaux de la modélisation et de la simulation. D'autre part, elles ne sont pas assez matures pour satisfaire les besoins industriels tels que le travail collaboratif, l'échange des informations, la gestion des modifications, etc.

Alors que les approches (semi-)formelles visent à réduire la croissance chaotique du nombre d'exigences au moyen d'une ingénierie rationnelle et rigoureuse, une deuxième voie de recherche adopte la démarche inverse en essayant de mettre de l'ordre dans le chaos du langage naturel et des données non structurées au moyen des sciences des données.

Les solutions basées sur la fouille de données

Les informaticiens se sont beaucoup investis dans le développement d'outils qui supportent l'ingénierie des exigences. Pour cela, certains ont utilisé les logiques formelles vues précédemment, tandis que d'autres ont préféré exploiter un vaste corps de connaissances qui permet l'interaction entre l'homme et la machine : les sciences des données.

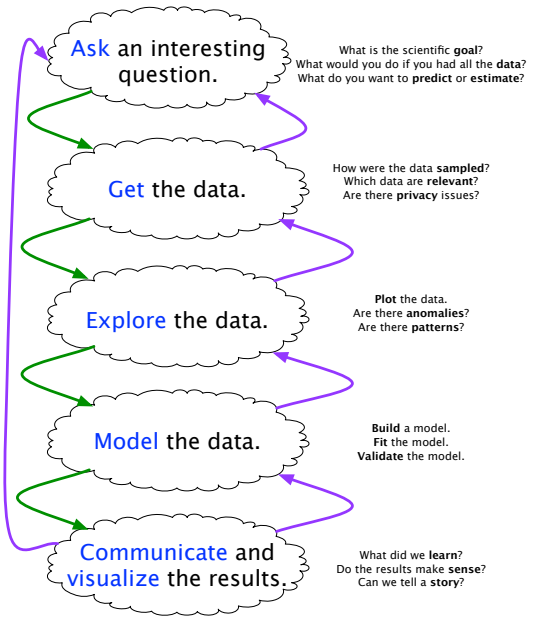


FIGURE 2.13 – Processus d'analyse des données (Extrait du cours CS109 Data Science de Hanspeter Pfister et Joe Blitzstein à Harvard).

Dans la méta-discipline correspondant aux sciences des données (Fig. 2.14), d'un côté, il y a les disciplines qui sont propres à la machine – e.g. traitement du langage naturel, apprentissage, fouille, etc. – avec laquelle on essaye d'automatiser des activités humaines. De l'autre côté, il y a les disciplines qui sont propres à l'homme – e.g. sciences cognitives, perception, décision etc. –, et l'interaction entre l'homme et la machine se fait principalement via des visualisations interactives. Ainsi, dans les sciences des données, nous incluons toutes les théories et technologies utiles à la collection, à l'exploration, à la modélisation et à la visualisation de données (Fig. 2.13). La statistique, l'apprentissage machine, la visualisation de données, la fouille de données, la fouille de textes, la recherche d'informations, le traitement du langage naturel, les ontologies, la théorie des graphes, le Web sémantique, etc. appartiennent à cette nouvelle, ou plutôt « ré-érotisée », méta-science. Les sciences des données ont profité à diverses activités de l'ingénierie des exigences.

Le traitement du langage naturel [Manning et al., 2008] consiste à utiliser des ressources informatiques pour comprendre un énoncé écrit ou oral, ou, inversement, pour exprimer une connaissance sous forme écrite ou orale. Les problèmes traités par le traitement du langage naturel sont de deux niveaux.

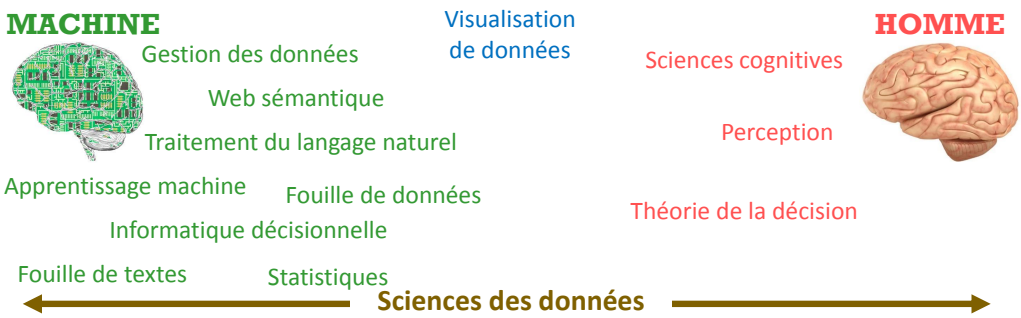


FIGURE 2.14 – Les sciences des données, une méta-science (Adapté du cours CS109 Data Science de Hanspeter Pfister et Joe Blitzstein à Harvard)

.

D'une part, il y a ceux qui sont qualifiés de difficiles car ils tendent à égaler les capacités de l'homme : comprendre un texte, produire un texte, participer à une discussion en temps réel, ou traduire un texte. D'autre part, il y a les problèmes dont la résolution relève plus du calcul mécanique que de l'intelligence : la recherche dans des gros corpus de documents, la correction de fautes grammaticales, ou la génération automatique de rapports. D'un point de vue historique, le traitement du langage naturel a été approché selon deux grandes approches. La première est formelle, symbolique, purement déterministe. Elle repose sur les théories de l'algèbre et des ensembles. C'est la représentation de grammaires à l'aide de machines à états finis [Chomsky, 1956]. La seconde approche est stochastique. Elle repose sur la théorie des probabilités. C'est le canal de communication de Shannon [1948]. La première a tendance à négliger le sens des mots et des phrases, tandis que la seconde raisonne au niveau sémantique pour interpréter et traduire le langage en tenant compte des variations volontaires ou non de la langue. Comme nous le verrons en détail dans les chapitres 4.2 et 5.2, les techniques de traitement du langage naturel ont respectivement servies à extraire les exigences [Kang and Saint-Dizier, 2013, Coatanéa et al., 2013, Bernard et al., 2014, Zeni et al., 2015] et à analyser leur qualité [Mich, 1996, Osborne and MacNish, 1996, Wilson et al., 1997, Mitch and Garigliano, 2002, Kof, 2004, Lami, 2005, Gervasi and Zowghi, 2005, Kasser, 2006b, Szczepaniak and Defarge, 2006, Kiyavitskaya et al., 2008, Lamar, 2009, Moser et al., 2011, Yang et al., 2011, Christophe et al., 2011, Rago et al., 2012, Ott, 2012, Génova et al., 2013, Coatanéa et al., 2013, Arora et al., 2013b, Carlson and Laplante, 2013, Thitisathienkul and Prompoon, 2015].

Pour analyser les exigences, les sciences des données fournissent les moyens de construire des ontologies, c'est-à-dire des structures (semi-)formelles de connaissances, exploitables lors du traitement informatique des données. En ingénierie des exigences, les ontologies ont des applications diverses [Dermeval et al., 2015]. Elles ont notamment servi à analyser la qualité des exigences [körner and Brumm, 2009, Fraga et al., 2015]. Plutôt que de consommer des ontologies, d'autres auteurs cherchent à les générer automatiquement à partir d'un ensemble d'exigences [Kof, 2004, 2005]. Enfin, les ontologies sont des entrepôts de connaissances qui comblent le manque d'expertise dans des domaines spécifiques comme la sureté des produits [Souag et al., 2012]. Le formalisme sous-jacent est généralement issu du Web sémantique, lequel est aussi appelé Web des données. En effet, les ontologies se matérialisent sous la forme d'un ou plusieurs graphes modélisés à l'aide d'un langage de triplets tel que RDF, RDFS ou OWL (Fig. 2.15). Toutes ces représentations dérivées du langage XML facilitent l'analyse automatique des données en général et des exigences en particulier.

Le nombre d'exigences ayant une fâcheuse tendance à croître exponentiellement, divers académiques ont alors exploré la mise en œuvre des techniques de fouille de données, en particulier de données textuelles (*text mining*, en anglais) [Feldman and Sanger, 2006]. Les techniques de fouille de textes sont généralement précédées d'opérations de traitement du langage naturel afin de préparer les données. Une fois pré-traités, les textes sont fouillés pour diverses raisons. Les opérations les plus populaires sont la classification et le découpage en communautés.

⟨ :Dupond :Leads :CSDept ⟩
⟨ :Dupond :TeachesIn :UE111 ⟩
⟨ :Dupond :TeachesTo :Pierre ⟩
⟨ :Pierre :EnrolledIn :CSDept ⟩
⟨ :Pierre :RegisteredTo :UE111 ⟩
⟨ :UE111 :OfferedBy :CSDept ⟩

Subject	Predicate	Object
:Dupond	:Leads	:CSDept
:Dupond	:TeachesIn	:UE111
:Dupond	:TeachesTo	:Pierre
:Pierre	:EnrolledIn	:CSDept
:Pierre	:RegisteredTo	:UE111
:UE111	:OfferedBy	:CSDept

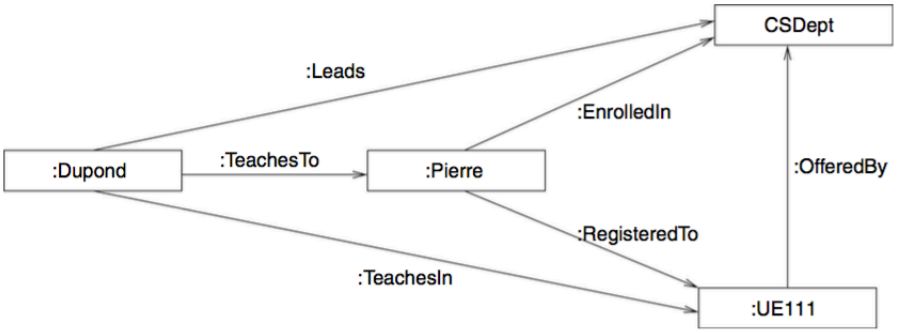


FIGURE 2.15 – Exemple de triplets RDF (en haut) représentés sous la forme d'un graphe (en bas) [Abiteboul et al., 2011].

La classification permet de catégoriser automatiquement des textes au sein de catégories pré-définies. Par exemple, les articles d'un blog peuvent être classifiés automatiquement selon une liste de sujets : politique, économie, sciences et technologies, sport, culture. Le modèle de classification automatique peut être appris à partir d'un gros volume d'exemples, on parle alors d'apprentissage machine ou d'apprentissage statistique. Il peut aussi être manuellement construit via l'implémentation de règles de connaissances telles que : si le texte contient le mot *football* alors il appartient à la catégorie *sport*. Les règles peuvent bien évidement être beaucoup plus sophistiquées. La classification automatique des exigences est dédiée à des tâches très spécifiques. Par exemple, il est communément admis que les exigences dites non fonctionnelles (sécurité, fiabilité, maintenance, etc.) impactent sur les choix d'architecture d'un système. Ainsi, pour distinguer les exigences fonctionnelles des non fonctionnelles, divers auteurs ont exploité les techniques de classification basées sur l'apprentissage machine [Hussain et al., 2008, Casamayor et al., 2010, 2012, Anish et al., 2015, Mahmoud, 2015], tandis que d'autres ont défini des règles de connaissances [Lamar, 2009]. Les méthodes de classification automatique ont aussi servi à classifier les exigences selon un ensemble pré-défini de sujets (température, test, voltage, etc.) [Ko et al., 2007, Knauss, 2011, Ott, 2013, Knauss and Ott, 2014] (Fig. 2.16).

Le découpage en communautés, ou segmentation (*clustering*, en anglais), permet de découper un ensemble de données en communautés (*clusters*, en anglais). C'est une approche exploratoire qui, à partir d'un ensemble de textes et d'une métrique de similarité, facilite l'identification de groupes de données appelées communautés. Les textes d'une communauté sont alors à la fois très similaires entre eux et très différents de ceux qui appartiennent aux autres communautés. À la différence de la classification, la segmentation ne nécessite pas de définir un ensemble de catégories à l'avance. Au contraire, les communautés identifiées sont étiquetées *a posteriori*. Comme la classification, le découpage en communautés a été utilisé pour divers objectifs en ingénierie des exigences [Duan, 2008]. Ces techniques ont servi à regrouper les exigences par sujets [Mokammel et al., 2015], à les prioriser [Achimugu et al., 2014b], à les tracer [Cleland-Huang et al., 2005, Duan and Cleland-Huang, 2007a] et à les trier [Laurent et al., 2007, Duan et al., 2009]. Elles ont aussi facilité la sélection de composants logiciels [Khan and Mahmood, 2012], la détection d'inquiétudes et de dangers transversaux en phase amont [Duan and Cleland-Huang, 2007b], la structuration de spécifications textuelles [Ferrari et al., 2013], la visualisation de communautés d'exigences [Reddivari et al., 2012, Niu et al., 2013, Reddivari et al., 2014], et l'identification de thèmes dans des régulations [Sannier and Baudry, 2012, Sannier, 2013, Sannier and Baudry, 2014].

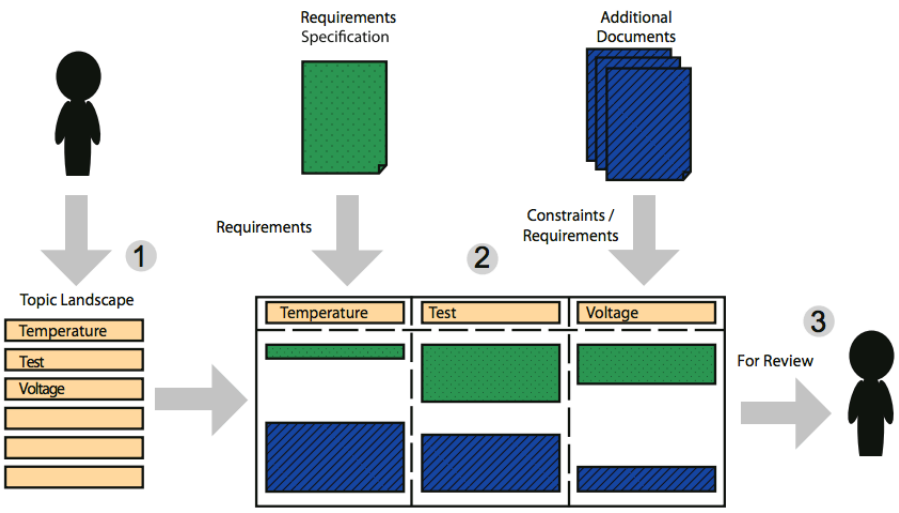


FIGURE 2.16 – Classification d'exigences dans des catégories pré-définies [Ott, 2013].

Qu'ils soient basés sur des règles déclaratives issues d'une ingénierie des connaissances ou de prédictions statistiques, ces algorithmes d'apprentissage automatique ont aussi permis la naissance des moteurs de recommandation. Les moteurs de recommandation sont des algorithmes qui recommandent quelque chose à quelqu'un via la prédiction d'une fonction de préférence. Ainsi, Amazon ou Netflix recommandent un livre ou un film à un utilisateur en fonction de l'historique de ses achats personnels et de ceux réalisés par des clients au profil similaire. Diverses initiatives ont permis d'intégrer les systèmes de recommandation au sein du processus d'ingénierie des exigences [Mobasher and Cleland-Huang, 2011, Felfernig et al., 2013]. Par exemple, ils supportent l'identification et la priorisation des parties prenantes [Lim et al., 2010] et des exigences [Castro-Herrera et al., 2008, Felfernig et al., 2010, Lim and Finkelstein, 2012, Hariri et al., 2014].

Synthèse de l'état de l'art académique

Ni l'état de l'art industriel, ni l'état de l'art technologique n'ont permis d'identifier de solution pour résoudre le problème de la prolifération des exigences. La revue de la littérature académique, elle, révèle que de nombreux résultats sont potentiellement utiles pour supporter la synthèse d'un gros volume d'exigences.

D'une part, il y a les approches (semi-)formelles. Ces dernières traitent le cœur du problème en s'attaquant aux causes qui font que l'ingénierie des exigences n'a parfois plus rien d'une activité d'ingénierie. Ces approches peuvent satisfaire certains secteurs d'activités, en particulier ceux pour qui la modélisation et la simulation sont des tâches familières, mais elle ne conviennent pas à toutes les industries dont la diversité des acteurs ne se limite pas à la fonction d'ingénieur. Il est par ailleurs malheureusement impossible qu'un sous-traitant puisse imposer une méthode et des descriptions (semi-)formelles à un donneur d'ordres au motif que celles-ci permettent de spécifier rigoureusement au juste besoin. C'est pour ces principales raisons que les approches (semi-)formelles ne peuvent pas directement satisfaire à notre mission de synthèse des exigences

D'autre part, il y a les approches issues de la fouille des données qui tentent de soigner les symptômes de la prolifération des exigences. De nombreux travaux de recherche ont démontré que les sciences des données peuvent supporter des tâches bien particulières – extraire, segmenter, tracer, visualiser, etc. – de l'ingénierie des exigences. Dans la suite de ce manuscrit de thèse, comme l'illustre la figure 2.17, nous proposons de s'appuyer sur ces briques théoriques et technologiques qui non seulement nécessitent d'être individuellement enrichies, mais aussi intégrées dans un cadre unique via une méthode, c'est-à-dire un ensemble d'activités dont l'organisation logique doit constituer un moyen de réaliser la mission de synthèse des exigences.

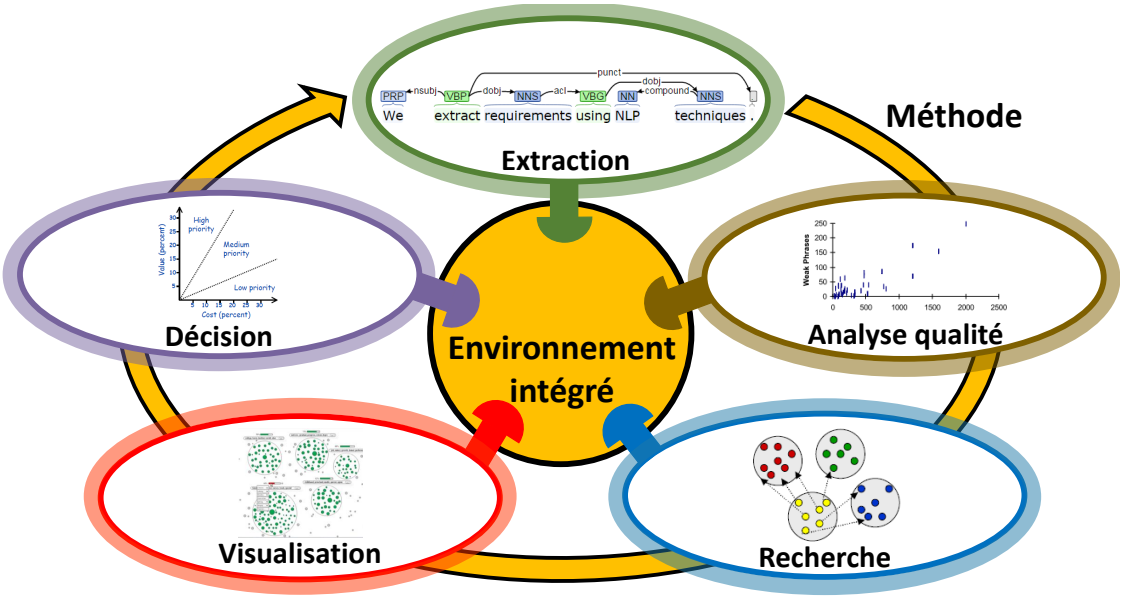


FIGURE 2.17 – Intégration méthodique de briques théoriques et technologiques dans un environnement numérique pour supporter la synthèse d'exigences.

2.5 Synthèse

Cet état de l'art générique confirme qu'aucune solution commerciale ou académique ne répond au problème récurrent rencontré par les sous-traitants qui héritent de gros volumes d'exigences.

Par ailleurs, ce problème, dont l'ampleur ne cesse de croître avec le temps, n'est pas évoqué par les normes et les guides de bonnes pratiques en ingénierie des exigences. En effet, un sous-traitant peut appliquer toutes les recommandations prescrites, cela ne l'aidera pas à faire face à l'amas d'exigences qui lui est confié.

Les technologies commerciales d'ingénierie des exigences n'apportent pas non plus de solution. Parmi les éditeurs de logiciels originellement dédiés à la conception et à la fabrication de produits manufacturés, certains d'entre eux se sont récemment inquiétés de la nécessité de fournir un environnement numérique intégré permettant de développer et de gérer des exigences conjointement avec les solutions de conception. Ces outils sont particulièrement utiles aux donneurs d'ordres qui développent et gèrent les exigences, mais elles ne facilitent pas le travail des sous-traitants qui croulent sous un amas d'exigences.

Les travaux de recherche académiques, eux, contiennent peut-être une partie de la réponse dans des propositions diverses et variées. Une première catégorie de contributions incitent les entreprises à spécifier leurs produits dans un cadre formel dont la rigueur limite le nombre d'exigences. Bien que les solutions de ce genre soient indéniablement efficaces, elles ne peuvent être pratiquées que par une minorité d'initiés. Par ailleurs, un fournisseur n'a aucun moyen d'imposer un tel formalise à un client, mais doit se contenter d'un gros volume de données désordonnées dans lequel il doit faire son propre tri. C'est dans cette seconde direction que la deuxième partie des travaux de recherche se concentrent. Plutôt que de s'astreindre à une ingénierie formelle et rigoureuse des exigences, les techniques des sciences des données sont exploitées pour fouiller dans de gros volumes d'exigences. Cette approche semble mieux convenir au problème des sous-traitants tributaires des pratiques des donneurs d'ordres ; cependant, aucune proposition ne s'intéresse à la synthèse des exigences. En effet, les techniques de fouilles de données servent principalement à extraire, analyser, rechercher, tracer, prioriser, ou visualiser des exigences, mais ne visent pas à en faire la synthèse. Néanmoins, certaines d'entre elles sont utiles pour résoudre notre problématique. Par exemple, pour synthétiser les exigences, il est sûrement nécessaire de les extraire, de les analyser et de les visualiser. Il est donc évident que certaines briques théoriques et technologiques ont leur place dans notre proposition, mais nécessitent d'être non seulement adaptées et améliorées, mais aussi articulées logiquement via une intégration méthodologique.

Deuxième partie

PROPOSITION

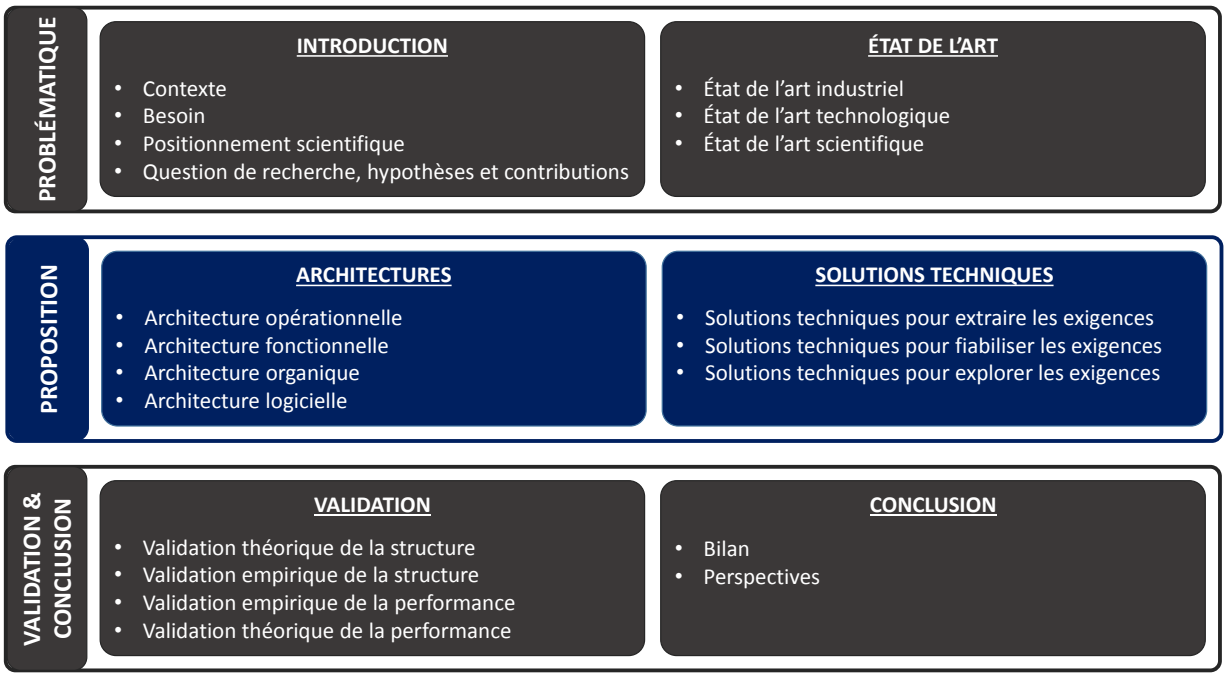


FIGURE 2.18 – 2ème partie dédiée à la proposition d'une solution de synthèse des exigences.

Après l'introduction et l'état de l'art général de la partie I de ce manuscrit, cette partie II est un exposé de notre proposition. La méthode outillée de synthèse des exigences que nous proposons y est présentée en deux temps.

Premièrement, dans le chapitre 3, nous donnons un aperçu général de la proposition selon quatre perspectives : opérationnelle, fonctionnelle, organique et logicielle. C'est la définition logique de ces quatre perspectives qui nous amène progressivement du besoin vers une solution.

Deuxièmement, les détails des trois grandes fonctions de service que l'environnement numérique doit fournir aux utilisateurs sont présentés. Chaque fonction de service – extraction, fiabilisation et exploration – est un chapitre – 4, 5 et 6 – dont la structure est systématique : introduction, état de l'art, proposition, expérimentation et synthèse.

Chapitre 3

ARCHITECTURES

La définition d'une solution nécessite trois perspectives conceptuelles : opérationnelle, fonctionnelle et organique [AFIS, 2012b].

La première perspective présentée en section 3.1 est la **définition opérationnelle** dans laquelle l'environnement numérique est vu comme une « boîte noire » en relation avec les éléments de son environnement. C'est une description des fonctions de service que l'environnement numérique doit fournir aux utilisateurs pour accomplir la mission de synthèse des exigences. La section répond à la question : Qu'est-ce que l'utilisateur doit être capable de faire avec l'environnement numérique ?

La deuxième perspective présentée en section 3.2 est la **définition fonctionnelle** de la solution. C'est une décomposition structurée (analyse fonctionnelle interne - « boîtes blanches ») décrivant l'agencement des fonctions techniques dont l'intégration reconstitue les fonctions de service. La section répond à la question : Qu'est-ce que l'environnement numérique doit être capable de faire pour rendre les services décrits dans l'architecture opérationnelle ?

La troisième perspective présentée en section 3.3 est la **définition organique** de la solution. Selon le produit, les organes sont physiques, logiciels ou hybrides. Ici, ils sont tous logiciels. Comme l'architecture fonctionnelle, l'architecture organique est une décomposition structurée décrivant l'agencement des organes logiciels et leurs liens d'interface. La section répond à la question : Comment est-ce que l'environnement numérique réalise les fonctions techniques décrites dans l'architecture fonctionnelle ?

Enfin, pour conclure, l'architecture logicielle du prototype de l'application est décrite en section 3.4.

Ces architectures ne sont pas totalement exhaustives. On pourrait, par exemple, décomposer les fonctions techniques et les organes logiciels. Cependant, les simplifications sont volontaires pour faciliter la lisibilité des descriptions [1], et suffisantes pour communiquer les détails de notre proposition selon différentes perspectives.

3.1 Architecture opérationnelle

Pour accomplir sa mission de synthèse des exigences, l'environnement numérique doit fournir trois fonctions de service : extraire, fiabiliser et explorer. Avant de détailler chacun de ces services, il est nécessaire d'introduire les parties prenantes qui en bénéficient.

Les parties prenantes : un jeu de rôle à quatre joueurs.

À l'image du diagramme de contexte de la figure 3.1, quatre acteurs – administrateur, manageur, analyste, expert – interagissent avec l'environnement numérique.

Comme son nom l'indique, le rôle de l'**Administrateur** est d'administrer l'application. Il a la charge de créer, modifier et supprimer un projet. Ici, la notion de projet est équivalente à celle que l'on retrouve dans l'industrie : « un projet est un effort temporaire entrepris pour réaliser un produit ou un service avec

1. Comme l'eût écrit Paul Valéry dans Œuvres II en 1942 : « Ce qui est simple est faux, ce qui est compliqué est inutilisable. ».

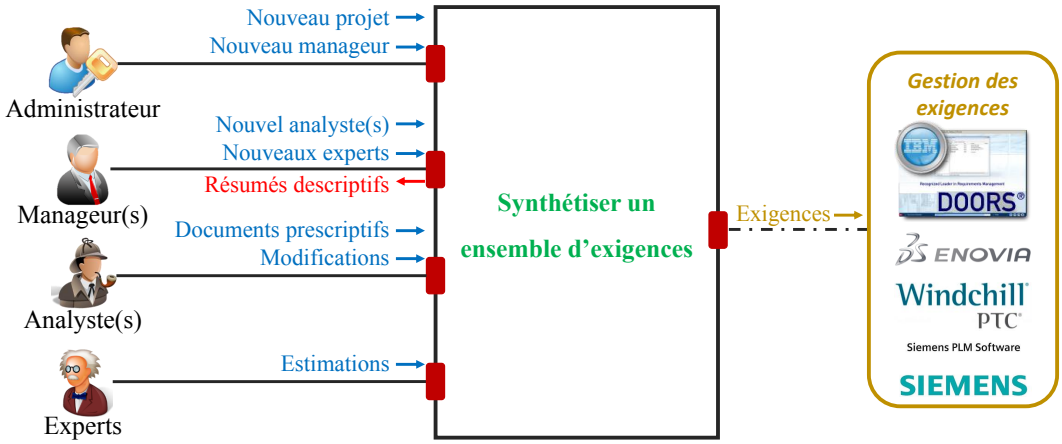


FIGURE 3.1 – Diagramme de contexte décrivant les interactions entre l'environnement numérique et l'environnement extérieur.

des ressources précises et des exigences » [ISO/IEC/IEEE 15288, 2015]. pour chaque nouveau projet, l'Administrateur créé ou sélectionne un Manageur de projet.

Un **Manageur** gère un ou plusieurs projets simultanément. L'activité de gestion d'un projet est double. D'une part, elle consiste à créer, modifier ou supprimer un ou plusieurs Analystes et Experts affectés à un ou plusieurs projets. D'autre part, le Manageur, doté d'une fonction décisionnelle au sein de l'entreprise, a la responsabilité de prendre des décisions stratégiques informées.

Un projet peut avoir un ou plusieurs **Analystes** suivant le contexte. Par exemple, si plusieurs employés sont familiers avec les bonnes pratiques de l'ingénierie des exigences, ou si le nombre d'exigences à analyser est important, alors plusieurs Analystes peuvent être actifs dans un projet. Le rôle principal d'un Analyste est de fiabiliser l'interprétation des exigences afin que les Experts puissent *a-posteriori* réaliser des estimations objectives des critères d'aide à la décision.

Les **Experts**, eux, doivent être multiples. Chacune des fonctions de l'entreprise doit être représentée par au moins un Expert. La mission d'un Expert est d'exploiter ses connaissances tacites afin d'estimer des critères d'aide à la décision associés aux exigences. C'est à partir de ces estimations que le Manageur du projet sera en capacité de prendre des décisions stratégiques informées.

Enfin, dans un futur proche qui aura peut-être permis de résoudre un certain nombre de limites actuelles, l'environnement numérique pourrait être utile à la construction d'un référentiel d'exigences géré en configuration par un outil de gestion des exigences. Le cas échéant, il faudra assurer l'interopérabilité entre les deux technologies. La norme d'échange des exigences *Requirements Interchange Format (ReqIF)* définie par l'OMG est une solution satisfaisante.

Synthétiser c'est : extraire, fiabiliser, et explorer

le diagramme de séquence de la figure 3.2 détaille l'enchainement des quatre fonctions de service permettant de remplir la mission de synthèse des exigences.

FS1 - Extraire les exigences. L'environnement numérique doit permettre à un analyste d'extraire les exigences spécifiées dans des documents prescriptifs non-structurés ou semi-structurés. L'extraction automatique des exigences doit éviter aux analystes de lire des centaines ou des milliers de pages pour identifier les exigences.

FS2 - Fiabiliser les exigences. L'environnement numérique doit permettre à un analyste de minimiser l'ambiguïté d'une exigence. Si le champ des interprétations est réduit en amont du processus de synthèse, alors les estimations réalisées par les experts et les décisions stratégiques prises par le manageur seront respectivement objectives et rationnelles.

FS3 - Explorer les exigences. L'environnement numérique doit permettre au mangeur d'analyser une synthèse de l'espace prescriptif engendré par les exigences afin qu'il puisse prendre des décisions informées.

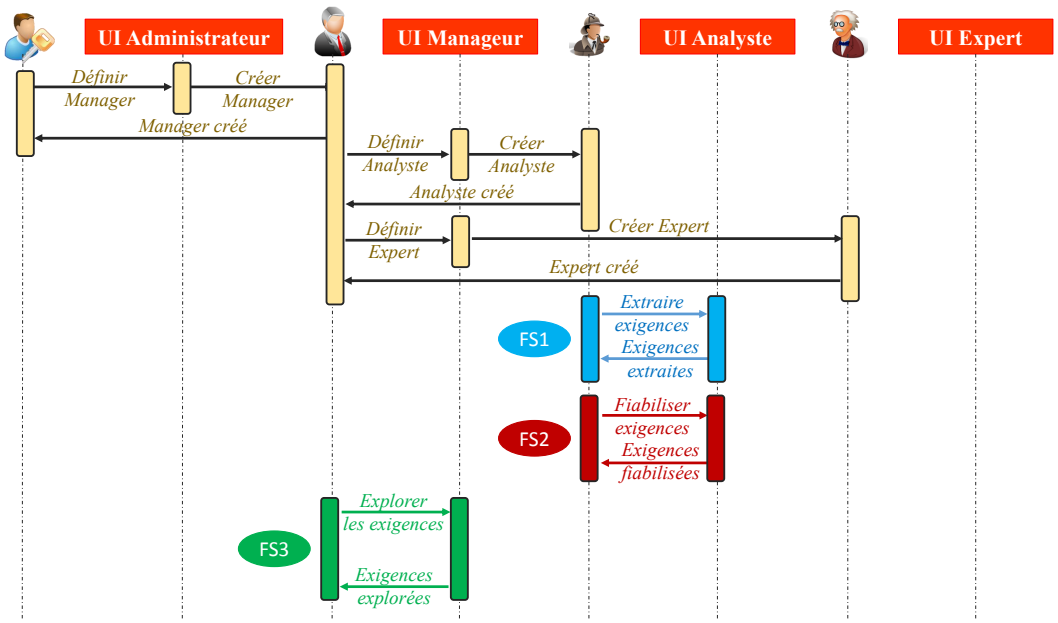


FIGURE 3.2 – Scénario d'utilisation de l'environnement numérique pour synthétiser des exigences.

3.2 Architecture fonctionnelle

Chacune des fonctions de service identifiées en section 3.1 est simplifiable via une décomposition en fonctions techniques. La figure 3.3 est un bloc-diagramme de flux fonctionnels décrivant la séquence logique des fonctions techniques (13 blocs élémentaires) dont l'intégration doit permettre de réaliser chacune des fonctions de service (3 blocs intermédiaires) qui, eux-mêmes, nécessitent d'être intégrés pour remplir la mission de synthèse des exigences (bloc racine).

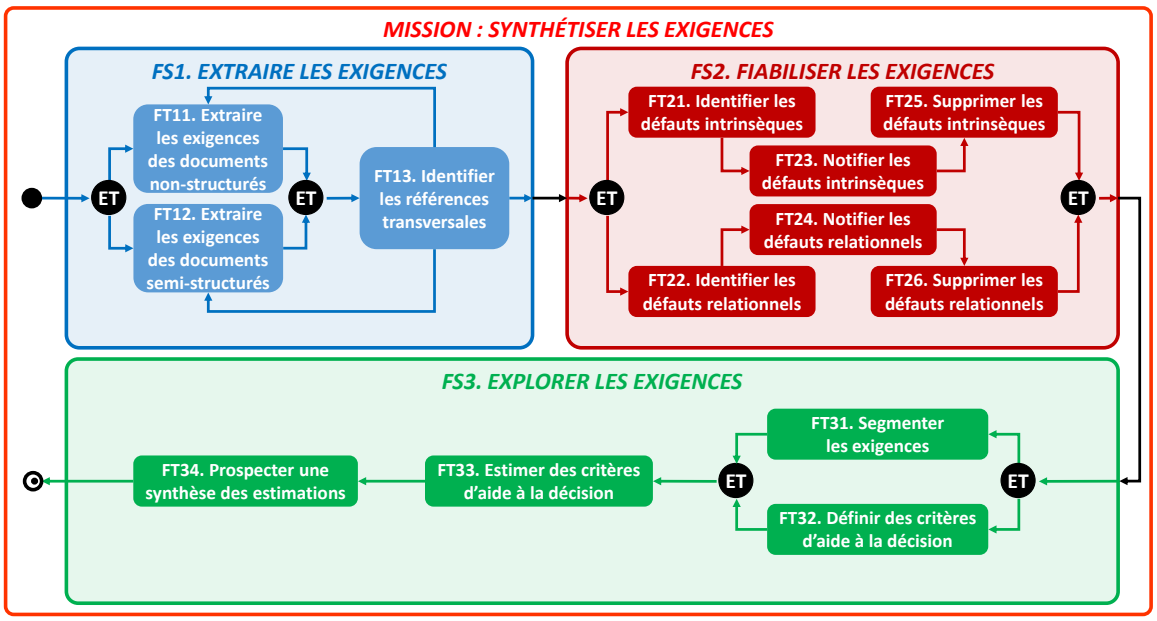


FIGURE 3.3 – Architecture fonctionnelle de l'environnement numérique.

Les données sont habituellement qualifiées comme non structurées, semi-structurées ou structurées. Les données structurées sont stockées dans une base de données, relationnelle en général, qui distingue la structure de données (le schéma) des données elles-mêmes (l'instance). On dit que « l'instance est

conforme au schéma ». Les données structurées que l'on désigne aussi sous le nom de données tabulaires sont des matrices dont chacune des lignes est une donnée et chacune des colonnes est un attribut. Les outils de gestion des exigences comme IBR Rational Doors et Enovia V6 Requirements stockent des spécifications structurées.

Les données non structurées, elles, sont des séquences de caractères sans structure sous-jacente qui ne peuvent pas être directement stockées dans des tableaux ou des graphes. Les spécifications qui sont des documents Word ou PDF sont non structurées.

Enfin, les données semi-structurées sont plus structurées qu'une séquence brute de caractères, mais moins qu'un tableau. Dans la plupart des cas, elles prennent la forme d'un graphe acyclique (arbre) – les fichiers XML, par exemple – ou d'un graphe cyclique – les fichiers XMI, par exemple. Une spécification qui est conforme à la structure de données définie par la norme d'interopérabilité *Requirements Interchange Format (ReqIF)* [Object Modeling Group, 2016] est semi-structurée.

Les fonctions techniques pour extraire les exigences

L'extraction des exigences se divise en trois fonctions techniques. D'une part, l'environnement numérique doit extraire les exigences spécifiées dans des documents prescriptifs non structurés (FT11) et semi-structurés (FT12). D'autre part, en parcourant la liste des exigences extraites, l'outil doit identifier celles qui contiennent des références transversales (FT13) vers des documents externes applicables qui, eux aussi, peuvent prescrire des exigences. L'analyste peut alors soumettre au module d'extraction les documents référencés applicables.

Les fonctions techniques pour fiabiliser les exigences

Après avoir extrait les exigences, l'environnement numérique doit successivement identifier et notifier les défauts de qualité, lesquels sont soit intrinsèques à une exigence (FT21 et FT23), soit relationnels entre plusieurs exigences (FT22 et FT24). L'analyste peut alors intervenir manuellement pour fiabiliser l'interprétation des exigences en supprimant tout ou partie des défauts (FT25 et FT26).

Les fonctions techniques pour explorer les exigences

Enfin, dans un troisième temps, l'environnement numérique doit permettre au manageur du projet d'explorer l'espace prescriptif pour prendre des décisions informées. Pour cela, le manageur doit commencer par segmenter les exigences en communautés (FT31) afin d'assurer le passage à l'échelle sur un gros volume d'exigences. En parallèle, le manageur doit définir les critères d'aide à la décision (FT32) qui sont associés à chacune des communautés d'exigences. Les experts doivent alors intervenir pour estimer les critères d'aide à la décision (FT33). Pour terminer, le manageur prospecte l'espace prescriptif engendré par les exigences (FT34) en vue d'y découvrir des opportunités et de prendre des décisions informées.

3.3 Architecture organique

La figure 3.4 illustre l'architecture organique de l'environnement numérique proposé. Une architecture organique, ou physique, est un ensemble de blocs constitutifs, des solutions de conceptions, implémentant les fonctions techniques introduites précédemment. La projection fonctionnelle des fonctions techniques sur les organes logiciels sont identifiables grâce au code couleurs qui est similaire à celui de l'architecture fonctionnelle en section 3.2. La projection fonctionnelle n'est généralement pas biunivoque. En effet, plusieurs fonctions techniques peuvent être assurées par un seul organe logiciel, ou, inversement, une seule fonction technique peut être assurée par plusieurs organes logiciels.

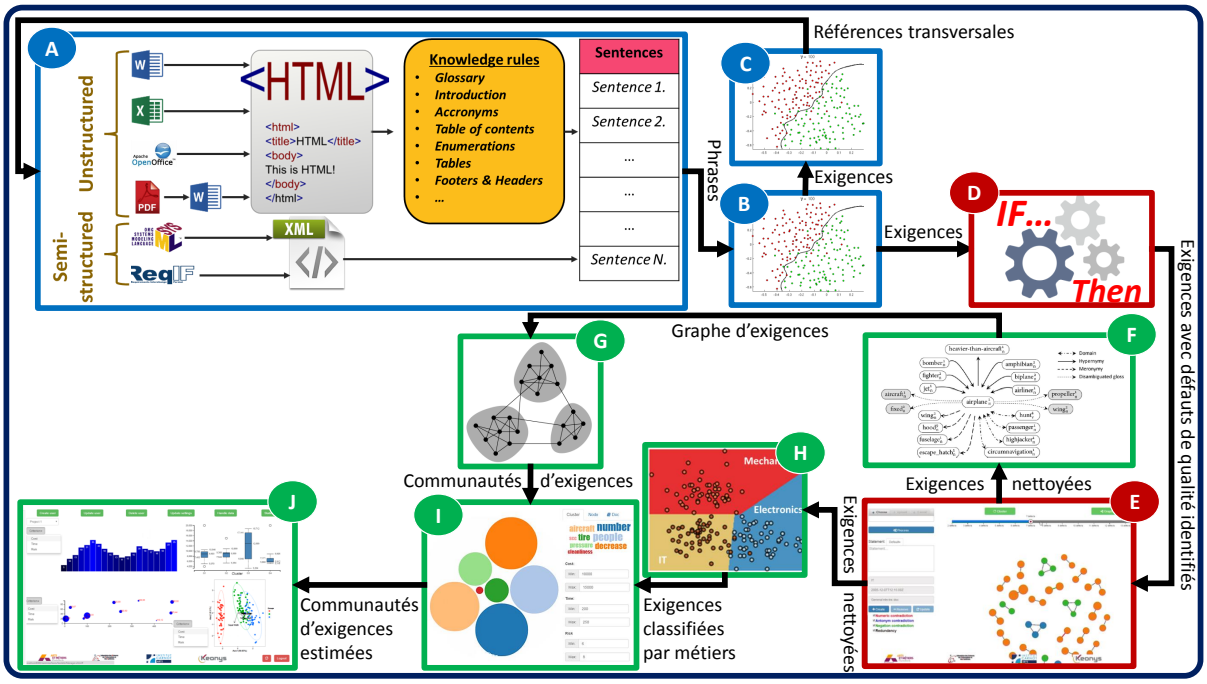


FIGURE 3.4 – Architecture organique de l'environnement numérique.

Les organes logiciels pour explorer les exigences

L'extraction des exigences est assurée par trois composants. Un chaînage numérique constitué d'extracteurs de textes et d'opérations de traitement du langage naturel (A) extrait les phrases. Les phrases passent ensuite dans un modèle de classification à base de règles (B) pour distinguer les énoncés prescriptifs des énoncés informatifs. Une fois extraites, les exigences sont envoyées dans un second modèle de classification basé sur l'apprentissage automatique (C), lequel identifie les exigences contenant des références transversales, afin que l'analyste puisse soumettre d'autres documents prescriptifs applicables au module A.

Les organes logiciels pour fiabiliser les exigences

Une fois que l'analyste a soumis tous les documents prescriptifs, un nouveau processus de traitement du langage naturel et d'analyse sémantique (D) permet de détecter les défauts de qualité intrinsèques à une exigence et relationnels entre plusieurs exigences. Tous les défauts potentiels sont alors notifiés à l'analyste au moyen de représentations graphiques interactives (E).

Les organes logiciels pour explorer les exigences

Toutes les exigences nettoyées de leurs défauts de qualité sont enrichies par une ontologie et un thésaurus (F). Ces deux sources de connaissances sont nécessaires à la construction d'un graphe dont les sommets sont les exigences, et les arêtes sont, d'une part, les relations conceptuelles et sémantiques entre les termes clés de chaque exigence, et, d'autre part, les références transversales préalablement extraites par le module C. Le graphe d'exigences est ensuite segmenté par un algorithme de découpage en communautés (G) à la demande du manageur du projet en cours. Une deuxième représentation graphique interactive (I) permet alors à chacun des experts d'explorer les sous-graphes sous-jacents à chaque communauté. Quand un expert a pris connaissance des exigences d'une communauté, cette même interface sert de point d'entrée pour estimer les critères d'aide à la décision (coût, temps, risques, etc.) qui sont associés à la communauté. Ainsi, une communauté est assimilable à un « mini-projet » contenant un nombre réduit d'exigences, et dont l'estimation est un juste milieu entre deux extrêmes : des centaines

ou des milliers d'exigences à estimer d'un coup ou bien une exigence seule isolée de tout contexte. Enfin, le manageur dispose d'un tableau de bord (J) composé de diverses représentations issues des techniques exploratoires uni, bi et multidimensionnelles de la statistique descriptive. Ces représentations ont deux objectifs. D'une part, elles donnent un aperçu de l'espace prescriptif en synthétisant les estimations réalisées pour chaque communauté d'exigences. D'autre part, elle permet la simulation de scénarios via une analyse « *What if ?* ».

3.4 Architecture logicielle

Tous les organes logiciels présentés dans la section précédente sont implémentés au sein d'un prototype d'application Web client-serveur dont l'architecture multi-couches est illustrée par la figure 3.5.

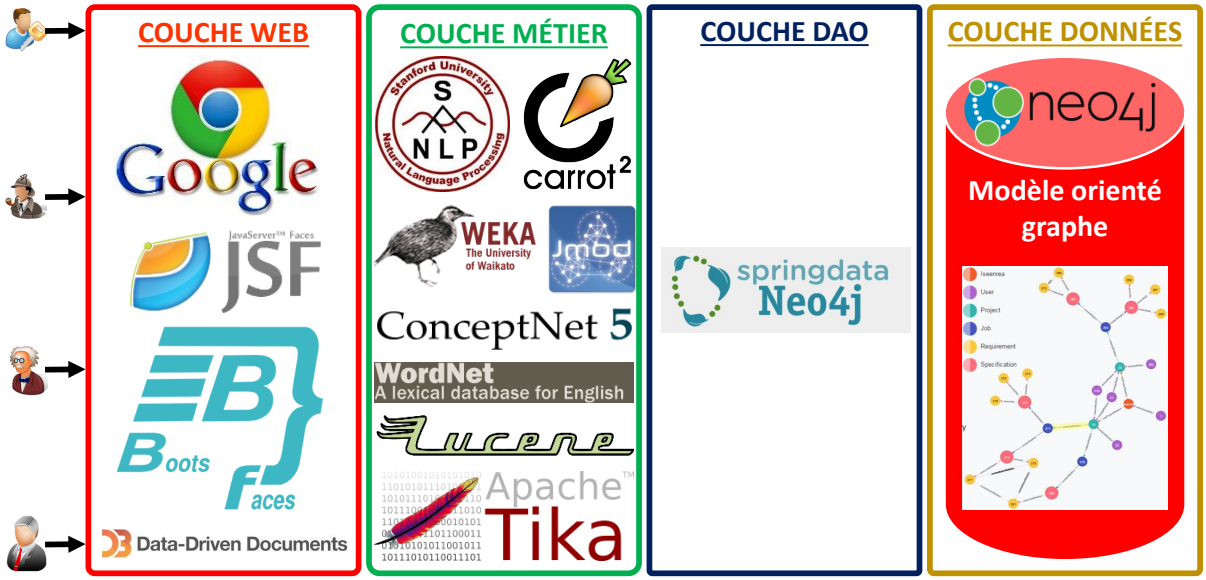


FIGURE 3.5 – Architecture logicielle multi-couches de l'environnement numérique.

Une application client-serveur multi-couches

En lisant l'architecture de gauche à droite on commence par les utilisateurs, la couche Web, la couche métier, la couche d'accès aux données (*Data Access Object (DAO)*) et la couche de données.

La **couche Web** est en contact direct avec les utilisateurs via les pages web d'un navigateur tel que Google Chrome. La définition des pages Web s'appuie sur : le patron de conception Modèle-Vue-Contrôleur (MVC) JavaServer Faces (JSF) 2, ainsi que les bibliothèques JavaScript BootsFaces et Data-Driven Documents (D3.js) pour respectivement construire les pages Web et visualiser les données.

La **couche métier** (*domain layer*, en anglais) correspondant à l'ensemble des algorithmes qui sont propres à notre application, lesquels bénéficient de méthodes issues de bibliothèques externes exploitées au moyen d'interfaces de programmation (*Application Programming Interface (API)*, en anglais). Cette couche utilise des données provenant de l'utilisateur via la couche Web et de la base de données située dans la couche de données via la couche DAO.

La couche **DAO** permet de communiquer avec la couche de données. Pour cela, l'application s'appuie sur le framework Spring Data Neo4J qui met a disposition un ensemble de fonctions pré-existantes – créer, lire, mettre à jour et supprimer, par exemple –, et dont l'appel se fait grâce à l'annotation de classes qui suivent le modèle de programmation Plain Old Java Object (POJO).

Enfin, la **couche de données** n'est ni plus ni moins qu'une base de données. Ici, la base de données NoSQL orientée graphe Neo4J est préférée, car nous faisons l'hypothèse que le graphe est une structure prometteuse pour modéliser les exigences. Une base de données NoSQL orientée graphe à de

nombreux avantages vis-à-vis des technologies SQL. Par exemple, la performance des opérations est considérablement plus importante car elles sont effectuées sur un sous-graphe sans passer par une succession de jointures. Un modèle orienté graphe est aussi beaucoup plus flexible qu'un modèle tabulaire car des sommets, des arêtes, ou des sous-graphes entiers peuvent être ajoutés, modifiés ou supprimés sans impacter les requêtes de fonctionnalités déjà mises en place. Ainsi, une telle solution ne nécessite pas d'avoir un modèle de données bien défini dès le départ, ce qui s'avère être un avantage lorsque les besoins sont encore très volatiles. Ce second argument trouve tout son intérêt en phase de recherche ou de prototypage comme la nôtre.

Un modèle de données orienté graphe

Dans un document de spécification ou dans un outil de gestion des exigences, les énoncés prescriptifs sont présentés linéairement dans un tableau ou dans un texte arborescent. Néanmoins, la structure naturelle d'une spécification est un ensemble d'exigences qui sont liées par des relations explicites – liens de dérivation mère-fille, liens d'implémentation, etc. – ou implicites – liens sémantiques, liens conceptuels, références-transversales, etc. Ces relations sont indispensables pour contextualiser chaque exigence dont la vue holistique facilite la prise de décisions informées. C'est pour cette raison que nous faisons l'hypothèse que la structure mathématique de graphe est l'approche la plus adéquate, sinon idéale, pour modéliser les exigences et les relations qui les unies. Dans ce graphe, les exigences sont des sommets, tandis que les interdépendances sont des arêtes. Il peut y avoir plusieurs relations entre deux exigences, ce n'est donc pas un graphe simple, mais un multi-graphe.

Alors que le modèle entité-association d'une base de données SQL est habituellement défini par un diagramme de classes UML, la base de données NoSQL orientée graphe Neo4J s'appuie sur un graphe de propriétés (*Property Graph Data Model*, en anglais). Un graphe de propriétés est, comme son nom l'indique, une structure mathématique de graphe dont les sommets et les arêtes sont des objets caractérisés par des propriétés (attributs) dont le type de données est primitif tel qu'un booléen, un entier (byte, short, int et long), un flottant, un double, un caractère ou une chaîne de caractères. D'un point de vue pratique, les sommets sont, dans la plupart des cas, des entités identifiées par des noms, tandis que les arêtes sont des verbes. On retrouve ainsi des 3-uplets, des triplets, comme « Travaille (Romain, ENSAM) » ou « Romain – Travaille –> ENSAM », lesquels sont similaires à ceux rencontrés dans les langages servant à modéliser des ontologies tels que RDF, RDFS et OWL [Abiteboul et al., 2011].

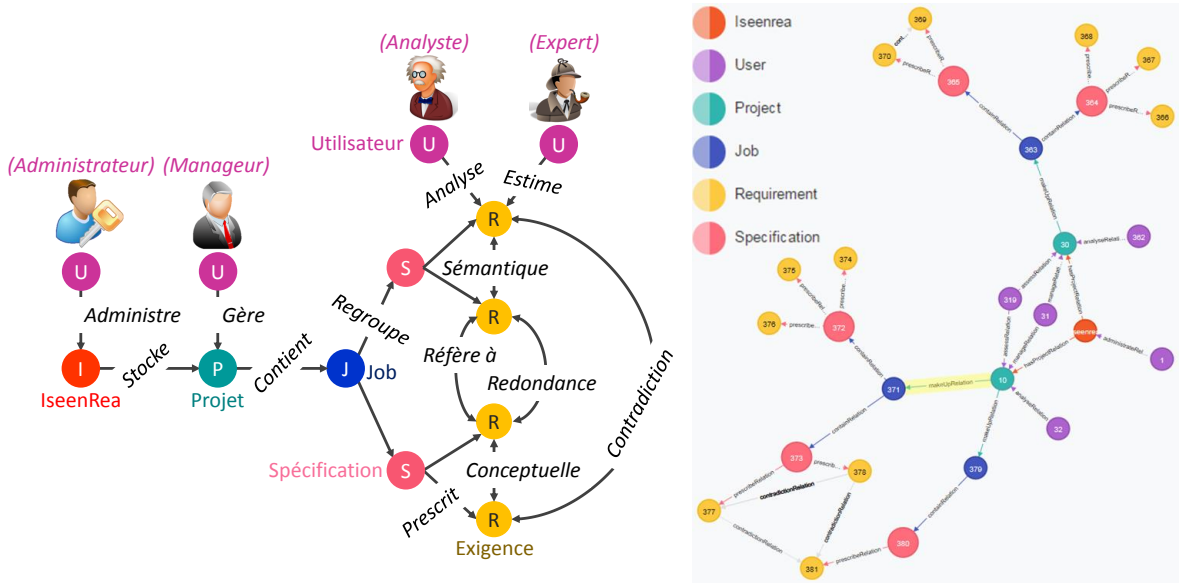


FIGURE 3.6 – Graphe de propriétés (à gauche) et instance (à droite).

La figure 3.6 représente le graphe de propriétés sur lequel repose l'application numérique, ainsi qu'une instance volontairement pauvre. L'application numérique, temporairement dénommée « IseenRea », se matérialise par le singleton de même nom (en orange) qui est administré par l'administrateur (en fuchsia) de l'environnement. Ce sommet racine stocke un ou plusieurs projets (en vert) dont chaque instance est gérée par un manageur (en fuchsia). Notons que tous les utilisateurs sont des sommets de même type « Utilisateur », seul un attribut change. À chaque fois qu'un ou plusieurs documents prescriptifs (en rose) sont soumis à l'application, un nouveau sommet « Job » est instancié. Ces « Jobs » servent à tracer les documents prescriptifs qui peuvent arriver au compte-gouttes. Après avoir été traitées, les exigences extraites sont modélisées par de nouveaux sommets « Exigences » (en jaune) . Une paire d'exigences peut alors être liée par des relations sémantiques ou conceptuelles, mais aussi par des défauts de qualité relationnels tels que les redondances et les contradictions. Enfin, certaines relations matérialisent les relations transversales quand une exigence réfère à un document prescriptif externe.

Somme toute, notre proposition contribue au passage du chaos non structuré quasi-inexploitable vers un modèle orienté graphe exploitable dans lequel les exigences, leurs propriétés, et certaines de leurs interdépendances sont formellement modélisées.

3.5 Synthèse

La mission de synthèse des exigences est, d'un point de vue opérationnel, la conjonction de trois services. Pour un projet donnée, les analystes doivent pouvoir extraire les exigences et fiabiliser leur interprétation, tandis que le manageur du projet doit être à même de prendre des décisions stratégiques informées.

Pour implémenter les fonctions techniques nécessaires à la réalisation des ces services, nous proposons une application Web collaborative et multi-disciplinaire qui s'appuie sur des techniques issues des sciences des données telles que : le traitement du langage naturel, la fouille de textes, la visualisation de données, et la théorie des graphes. Le modèle de données sous-jacent à ces fonctionnalités est un multigraphe de propriétés base de données Neo4J. Neo4j et divers logiciels libres tels que Stanford CoreNLP, Apache Tika ou Weka sont embarqué dans un prototype d'application Web Java.

Cette définition systémique de notre solution sert de colonne vertébrale pour la suite du manuscrit. Chacune des fonctions de service – extraire, fiabiliser et explorer – nécessaires à la synthèse des exigences fait l'objet d'un chapitre. Ces chapitres ont une structure identique : (1) une introduction de l'objectif ; (2) un état de l'art spécifique des solutions existantes ; (3) notre proposition : (4) une vérification de notre proposition ; et (5) une synthèse.

Chapitre 4

EXTRAIRE LES EXIGENCES

4.1 Introduction

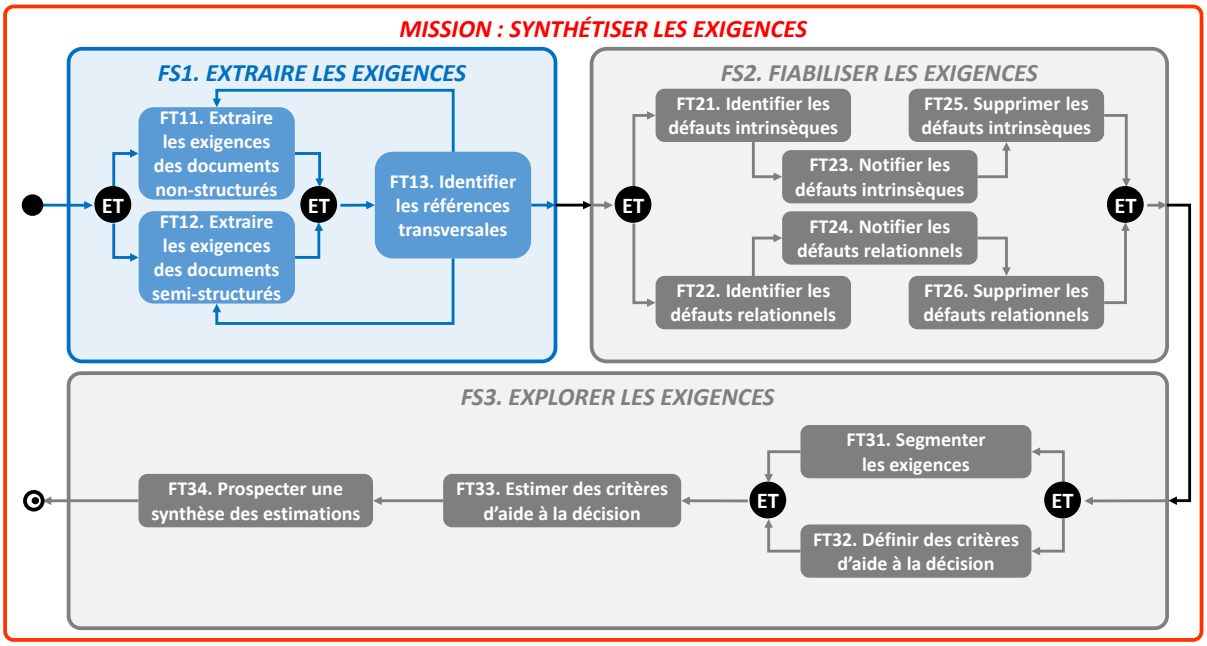


FIGURE 4.1 – FS1 - Extraire les exigences.

Cette section traite la première fonction de service FS1, laquelle doit permettre à un ou plusieurs analystes d'extraire les exigences à partir de documents.

Pour cela, l'outil doit extraire les exigences spécifiées dans des documents non structurés ou semi-structurés. Par ailleurs, l'environnement numérique doit détecter les exigences qui font référence à des documents potentiellement prescriptifs et applicables.

La section 4.2 mettra en évidence les fonctionnalités et les limites des solutions technologiques commerciales, ainsi que celles des prototypes développés à travers des recherches académiques.

La section 4.3 décrira le chaînage numérique qui extrait les exigences et identifie celles qui contiennent des références transversales. Des parseurs et des opérations de traitement du langage naturel sont appliqués pour extraire les exigences, tandis qu'une fonction de classification basée sur l'apprentissage machine identifie les références transversales.

La section 4.4 vérifiera si le chaînage extrait bien les exigences, et si la fonction de classification identifie les exigences contenant des références transversales.

La section 4.5 est une synthèse qui résumera ce qui est nouveau ou différent dans l'approche proposée, ainsi que les limites et les perspectives d'amélioration.

4.2 État de l'art des solutions d'extraction des exigences

Pour commencer cet état de l'art des solutions permettant d'extraire les exigences de documents, nous analysons les capacités des outils d'ingénierie des exigences les plus populaires sur le marché.

Extraire les exigences avec les solutions commerciales

Les solutions commerciales qui outillent le processus d'ingénierie des exigences, en particulier les logiciels de gestion des exigences, proposent des fonctionnalités pour extraire les exigences contenues dans des documents non structurés. Par exemple, quand un document prescriptif est importé dans IBM Rational DOORS, l'utilisateur choisit le niveau de détail – mots clé, phrase, paragraphe ou document – qui permet de créer les objets DOORS correspondants. Si le document Word est importé dans son intégralité, alors un document DOORS jumeau est généré sans que les énoncés prescriptifs et informatifs soient distingués. Les titres Word de niveau 1 à 9 sont importés comme des titres DOORS, tandis que les tableaux et les listes deviennent des tableaux et des listes DOORS. De manière similaire, les outils PLM comme ENOVIA V6 permettent de capturer manuellement la structure d'un document avant de l'importer. Par ailleurs, plutôt que d'importer un document dans sa totalité, l'utilisateur peut sélectionner manuellement les exigences dans leur forme originale (phrase, liste, tableau, etc.) afin de générer des objets jumeaux dans la base de données. Enfin, la figure 4.2 illustre une troisième approche qui consiste à extraire les exigences à partir d'une liste de mots clés préalablement définie par l'utilisateur. Ce dernier doit alors entrer toutes les formes en tenant compte de la flexion. La flexion fait référence aux phénomènes purement grammaticaux (genre, nombre, personne, mode, temps) n'affectant pas la catégorie syntaxique d'un terme. Par exemple, toutes les formes conjuguées des verbes (doit, doivent, devra, désire, désirent, désirera, etc.) doivent être spécifiées par l'analyste au risque que l'extraction soit très incomplète.

Pour résumer, avec les outils existants soit on clone automatiquement la totalité du document dans une base de données, soit il faut réaliser diverses opérations manuelles pour identifier les exigences en détail, car ils ne disposent d'aucune capacité de traitement du langage naturel.

Extraire les exigences grâce aux verbes modaux

Les verbes modaux tels que *shall*, *must* et *should* sont des indices clés pour identifier les exigences [Lash, 2013]. En effet, beaucoup, voire tous les guides de rédaction des exigences préconisent d'utiliser le modal *shall*. Certains proposent aussi de caractériser le niveau de priorité d'une exigence en fonction du verbe modal. C'est le cas de la méthode MoSCoW par exemple [Wiegers and Beatty, 2013]. Les exigences non négociables pour que le projet soit un succès sont alors spécifiées avec le modal *must*. Les exigences importantes mais optionnelles emploient le modal *should*. Les exigences désirables si les contraintes de temps et les ressources le permettent sont rédigées avec le modal *could*. Enfin, les exigences qui ne seront pas implémentées, mais qui pourraient l'être dans une prochaine version du produit sont spécifiées avec le modal *won't*. La plupart des propositions [Sawyer et al., 2002, Coatanéa et al., 2013, Bernard et al., 2014] qui visent à extraire les exigences sont des modèles de classification basés sur la règle : « SI le terme $\mathcal{T}i$ de la phrase $\mathcal{P}i$ est un verbe modal, ALORS $\mathcal{P}i$ est une exigence, SINON $\mathcal{P}i$ est une information ». Les verbes modaux sont alors identifiés grâce à des parseurs morpho-syntaxiques qui analysent la structure grammaticale d'une phrase afin d'annoter chacun des termes avec sa catégorie syntaxique (sujet, verbe, complément, adjectif, adverbe, pronom, préposition, article, etc.). Cette fonctionnalité est appelée : Part-Of-Speech (POS) Tagging. Théoriquement, un parseur morpho-syntaxique est un modèle probabiliste qui a été préalablement entraîné sur un ensemble de phrases manuellement annotées. L'analyse morpho-syntaxique d'un texte permet alors d'identifier les exigences en recherchant toutes les phrases contenant un verbe modal.

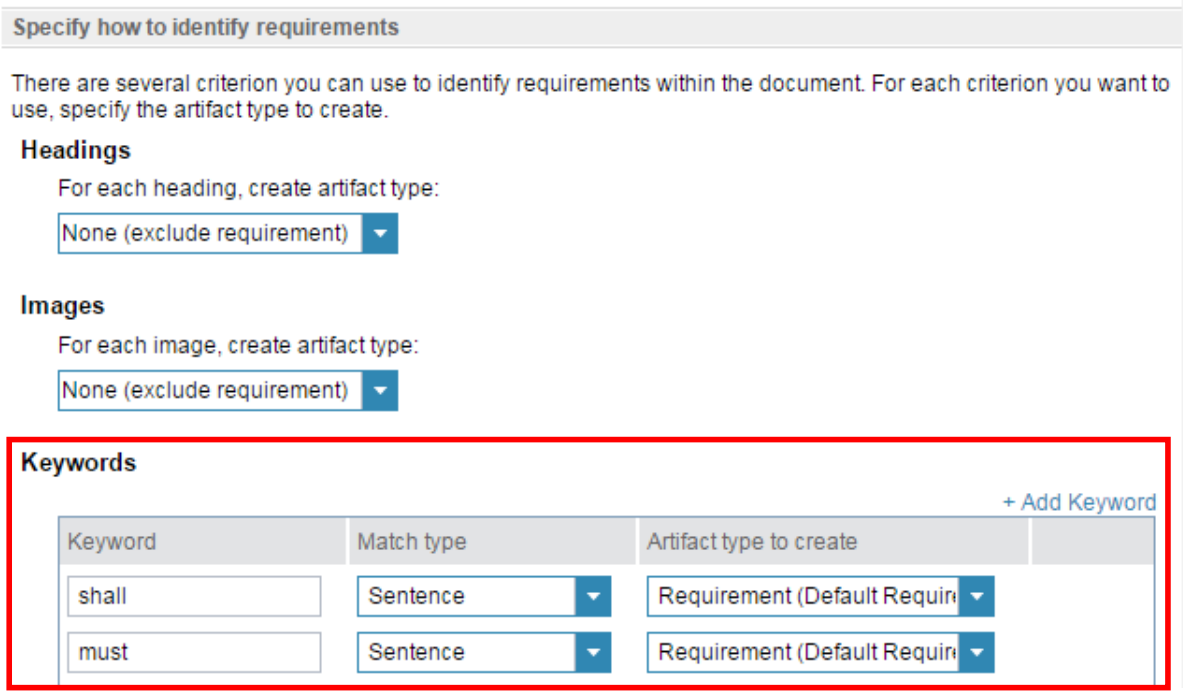


FIGURE 4.2 – Extraction des exigences à partir d'une liste de mots clés prédéfinie dans DOORS.

Les verbes modaux sont effectivement des caractéristiques clés pour extraire les exigences. Cependant, nous identifions deux limites à cette solution. La première limite est relative au contexte d'application. Si les documents analysés sont des livrables d'un processus d'ingénierie système et que, par conséquent, les auteurs sont familiers avec les règles de rédaction des exigences, alors les modaux peuvent suffire pour extraire les exigences. Ceci est une hypothèse forte, car certaines entreprises adoptent une démarche d'ingénierie système, mais, en réalité, leurs employés sont parfois bien loin de maîtriser les bonnes pratiques de rédaction des exigences. Par ailleurs, même si ces fondamentaux sont maîtrisés, ils ne le sont que pour un nombre limité de spécialistes qui ne font pas l'unanimité en ingénierie des produits. En pratique, les exigences peuvent être plus ou moins implicites et exprimées par des verbes prescriptifs – *require*, *want*, *expect*, *desire*, etc. – qui ne sont pas des verbes modaux. Dans notre contexte d'application qui ne fait aucune hypothèse vis à vis de la mise en œuvre d'une méthode spécifique – l'ingénierie système, par exemple – qui pourrait influencer sur la forme et le contenu des documents, nous devons tenir compte de la diversité du vocabulaire prescriptif.

La seconde limite correspond à la flexibilité du langage naturel. Dans la documentation technique, la rigueur littéraire laisse souvent à désirer. Par exemple, l'identification des phrases n'est pas triviale, car le point de fin de phrase est parfois omis. C'est notamment le cas après le titre d'un chapitre (A Fig. 4.3), une énumération, une légende ou dans la cellule d'un tableau. De plus, les exigences textuelles ne sont pas toujours de simples phrases. Par exemple, elles peuvent prendre la forme de listes (B Fig. 4.3). Étonnamment, il n'y a que Kof [2004] qui mentionne les difficultés liées à ces structures dont l'interprétation est évidente pour un humain mais difficile pour une machine. Dans son article, l'auteur rappelle que les outils d'analyse syntaxique ne fonctionnent qu'avec des phrases complètes et dont la grammaire est correcte. Pour obtenir de telles phrases, il reformule celles qui sont tronquées, comme les titres, les listes, les tableaux, etc. Pour un document de 80 pages cela a nécessité une journée et demi de travail manuel.

Extraire les exigences grâce aux termes légaux

Les termes légaux constituent un deuxième indice clé pour extraire des exigences. Breaux and Antón [2005] proposent trois listes de schémas permettant de reconnaitre trois formes légales : les droits, les

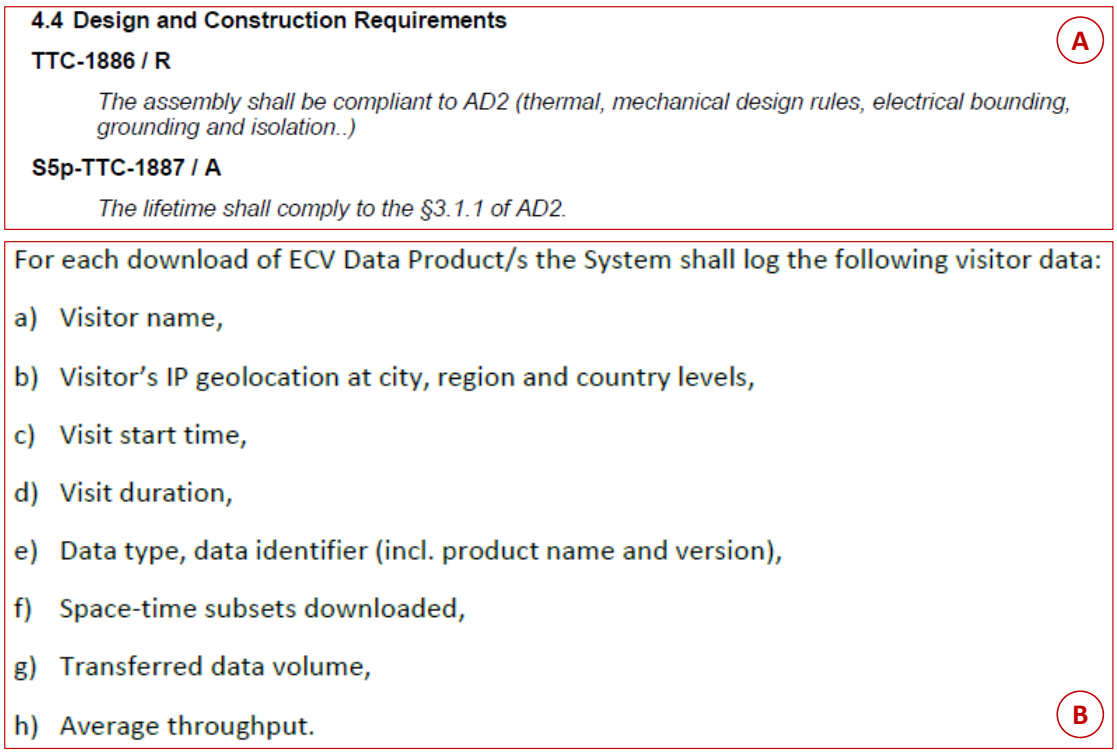


FIGURE 4.3 – Exemples de difficultés engendrées par la flexibilité du langage naturel. Le cas A montre que les phrases ne se terminent pas systématiquement par un point. Le cas B illustre 8 exigences exprimées au moyen d'une liste assimilable à 8 phrases.

obligations et les contraintes. Parmi les droits on retrouve les termes prescriptifs : *should*, *may be able to*, *may*, *can*, *could*, *permit*, *would not have to*, *does not restrict*, *does not require*. Parmi les obligations il y a : *should*, *should be*, *will*, *would*, *must*, *must be*, *which is charged with*, *requires* et *may not*. Enfin, les termes prescriptifs qui caractérisent les contraintes sont : *should be able to*, *may ... but ... would not have to*, *will ... on/upon*, *may ... for/for each*, *must ... to ensure that ... will*, *would not have to ... before*, *must first ... before*, *must ... by* et *should ... within*.

Plus récemment, Zeni et al. [2015] ont proposé GaiuST, un outil qui extrait les exigences légales afin d'assurer la conformité avec les régulations. Divers termes légaux servent à identifier les droits (*may*, *can*, *permit*, etc.), les anti-droits (*does not have a right to*, etc.), les obligations (*must*, *requires*, etc.) et les anti-obligations (*is not required*, *does not restrict*, etc.).

Comme les verbes modaux, les verbes légaux sont des indices clés pour extraire les exigences de documents, mais, dans notre contexte d'application, la liste des verbes prescriptifs ne se limite pas au jargon des textes légaux. La liste doit être plus exhaustive.

Enfin, plutôt que de rechercher des formes fléchies, nous proposons de normaliser tous les termes. La forme canonique des verbes étant l'infinitif, nous pourrions alors extraire un maximum d'exigences au risque de dégrader la précision.

Extraire les exigences grâce à des règles syntaxiques

Kang and Saint-Dizier [2013] ont proposé 12 règles syntaxiques pour identifier deux catégories d'exigences : les exigences induites par un terme lexical (*require* ou *need* et leurs formes fléchies) et les exigences composées d'un verbe modal. L'évaluation réalisée dans leur article a permis d'obtenir une précision de 76% sur un petit ensemble de 45 exigences. L'expérience menée sur un corpus de 64 pages

de texte prescrivant 215 exigences a donné des résultats encourageants avec une précision [1] de 97% et un rappel [2] de 96% [Kang and Saint-Dizier, 2014].

On peut s'interroger sur la diversité des données de test car un seul corpus a été utilisé. Par ailleurs, l'implémentation de règles syntaxiques est un choix restrictif vis-à-vis de notre objectif qui est de maximiser le rappel. En effet, on peut difficilement être sûr d'avoir imaginé toutes les variantes grammaticales permises par le langage naturel. Les auteurs font parfois preuve d'imagination.

Identifier les références transversales dans des exigences

Aucune solution technologique commerciale d'ingénierie des exigences ne peut identifier les exigences contenant des références transversales. Cependant, quelques travaux de recherche académique ont récemment proposé des éléments de réponse.

Zeni et al. [2015] ont implémenté un ensemble de règles pour extraire les références transversales contenues dans des exigences légales. Pour cela, ils s'appuient sur les règles de rédaction recommandées par « *The Guide of Legal writting* ». Ces règles font probablement office de référence pour rédiger des textes légaux, mais elles ne font certainement pas l'unanimité en ingénierie de produits.

Sannier et al. [2015] ont proposé une approche outillée pour détecter et résoudre les références transversales au moyen de deux éléments. Le premier est un schéma spécifique qui définit la structure d'un texte légal (chapitre, titre, section, paragraphe, article, etc.). Le second est un ensemble de patterns textuels caractéristiques d'une référence transversale selon la législation luxembourgeoise et validé sur des textes légaux canadiens qui ont la particularité d'être bilingues. Une méthode similaire avait déjà été mise en œuvre par de Maat et al. [2006] sur des textes de loi néerlandais. La contribution est très intéressante, mais elle est cependant trop spécifique pour répondre à notre besoin. D'une part, elle se limite aux textes légaux, d'autre part, c'est un exercice d'extraction d'informations qui ne correspond pas à notre objectif. En effet, nous ne cherchons pas à extraire les morceaux de textes qui correspondent aux références transversales, mais à catégoriser les exigences qui en contiennent au moins une. L'exercice est *a-priori* moins difficile mais mieux adapté à notre besoin.

Par ailleurs, comme Sannier et al. [2015] le soulignent justement, un algorithme d'apprentissage statistique tel que celui utilisé par Tran et al. [2014] est moins précis mais plus robuste, car on n'est jamais sûr de l'exhaustivité des règles vis-à-vis de la diversité du langage naturel.

Sannier et al. [2016] ont poursuivi leurs travaux en développant un modèle de classification automatique pour catégoriser chacune des références transversales selon sa sémantique (référence pour la conformité, référence vers une définition, référence vers des contraintes, etc.). Un tel niveau de détail d'analyse n'est pas nécessaire pour notre application.

Synthèse des méthodes d'extraction des exigences

Dans cet état de l'art, nous avons vu que des travaux avaient été réalisés pour extraire les exigences sur la base de verbes modaux ou de termes légaux. Cette définition du concept d'exigence est beaucoup plus restrictive que la notre, car nous souhaitons extraire tous les énoncés prescriptifs : besoins, souhaits, désirs, nécessités, etc. D'autres font l'hypothèse que les exigences suivent des structures syntaxiques. Cette solution est aussi trop restrictive pour notre définition du concept d'exigence. Nous ne faisons aucune hypothèse concernant la structure grammaticale d'une exigence. Par ailleurs, personne ne fait référence aux difficultés liées à l'analyse des documents de formats variés. Par exemple, les documents au format PDF sont parfois difficiles à analyser.

Des recherches ont été menées pour extraire des bouts de textes correspondant à des références transversales vers des documents externes. Ces études sont destinées à l'analyse de textes de lois dont la rédaction est normée. Nous souhaitons travailler avec des documents prescriptifs qui ne se limitent pas aux textes de lois, ces solutions pré-existantes ne sont donc pas applicables.

1. La précision P est la proportion d'objets pertinents parmi ceux qui sont retournés.
2. Le rappel R est la proportion d'objets pertinents retournés parmi tous les objets pertinents.

On peut donc conclure qu'aucune solution technologique ne permet d'identifier des exigences, de quelque origine que ce soit (légales et non légales), rédigées en anglais, qui contiennent des références transversales vers d'autres documents.

4.3 Proposition

Pour extraire les exigences spécifiées dans des documents semi-structurés et structurés, nous proposons un chaînage numérique constitué, d'une part, de divers parseurs et, d'autre part, d'un modèle de classification basé sur un ensemble de règles qui bénéficient des techniques de traitement du langage naturel. Enfin, une fonction de classification basée sur l'apprentissage machine identifie automatiquement les exigences qui font référence à des documents applicables.

Extraire les exigences des documents non structurés (FT11) et semi-structurés (FT12)

À chaque fois qu'un analyste soumet un ou plusieurs documents prescriptifs à l'environnement numérique, un processus d'analyse est initié. Le processus est différent selon le format des documents (Fig. 4.4).

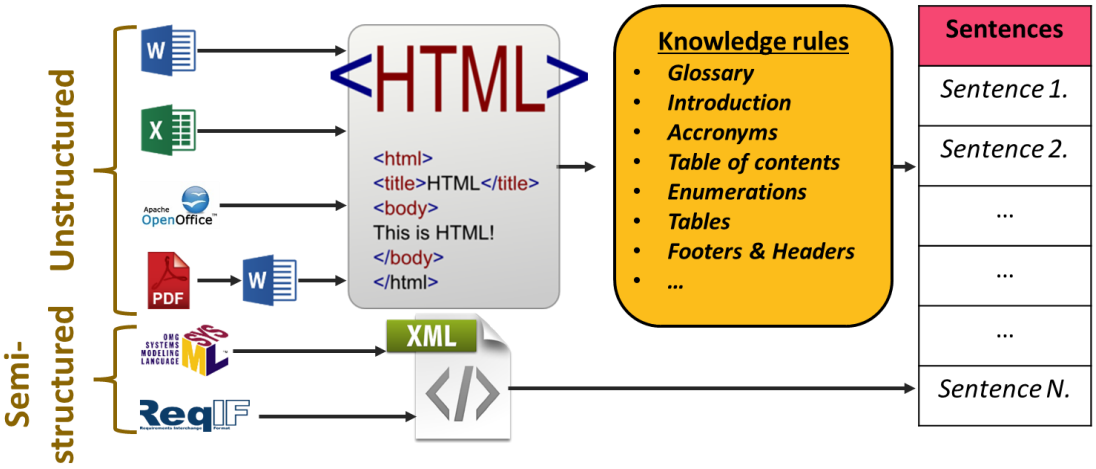


FIGURE 4.4 – Le processus global pour les différents formats supportés.

La première étape sert donc à identifier le format du document. Dans cette étude, nous considérons les documents non structurés MS Word, OpenOffice, PDF, MS Excel et OpenOffice Calc, ainsi que deux types de documents semi-structurés (XML/XMI).

Parmi les documents semi-structurés il y a les diagrammes d'exigences SysML édités avec l'éditeur Papyrus. Papyrus a été préféré car c'est l'un des éditeurs libres les plus populaires. Deuxièmement, les documents prescriptifs semi-structurés peuvent être des fichiers qu'un donneur d'ordres exporte à partir de son outil de gestion des exigences. Cet export doit être conforme à la norme d'échange ReqIF qui est supportée par une grande majorité des outils de gestion des exigences. Un second parseur ReqIF a été développé pour extraire les exigences.

Avec la boîte à outils Apache Tika [Mattmann and Zitting, 2011], nous pouvons non seulement identifier le format original d'un document (le MIME), mais aussi extraire les méta-données et le contenu textuel. Les méta-données qui correspondent à l'auteur, la date de dernière modification et le titre d'un document servent à renseigner les propriétés « Auteur », « Date de création » et « Source » des sommets « Exigences », lesquelles seront extraites ultérieurement. La date de dernière modification d'un

document peut ne pas correspondre à la date de création des exigences. Ces méta-données servent néanmoins à retrouver le document original et à posséder un minimum d'information : l'auteur ou le titre par exemple.

Après que le format du document a été reconnu, il faut extraire le contenu de chacun afin de pouvoir récupérer les phrases qui sont potentiellement des exigences. Pour les documents semi-structurés (XMI/XML), la tâche a déjà été traitée par Mokammel et al. [2013]. Néanmoins, tous les éditeurs SysML n'ont pas la même structure de données, donc pour éviter de développer un parseur ad-hoc pour chaque schéma, soit on transforme chacune des structures de données vers un schéma pivot unique – la norme d'échange des exigences ReqIF, par exemple –, soit on transforme chacune des structures de données vers notre graphe de propriétés. Ici, pour des contraintes de temps [3], un parseur XML a été développé selon la structure de données de l'éditeur Papyrus. Deuxièmement, les documents prescriptifs semi-structurés peuvent être des exports qu'un donneur d'ordres réalise à partir de son outil de gestion des exigences. Cet export doit être conforme à la norme d'échange ReqIF qui est supportée par une grande majorité des outils de gestion des exigences. Un second parseur ReqIF a été développé pour extraire les exigences.

Dans cette étude nous faisons l'hypothèse qu'une exigence est une phrase. Cette bonne pratique est appliquée dans la plupart des documents. Ainsi, pour les feuilles de calculs Excel et OpenOffice Calc, nous faisons aussi l'hypothèse qu'une exigence est une phrase spécifiée dans une cellule. La librairie Apache POI qui est embarquée dans Apache TIKA suffit donc à extraire le contenu textuel de chaque cellule pouvant potentiellement correspondre à une exigence.

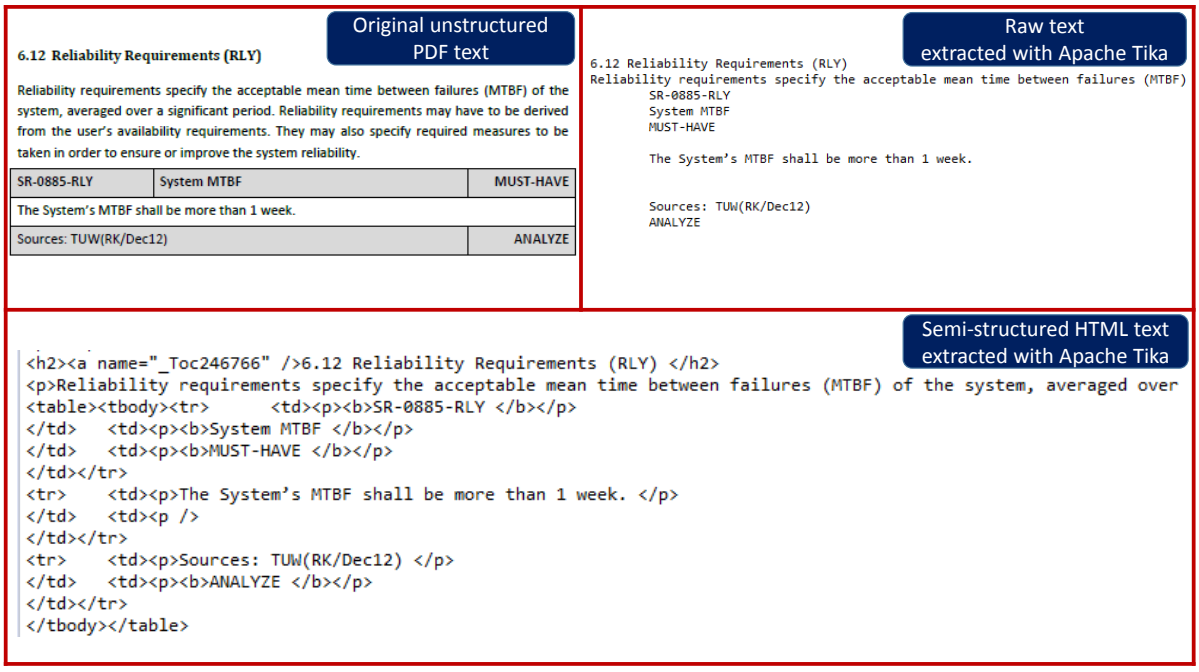


FIGURE 4.5 – Un extrait de spécification (en haut à gauche) et les deux types de sorties : texte brut (en haut à droite) ou texte HTML (en bas).

Apache Tika offre aussi la possibilité d'extraire le contenu [4] textuel brut comme une seule chaîne de caractères (Fig. 4.5). Néanmoins, pour les documents non structurés édités avec l'outil de traitement de texte Word ou OpenOffice, travailler avec une seule chaîne de caractères brute ne permet pas d'obtenir une bonne segmentation d'un texte en phrases. En effet, les éléments de la structure [5] interne au

3. Les transformations nécessaires n'ont aucune valeur ajoutée scientifique et peuvent être réalisées avec des langages comme ATL et des outils comme ceux développés par Talend.

4. En informatique on utilise communément l'anglicisme *parser*.

5. Attention, ici le terme de structure ne doit pas laisser penser que ce type de document est structuré tel que cela a été

document tels que les titres, les tableaux, les en-têtes et pieds de pages, et les listes sont concaténés avec le texte environnant. Il existe néanmoins une alternative pour contourner ce problème : convertir le contenu textuel de chaque document au format standard HTML avec Tika (Fig. 4.5). Avec un parseur HTML comme JSOUP, les balises HTML peuvent alors être exploitées par des règles pour reconstruire les éléments de la structure interne d'un document.

Une première limitation de notre proposition concerne les documents au format PDF. Quand ces derniers sont transformés en HTML, la plupart des balises utiles à l'analyse de la structure interne ne sont pas présentes, et ce, même en passant par des convertisseurs tels que PDFBox, car un fichier PDF n'embarque aucun élément de structure, hormis un sommaire dans les dernières versions d'Adobe Acrobat Reader. Il existe néanmoins une solution partielle en utilisant les capacités natives de Word et Excel à restituer des fichiers PDF dont ils sont à l'origine. Ainsi, dès qu'un document prescriptif au format PDF est détecté, il faut vérifier si ce fichier a été généré avec le logiciel Word ou Excel. Pour identifier les fichiers Word, on fait l'hypothèse qu'un document contient au moins un élément de structure interne (en-tête, pied de page, tableau, liste, etc.) et recherche les balises HTML correspondantes. Les balises HTML servant à modéliser les cellules d'une feuille de calcul suffisent pour conclure que le document a été rédigé avec Excel. La conversion d'un document PDF en un nouveau document Word ou Excel se fait via un script en Batch qui, en masqué, ouvre le document PDF avec Word ou Excel et l'enregistre dans le format natif avant que le contenu soit extrait avec le parseur adéquat. Dans le second cas, si le PDF n'a pas été généré avec Word ou Excel, deux options sont envisageables. Soit on utilise un ensemble d'expressions régulières pour essayer de reconstruire la structure interne du document, soit on signale à l'utilisateur que ce type de format n'est pas supporté. Les expressions régulières peuvent suffire pour des documents dont la structure est relativement simple (pas de listes imbriquées, pas de tableaux, etc.). Une telle approche a néanmoins montré ses limites face à la diversité des contenus textuels. Les documents prescriptifs sont tous différents les uns des autres, il n'y a donc pas de règles universelles. Les règles qui améliorent l'analyse d'un document dégradent l'analyse d'un autre. Jusqu'ici, l'environnement numérique ne traite donc pas ce type de document et le signale à l'analyste. Cette technique d'utiliser les éditeurs Microsoft n'est pas une limite majeure car ils dominent le marché.

Chacun des contenus textuels est finalement stocké dans une table de Hachage, c'est-à-dire une structure de données associative tabulaire à deux entrées : clé et valeur. La clé stocke le titre d'un chapitre, tandis que la valeur stocke le contenu textuel du chapitre. Le fait de savoir à quel chapitre un énoncé textuel appartient peut être utile pour diverses raisons : classification des énoncés, classification des images, traitement parallèle, etc. Certains chapitres peuvent contenir des énoncés prescriptifs qui n'ont rien à voir avec le produit spécifié – les *copyrights*, par exemple –, lesquels sont susceptibles de dégrader la qualité de l'analyse. Ces chapitres sont volontairement négligés en identifiant des mots clés tels que *appendix*, *glossary*, *scope*, *introduction*, *accronym*, *preamble*, et *terminology*.

Quand tous les documents ont été parsés, leur contenu textuel est envoyé dans un processus de traitement du langage naturel. Pour réaliser les opérations de traitement du langage naturel nous avons choisi d'utiliser la bibliothèque libre Stanford CoreNLP [Manning et al., 2014] pour plusieurs raisons. La première est parce que notre prototype est, comme cela nous a été fortement suggéré par notre partenaire industriel, une application Java qui est aussi le langage utilisé pour développer CoreNLP. Par ailleurs, CoreNLP est non seulement la solution la mieux documentée, mais c'est aussi la plus sophistiquée grâce à un groupe de recherche « hyperactif ». Par exemple, il supporte plusieurs langues – chinois, français, allemand, et espagnol – ce qui peut potentiellement servir à étendre les capacités de notre solution.

La première étape de traitement consiste à segmenter chacun des textes. La segmentation (*tokenization*, en anglais) découpe une chaîne de caractères en unités lexicales distinctes : les *tokens*. Les tokens sont des séquences de caractères, des mots ou de la ponctuation, destinés à être traité. Le PTBTokenizer de CoreNLP est un automate fini déterministe capable de segmenter un énoncé textuel à la cadence de 1 000 000 mots par seconde. Diverses heuristiques permettent, par exemple, de deviner si une apostrophe fait partie d'un mot ou si un point est un point de fin de phrase.

défini pour les données structurées en section 3.2, car les données qu'il contient ne sont pas sous forme tabulaire comme dans une base de données relationnelle.

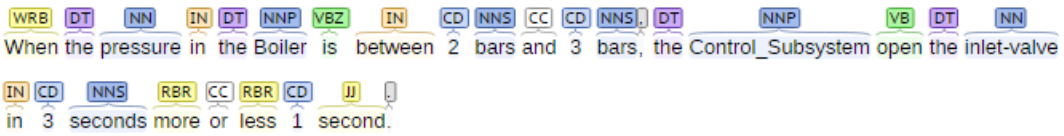


FIGURE 4.6 – Phrase segmentée en tokens annotés avec leur catégorie syntaxique.

Ces tokens sont ensuite normalisés. Dans le jargon du traitement du langage naturel on dit qu'on lemmatise (*lemmatization*, en anglais) les tokens. La lemmatisation normalise chacun des tokens en réduisant les formes dérivées aux formes canoniques : les lemmes (*lemmas*, en anglais). Les lemmes correspondent aux entrées d'un dictionnaire. Par exemple, dans la phrase de la figure 4.6, le forme conjuguée du token « *is* » et le nom au pluriel « *bars* » sont respectivement réduits aux lemmes « *be* » et « *bar* ». La lemmatisation non seulement accroît le rappel, mais aussi réduit la taille du vocabulaire.

En plus d'être normalisés, les tokens sont annotés par un annotateur syntaxique (*Part-Of-Speech tagger*, en anglais). Ainsi, comme dans l'exemple de la figure 4.6, une catégorie grammaticale (nom, verbe, adjectif, adverbe, pronom, etc.) est attribuée à chacun des tokens. Le POS-tagger implémenté par Toutanova et al. [2003] dans la bibliothèque CoreNLP est un modèle de séquence probabiliste qui apprend les séquences grammaticales les plus probables à partir d'un corpus d'entrainement préalablement annoté à la main.

Enfin, la quatrième et dernière tâche de traitement du langage naturel est la segmentation du texte en phrases (*sentence splitting*, en anglais) qui sont potentiellement des exigences.

Cette analyse de traitement du langage naturel permet de construire un modèle de classification. Il existe deux approches pour classifier du texte, ou, plus généralement, des objets. La première approche vise à programmer explicitement une ou plusieurs règles du type « *SI ... ALORS ... SINON* » basées sur des connaissances. Un objet est alors classifié dans une ou plusieurs classes selon la valeur de la condition « Vraie ou Faux ». La seconde approche s'appuie sur un algorithme d'apprentissage pour apprendre une fonction de classification qui associe une catégorie à un objet. La phase d'apprentissage d'une fonction de classification détermine un ensemble de paramètres statistiques à partir d'un corpus d'objets qui a été préalablement annoté par une ou plusieurs personnes. À ce jour, aucune solution d'apprentissage machine n'a été mise en œuvre dans un problème de classification « exigence » *vs.* « information ». Divers arguments peuvent justifier cette tendance. Dans de nombreux cas, les modèles de classification basés sur des règles ont la particularité d'obtenir des résultats précis limitant le nombre de résultats faux positifs [6]. Au contraire, les modèles issus de l'apprentissage machine sont beaucoup plus flexibles car l'optimisation des paramètres tend à dessiner une ou plusieurs fonctions de décision résultant d'un compromis sur un échantillon d'exemples. Comme cela a été discuté dans l'état de l'art, les termes prescriptifs suffisent à identifier les exigences parmi un ensemble d'énoncés textuels. Ici, les termes prescriptifs ne se limitent pas aux verbes modaux (*shall*, *must*, etc.), ils incluent aussi les verbes qui expriment une requête (*require*), une obligation (*obligate*), un devoir (*have to*, *demand*), une restriction (*restrict*), un service (*provide*), un droit (*permit*), un besoin (*need*), un désir (*desire*), *crave*, une volonté (*want*), une attente (*expect*),)une commande (*mandate*), ou un souhait (*wish*). À la différence des outils de gestion des exigences, l'approche proposée ne s'appuie par sur une correspondance exacte entre les termes prescriptifs définis par l'utilisateur et le texte brut. En effet, puisqu'un terme prescriptif – *require*, par exemple – peut prendre des formes dérivées – *required* et *requires*, par exemple –, dans un souci d'exhaustivité, les règles classifient les phrases à partir des lemmes, ce qui évite de définir toutes les formes dérivées.

Chacune des exigences extraites est modélisée par un sommet dans le graphe de propriétés. En plus des propriétés correspondant aux méta-données du document prescriptif et du titre de la section dans laquelle l'exigence se trouve, les objets « Exigence » sont enrichis avec l'énoncé, la liste des tokens, la liste des lemmes, et la liste des annotations syntaxiques (Fig. 4.7). Par ailleurs, tous les tokens clés, c'est-

6. Les résultats faux postitifs sont les objets non pertinents retournés.

à-dire ceux dont la catégorie syntaxique est un nom, un verbe, un adjectif ou un adverbe (*open classes*, en anglais [Jurafsky and Martin, 2000]), ainsi que les lemmes et catégories grammaticales associées sont stockés dans trois listes supplémentaires pour faciliter les opérations de traitement qui suivront.

uniqueId	2d3f2d32-df4d-464e-a511-aaf01a8bca59
author	IT
creationDate	1464769099730
modificationDate	1464769099730
statement	2.1.2 The installation shall comply with SANS 10142.
originalStatement	2.1.2 The installation shall comply with SANS 10142.
source	General electric.doc
version	2005-12-07T12:15:00Z
heading	2.1 General
similarityScore	1
words	[2.1.2, The, installation, shall, comply, with, SANS, 10142, .]
lemmas	[2.1.2, the, installation, shall, comply, with, sans, 10142, .]
tags	[CD, DT, NN, MD, VB, IN, NN, CD, .]
keyWords	[installation, comply, sans]
keyLemmas	[installation, comply, sans]
keyTags	[NN, VB, NN]

FIGURE 4.7 – Propriétés d'un objet exigence dans le modèle de graphe de propriétés.

Identifier les références transversales (FT13)

Maintenant que les exigences sont extraites et modélisées, nous souhaitons identifier celles qui font référence à un document externe potentiellement prescriptif et applicable. Par exemple, l'énoncé de l'exigence de la figure 4.7 contient une référence transversale vers la norme nationale Sud-Africaine (*South African National Standard* (SANS), en anglais) 10142.

Les travaux résumés dans l'état de l'art se concentre sur l'extraction de références transversales contenues dans des exigences au moyen de patrons de rédaction de textes de loi. Ici, nous souhaitons nous affranchir des limites liées au domaine d'application, car les références transversales peuvent, certes, référer à un texte de loi, mais aussi à une norme, une police, une régulation, une spécification, etc.

Il existe trois approches pour extraire des informations. La première consiste à définir des règles en programmant des expressions régulières qui exploitent des dictionnaires et des connaissances spécifiques au domaine d'application. Ce type de solution marche plutôt bien quand la diversité est limitée comme dans l'exercice d'extraction de numéros de téléphone dans une page Web.

La deuxième approche, celle que nous avions initialement envisagée, ce sont les modèles de séquence. Les modèles de séquence servent à annoter une séquence de données. Par exemple, en considérant une phrase comme une séquence de mots notée X, on peut chercher une séquence d'annotations notée Y correspondant aux catégories syntaxiques. La séquence de mots X = « La personne mange » pourrait être annotée avec la séquence d'annotations Y = « Déterminant Nom Verbe_intransitif ». L'intuition sous-jacente aux modèles de séquence est qu'il existe des dépendances entre les éléments de la séquence. En effet, en décomposant X et Y en deux ensembles de variables aléatoires $X = \{X_1, X_2, ..., X_n\}$ et Y = $\{Y_1, Y_2, ..., Y_n\}$ on peut intuitivement identifier que :

– si $X_i = la$, alors Y_i = Déterminant ou Y_i = Pronom, et
– si en plus Y_{i+1} = Nom, alors Y_i = Déterminant.

Un modèle de séquence est un modèle graphique (Fig. 4.8) qui modélise les dépendances entre des variables aléatoires correspondant aux nœuds d'un graphe. En supposant qu'il existe une distribution de probabilité $P(X|Y)$, l'exercice est de définir les paramètres d'un modèle p à l'aide de couples (X, Y). Il existe différents modèles de séquence habituellement divisés en deux catégories : les modèles discriminatifs (Conditional Random Fields [Lafferty et al., 2001], par exemple) qui utilisent la probabilité conditionnelle $P(X|Y)$, et les modèles génératifs (Hidden Markov Model, par exemple) qui s'appuient sur la probabilité jointe $P(X, Y)$.

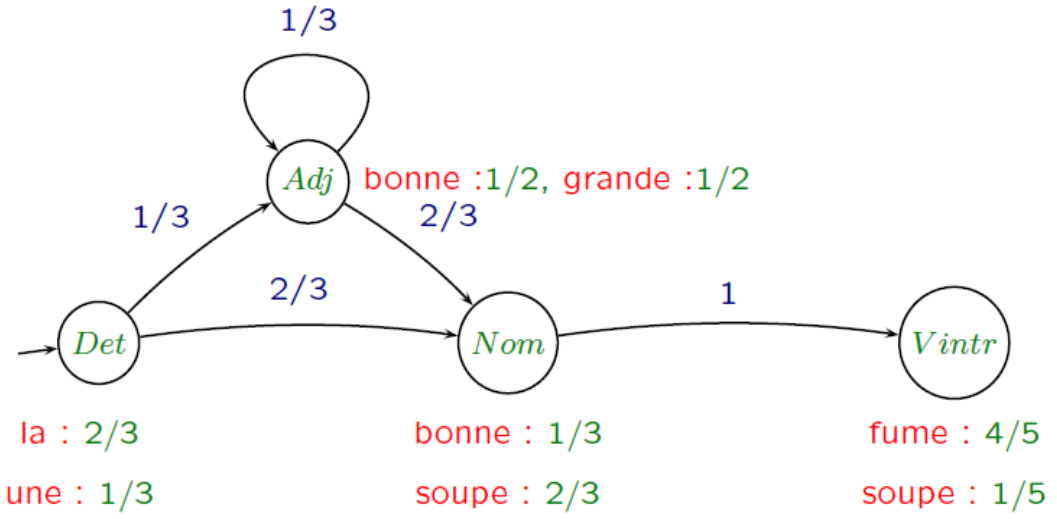


FIGURE 4.8 – Exemple de modèle de séquence (*Hidden Markov Model*).

Pour apprendre les paramètres, le modèle exploite un corpus d'apprentissage qui a été manuellement annoté. Une fois que le modèle de séquence p a appris les paramètres, pour toute nouvelle séquence X, le modèle annote chacune des unités de la séquence avec l'annotation Y la plus probable, c'est-à-dire $\hat{Y} = \arg\max_Y P(Y|X)$. Dans l'exemple de la figure 4.8 il y a deux séquences possibles « Déterminant Adjectif Nom » ou « Déterminant Nom Verbe », mais la plus probable est la première séquence car :

– P(x = une bonne soupe, y = Det Adj Nom) = 1/3*1/3*1/2*2/3*2/3 = 2/81
– P(x = une bonne soupe, y = Det Nom Vbe_int) = 1/3*2/3*1/3*1*1/5 = 2/135

La construction d'un corpus d'apprentissage constitué d'exemples manuellement annotés est fastidieuse, en particulier pour les corpus qui nourrissent les modèles de séquence. En effet, pour fournir des résultats pertinents, un algorithme d'apprentissage machine nécessite, au minimum, des centaines, voire des milliers d'exemples annotés. Dans le cas d'un modèle de séquence, le nombre d'annotations nécessaires n'est pas égal au nombre d'exemples, mais au nombre d'unités qui constituent une séquence. Autrement dit, dans notre cas, nous n'annoterions pas une centaine d'exigences, mais chacun des termes constituant la centaine d'exigences. Étant donné l'envergure de notre environnement numérique, nous ne pouvions pas réaliser ce travail manuel dans le temps imparti. Par ailleurs, cette activité d'extraction d'information vise à extraire la chaîne de caractères correspondant à une référence transversale. Or, nous

ne souhaitons pas atteindre un tel niveau de détail. Nous cherchons à identifier les exigences contenant des références transversales. Par conséquent, bien que les modèles de séquence soient prometteurs, nous avons opté pour la troisième approche – la classification par apprentissage machine –, laquelle vise à classifier automatiquement les exigences selon deux classes : « exigences avec référence transversale » *vs.* « exigences sans référence transversale ».

La statistique peut être divisée en trois catégories selon l'objectif de modélisation : décrire, expliquer ou prévoir. La statistique descriptive vise à, d'une part, fournir des représentations graphiques synthétiques de données (histogramme, nuage de points, diagramme boîte, etc.) et, d'autre part, à calculer des résumés numériques (moyenne, médiane, écart type, variance, etc.). La statistique inférentielle a pour objectif de préciser un phénomène sur une population globale à partir de l'observation d'un échantillon. Enfin, l'apprentissage statistique vise à prévoir la valeur d'une variable de sortie à partir d'une ou plusieurs variables d'entrée. Alors que les mathématiciens parlent d'apprentissage statistique (*statistical learning*, en anglais) comme une sous-discipline de la statistique, les informaticiens parlent d'apprentissage machine (*machine learning*, en anglais) comme une sous-discipline de l'intelligence artificielle. Il n'y a pas de distinction stricte entre les deux domaines si ce n'est le profil des acteurs et les applications qui les préoccupent. Les marketeurs ont néanmoins popularisés la dénomination la plus « mystique » : l'apprentissage machine.

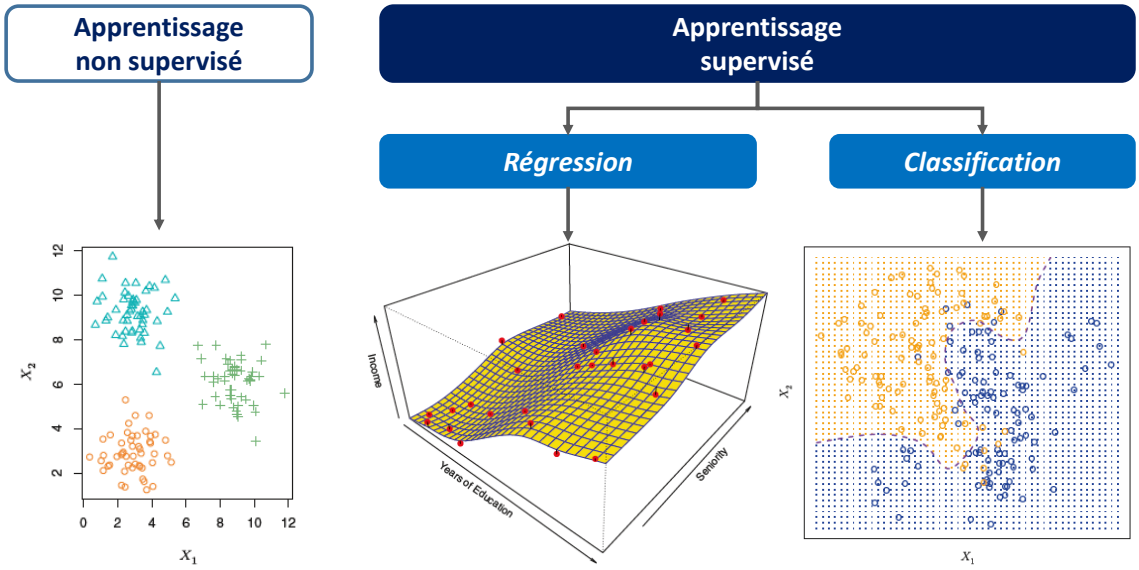


FIGURE 4.9 – Les 3 grandes approches d'apprentissage automatique (adapté de James et al. [2013]).

Une machine [7] dont le comportement se base sur un algorithme d'apprentissage statistique ne nécessite pas d'être explicitement programmée pour fonctionner. La machine apprend, ponctuellement ou continuellement, à réaliser une tâche ou une séquence de tâches à partir d'un grand nombre d'exemples. Une première typologie des approches d'apprentissage machine sépare l'apprentissage « supervisé » et « non supervisé » (Fig. 4.9).

L'apprentissage supervisé (Fig. 4.10) consiste à entraîner une machine de sorte à ce qu'elle puisse prévoir une réponse [8] y à partir d'un vecteur de caractéristiques [9] (*features vector*, en anglais) $X \in \mathbb{R}^n$ tel que $X = (x_1, x_2, ..., x_n)^T$. Si la réponse y à prévoir est une variable continue [10], alors c'est un problème de « régression ». La prévision du coût d'une habitation en fonction du nombre de pièces et de la superficie totale est un exercice de régression. Si la réponse y à prévoir est une variable discrète [11],

7. Le mot machine est à prendre au sens large : un ordinateur, un robot, un véhicule automobile, etc.
8. On parle aussi de sortie, de variable de sortie, de variable dépendante, ou de variable expliquée.
9. On parle aussi d'entrées, de variables d'entrée, de prédicteurs, de variables indépendantes, d'attributs, ou de variables explicatives.
10. On parle aussi de variable quantitative.
11. On parle aussi de variable qualitative.

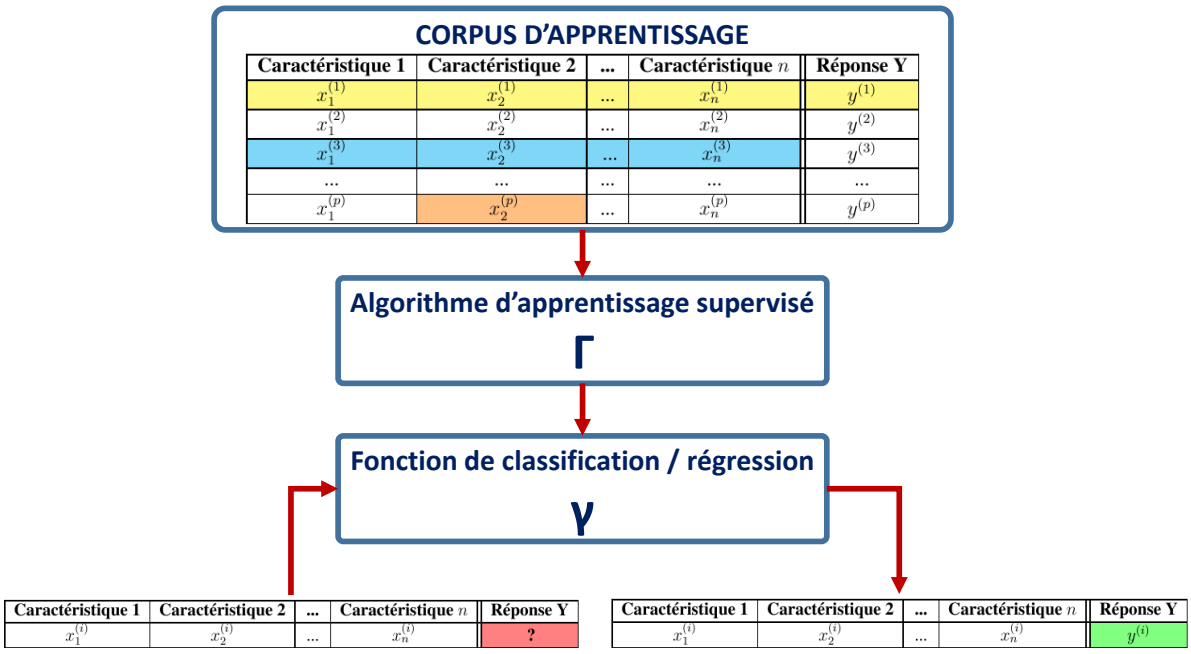


FIGURE 4.10 – Apprentissage supervisé dont les étapes successives sont : (1) construire un corpus d'apprentissage constitué d'un grand nombre d'exemples p ; (2) choisir un algorithme d'apprentissage supervisé Γ ; (3) entraîner Γ avec le corpus d'apprentissage afin d'obtenir une fonction de classification (resp. de régression) γ ; (4) utiliser γ pour prévoir la valeur d'une réponse discrète (resp. continue) y à partir d'un vecteur de caractéristiques X.

alors c'est un problème de « classification ». La prévision binaire (oui ou non) d'un cancer à partir de l'image d'une tumeur est un exercice de classification. Avec l'arrivée des méthodes de classification, certains ont vu le salut de l'intelligence artificielle – des machines qui battent les meilleurs joueurs de Go, par exemple –, mais l'apprentissage machine n'a rien de magique ; ce ne sont ni plus ni moins que des théories mathématiques qui nécessitent d'être alimentées par de gros volumes de données servant à définir les paramètres d'un modèle. En effet, en supposant qu'il existe une relation entre un vecteur de caractéristiques X et une réponse y, l'apprentissage supervisé est un ensemble de techniques pour estimer une fonction inconnue f tel que $y = f(X) + \epsilon$, c'est-à-dire $y = f(x_1, x_2, ..., x_n) + \epsilon$ avec ϵ une erreur indépendante de X dont la moyenne est nulle. Pour estimer la fonction inconnue f qui relie X et y, c'est-à-dire pour trouver une fonction $\hat{f}$ tel que $y \approx \hat{f}(X)$, l'algorithme d'apprentissage s'appuie sur un corpus d'apprentissage. Comme l'illustre la table 4.1, un corpus d'apprentissage $\mathbb{X}$ est un espace de grande dimension correspondant à un ensemble d'exemples [12] $(X^{(i)}, y^{(i)})$ (en jaune) tel que $\mathbb{X} = \{(X^{(1)}, y^{(1)}), (X^{(2)}, y^{(2)}), ..., (X^{(p)}, y^{(p)})\}$, avec $X^{(i)} = (x_1^{(i)}, x_2^{(i)}, ..., x_n^{(i)})^T$ le i$^{\text{ème}}$ vecteur de caractéristiques (en bleu). L'entrée $x_n^{(i)}$ (en orange) est alors la p$^{\text{ème}}$ caractéristique du i$^{\text{ème}}$ exemple. C'est parce qu'un grand nombre d'exemples est utilisé pour entraîner un algorithme de classification que l'apprentissage est dit supervisé.

L'apprentissage statistique non supervisé (*unsupervised learning* ou *clustering*, en anglais) [Fahad et al., 2014] ne cherche pas à estimer une fonction f pour prévoir une réponse y à partir d'un vecteur de caractéristiques X. Il n'y a tout simplement pas de réponse y. L'apprentissage statistique vise à découvrir des sous-groupes (*clusters*, en anglais) où les objets d'un sous-groupe sont à la fois similaires entre eux et différents de ceux contenus dans les autres sous-groupes. La recherche de communautés dans un réseau social est un exemple de problème d'apprentissage non supervisé. Ce type d'apprentissage est donc beaucoup plus dédié à l'exploration de données qu'à la prévision d'une variable continue ou discrète.

Si nous revenons à notre problème initial de classification des exigences contenant ou ne contenant

12. On parle aussi d'observations ou d'instances.

Caractéristique 1	**Caractéristique 2**	**...**	**Caractéristique n**	**Réponse Y**
$x_1^{(1)}$	$x_2^{(1)}$	...	$x_n^{(1)}$	$y^{(1)}$
$x_1^{(2)}$	$x_2^{(2)}$	...	$x_n^{(2)}$	$y^{(2)}$
$x_1^{(3)}$	$x_2^{(3)}$	...	$x_n^{(3)}$	$y^{(3)}$
...	...	...	...	...
$x_1^{(p)}$	$x_2^{(p)}$	...	$x_n^{(p)}$	$y^{(p)}$

TABLE 4.1 – Exemple d'un corpus d'apprentissage dédié à l'apprentissage statistique supervisé. La cellule orange est la 2ème caractéristique du pème exemple. Les cellules bleues correspondent au troisième vecteur de caractéristiques $X^{(3)}$. La ligne jaune est le 1er exemple d'apprentissage $(X^{(1)}, y^{(1)})$.

pas de référence transversale, on peut conclure que c'est un problème d'apprentissage supervisé et, plus précisément, que c'est un problème de classification car la réponse à prévoir y est une variable discrète binaire {0,1}.

La première étape d'un exercice d'apprentissage statistique correspond à la formulation du problème. Soit :

- $E = (e_1, e_2, ..., e_n) \in \mathbb{R}^n$ un vecteur de n = 11 caractéristiques représentant une exigence.
- $C = \{c_1, c_2\}$ une réponse discrète binaire pour laquelle les valeurs c_1 et c_2 représentent deux classes :
  - c_1 = exigence avec référence transversale, et
  - c_2 = exigence sans référence transversale.

L'objectif est alors d'utiliser un algorithme d'apprentissage supervisé Γ pour apprendre une fonction de classification γ qui annote une nouvelle exigence $E^{(i)}$ avec une classe $C^{(i)}$.

La deuxième étape correspond à la définition d'un corpus d'entrainement constitué de p = 500 exemples $(E^{(i)}, C^{(i)})$, tel que $\mathbb{E} = \{(E^{(1)}, C^{(1)}), (E^{(2)}, C^{(2)}), ..., (E^{(p)}, C^{(p)})\}$. Les 500 exemples ont été manuellement extraits de spécifications et annotés par une personne experte en ingénierie des exigences avec l'une des deux classes c_1 ou c_2. Un extracteur de caractéristiques (*features extractor*, en anglais) a été spécifiquement développé pour transformer chacune des exigences du corpus d'apprentissage en un vecteur de 11 caractéristiques. L'extracteur de caractéristiques est un ensemble de règles qui permet de savoir si une exigence donnée possède ou non certains indicateurs qui caractérisent une exigences contenant une référence transversale. Pour cela, il faut préalablement segmenter chacune des exigences en tokens, puis normaliser les tokens en lemmes avant d'appliquer les règles suivantes :

- « isPrescriptiveTerm ». Est-ce que l'un des lemmes correspondant à un terme prescriptif tel que *specify*, *detail*, *define*, *accordance*, *prescribe*, *set*, *comply*, *agreement*, *i.a.w*, *compatible*, *as per*, *conform*, *state*, *refer* est dans l'exigence ? Si oui, alors la 1ère caractéristique e_1 = 1, sinon e_1 = 0.
- « isPreposition ». Est-ce que l'un des lemmes correspondant à une préposition telle que *as*, *in*, *at*, *under*, *with*, *to*, *herein* est dans l'exigence ? Si oui, alors la 2ème caractéristique e_2 = 1, sinon e_2 = 0.
- « isStructure ». Est-ce que l'un des lemmes correspondant à un élément de structure tel que *paragraph*, *$*, *section*, *chapter* est dans l'exigence ? Si oui, alors la 3ème caractéristique e_3 = 1, sinon e_3 = 0.
- « isMixDigitsCharacters ». Est-ce que l'un des lemmes est un mélange de lettres et de chiffres tel que *ECSS-E-40* ? Si oui, alors la 4ème caractéristique e_4 = 1, sinon e_4 = 0.
- « isMultipleCapitals ». Est-ce que l'un des lemmes contient plusieurs majuscules tel que *ECSS* ? Si oui, alors la 5ème caractéristique e_5 = 1, sinon e_5 = 0.
- « isLegalDocument ». Est-ce que l'un des lemmes correspondant à un document légal tel que *standard*, *policy*, *regulation*, *guideline*, *law* ? Si oui, alors la 6ème caractéristique e_6 = 1, sinon e_6 = 0.
- « isNotInWordNet ». Est-ce que l'un des lemmes n'est pas une entrée du thésaurus linguistique WordNet tel que *ECSS* ? Si oui, alors la 7ème caractéristique e_7 = 1, sinon e_7 = 0.

– « isStandard ». Est-ce que l'un des lemmes est une norme telle que *ISO*, *IEEE*, *ECSS*, *IEC*, *CS*, *ESA*, *CEN*, *CENELEC*, *ETSI* ? Si oui, alors la 8ème caractéristique $e_8 = 1$, sinon $e_8 = 0$.
– « isMultipleFullStops ». Est-ce que l'un des lemmes contient plusieurs points tel que *3.10.2.11* ? Si oui, alors la 9ème caractéristique $e_9 = 1$, sinon $e_9 = 0$.
– « isDash ». Est-ce que l'un des lemmes contient un trait d'union tel que *ECSS-E-40* ? Si oui, alors la 10ème caractéristique $e_{10} = 1$, sinon $e_{10} = 0$.
– « isBracket ». Est-ce que l'un des lemmes contient une parenthèse ou un crochet tel que *{*, *[*,*(* ? Si oui, alors la 11ème caractéristique $e_{11} = 1$, sinon $e_{11} = 0$.

Par exemple, l'énoncé de l'exigence « *Requirement and Architecture Engineering shall be done in accordance to ECSS-E-40.* » est transformée en un vecteur de 11 caractéristiques : [1,1,0,1,1,0,1,0,0,1,0]. La figure 4.11 est un extrait du corpus d'apprentissage utilisé pour entraîner un algorithme de classification.

```
@RELATION trainingLegalRequirements

@ATTRIBUTE statement string
@ATTRIBUTE isPrescriptiveTerm NUMERIC
@ATTRIBUTE isPreposition NUMERIC
@ATTRIBUTE isParagraph NUMERIC
@ATTRIBUTE isMixDigitsCharacters NUMERIC
@ATTRIBUTE isMultipleCapitals NUMERIC
@ATTRIBUTE isLegalDocument NUMERIC
@ATTRIBUTE isNotInWordNet NUMERIC
@ATTRIBUTE isStandard NUMERIC
@ATTRIBUTE isMultipleFullStops NUMERIC
@ATTRIBUTE isDash NUMERIC
@ATTRIBUTE isBracket NUMERIC
@ATTRIBUTE @@class@@ {L,NL}

@DATA
'For larger conduit sizes the requirements of SANS 10142 shall be met.',0,0,0,0,1,0,0,0,0,0,0,L
'The ranging signal shall be compatible with [ES 050].',1,1,0,0,1,0,1,0,0,0,1,L
'Material selection shall be in accordance and ECSS-E-30 Part 8.',1,1,0,1,1,0,0,0,0,1,0,L
```

FIGURE 4.11 – Extrait du corpus d'apprentissage utilisé pour estimer une fonction de classification qui identifie les exigences qui font appel à des références externes. L'énoncé en position de première caractéristique est purement informatif et n'est pas présent en pratique.

La troisième étape consiste à choisir un algorithme d'apprentissage Γ à entraîner pour estimer la fonction de classification γ. Il existe une multitude d'algorithmes d'apprentissage supervisé. La sélection passe donc par une phase expérimentale qui sera détaillée en section 4.4. Ici, l'objectif n'est pas de restituer une leçon exhaustive des algorithmes d'apprentissage supervisé. De nombreux ouvrages de référence le font déjà très bien [Mitchell, 1997, Bishop, 2007, Hastie et al., 2009, Witten et al., 2011, James et al., 2013]. Les principaux algorithmes applicables à un problème de classifications que nous utilisons par la suite sont néanmoins introduits ci-dessous.

– **Modèle Bayésien.** On peut montrer que le taux d'erreur de classification peut être minimisé par une fonction de classification qui assigne la réponse y la plus probable à un vecteur de caractéristiques $X = (x_1, x_2, ..., x_n)$. Il suffit alors de calculer la probabilité conditionnelle $P(y|X)$. Ce type d'algorithme de classification est un modèle Bayésien. Dans un problème de classification binaire où la réponse y peut prendre l'une des deux valeurs du couple {1,0}, si $Pr(y = 1|X) > 0.5$, alors le vecteur de caractéristiques X appartient à la classe 1, sinon il appartient à la classe 2.
– **Régression logistique.** Comme son nom l'indique, la régression logistique est dérivée de la régression. La régression vise à prévoir la valeur d'une variable continue y à partir d'un vecteur de caractéristiques $X = (x_1, x_2, ..., x_n)$. Dans le cas d'une régression linéaire simple, on cherche un couple de paramètres (β_0, β_1) tel que $\beta_0 + \beta_1 x_1 \approx y$. Par exemple, quel est le coût y d'une habitation en fonction de la surface x_1. Dans le cas d'une régression linéaire multiple, on cherche un vecteur de paramètres $(\beta_0, \beta_1, ..., \beta_n)$ tel que $\beta_0 + \beta_1 x_1 + ... + \beta_n x_n \approx y$. La régression

logistique embarque le modèle de la régression multiple dans une fonction logistique $\frac{e^X}{1+e^X}$ ou $\frac{e^{(\beta_0+\beta_1 x_1+...+\beta_n x_n)}}{1+e^{(\beta_0+\beta_1 x_1+...+\beta_n x_n)}}$ afin de calculer la probabilité qu'un nouveau vecteur de caractéristiques $X = (x_1, x_2, ..., x_n)$ appartient à l'une des deux classes $y = 1$ ou $y = 0$.

– **Arbre de décision.** Un arbre de décision est un arbre tel qu'il est défini au sens mathématique, c'est-à-dire un graphe acyclique constitué de nœuds et d'arêtes. La phase d'apprentissage sert à déterminer une séquence de nœuds. Chaque nœud contient un test sur une caractéristique x_i d'un exemple X. Par exemple, est-ce que l'exigence X contient plusieurs points de fin de phrases - oui ou non ? Le nœud est alors divisé en autant de valeurs que peut prendre la caractéristique x_i. Dans notre exemple, il n'y a que deux valeurs possibles, oui ou non, soit deux divisions qui donnent naissance à deux nœuds fils. La procédure est répétée pour chaque nœud. Chacune des feuilles correspond à la valeur de la classe y associée à l'exemple X. En lisant l'arbre on peut extraire un ensemble de règles du type SI ... ALORS ..., SINON SI ... ALORS ..., SINON ... .

– **Machine à vecteur support (SVM).** Un espace à p-dimension peut être séparé par un hyperplan de dimension p - 1 et d'équation $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p = 0$. Par exemple, un espace à deux dimensions peut être séparé par une droite d'équation $\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$ (Fig. 4.12 A). Si un vecteur de caractéristiques $X = \{x_1, x_2, ..., x_n\}$ ne satisfait pas cette équation, alors X appartient d'un côté ou de d'autre de l'hyperplan selon le signe de la partie à gauche de l'égalité. Autrement dit, il appartient à une catégorie ou à l'autre (*spam* vs. *non spam*, par exemple). La figure 4.12 (B) montre qu'il existe une infinité d'hyperplans pour séparer l'espace engendré par les caractéristiques (*feature space*, en anglais). On peut néanmoins s'appuyer sur certains exemples – les vecteurs supports – pour trouver un plan unique qui maximise la frontière de décision (Fig. 4.12 C). C'est le principe de l'algorithme « *Maximal Margin Classifier* ». Ce type d'algorithme s'applique à des exemples qui sont strictement séparables par un hyperplan. Dans les autres cas, l'exercice de recherche d'un optimum est sans solution. Une première évolution a donc été de définir le concept de marge faible (*soft margin*, en anglais) qui fait qu'un hyperplan n'est plus contraint de strictement séparer les exemples. Une tolérance d'erreur paramétrable est autorisée. Cette flexibilité permet de séparer des données qui étaient originellement non séparables moyennant quelques erreurs de classification (Fig. 4.12 D). Ce type d'algorithme est appelé « *Support Vector Classifier* ». C'est la solution standard pour un problème de classification binaire linéairement séparable. Cependant, parfois, la solution est non linéaire (Fig. 4.12 E et F). L'algorithme de « *Support Vector Classifier* » a donc, à son tour, été amélioré pour donner naissance aux machines à vecteurs supports (*Support Vector Machine* (SVM), en anglais). L'algorithme SVM utilise des fonctions dites noyaux (*kernel*, en anglais) pour étendre l'espace engendré par les caractéristiques afin de pouvoir séparer les exemples avec une ou plusieurs frontières. Ces fonctions noyaux sont généralement non linéaires telles que les fonctions polynomiales (Fig. 4.12 E) ou radiales (Fig. 4.12 F). Par abus de langage, il est fréquent de lire ou d'entendre parler de machines à vecteurs supports pour référer à ces trois algorithmes.

– **Réseau de neurones.** Il y a une multitude de variantes des réseaux de neurones. Le modèle qui est à l'origine de tous, le Perceptron, utilise aussi un hyperplan comme frontière de décision d'un problème de classification binaire linéairement séparable. Ce qui change par rapport au Support Vector Classifier, c'est la méthode de construction de l'hyperplan. Au départ de l'apprentissage, tous les poids a_i de l'équation de l'hyperplan sont initialisés à 0 tel que $w_0 a_0 + w_1 a_1 + w_2 a_2 + ... + w_k a_k = 0$. Si la somme est supérieure ou égale à 0, alors la classe 1 est prévue, sinon la classe 2 est prévue. La recherche des poids optimums se fait en itérant sur l'ensemble des exemples du corpus d'apprentissage. Pour chaque exemple i, si l'exemple est bien classifié, alors on ne change pas la valeur des poids, sinon les poids sont modifiés de sorte à ce que l'exemple mal classifié se rapproche de l'hyperplan ou traverse la frontière pour se retrouver du bon côté. Ainsi, si l'exemple appartient à la classe 1, alors la valeur de chaque caractéristiques a_i est ajoutée à chaque poids w_i, sinon la valeur est soustraite. Par exemple, si un exemple a appartenant à la classe 1 est mis à jour tel que $(w_0 + a_0)a_0 + (w_1 + a_1)a_1 + (w_2 + a_2)a_2 + ... + (w_k + a_k)a_k$, alors la sortie pour a a augmenté de $a_0 * a_0 + a_1 * a_1 + xa_2 * a_2 + ... + a_k * a_k$. Cette valeur est toujours

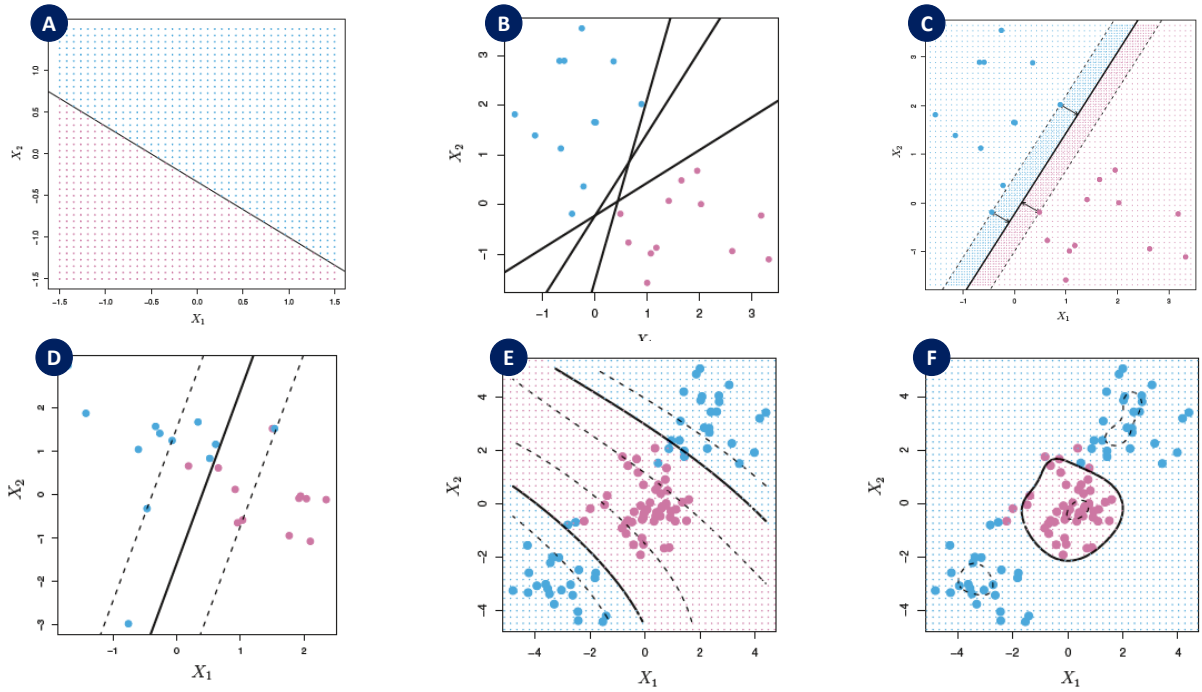


FIGURE 4.12 – Évolution des algorithmes qui ont mené à la définition des machines à vecteurs support (adapté de James et al. [2013]).

positive, ainsi l'hyperplan a bougé dans la bonne direction pour classifier l'exemple a comme positive. Inversement, si un exemple appartenant à la classe 2 est mal classifié, alors la sortie de l'exemple diminue après la mise à jour, ce qui fait que l'hyperplan se déplace toujours dans la bonne direction. En itérant sur tous les exemples, si le problème est linéairement séparable, alors l'algorithme doit converger en un nombre fini d'itérations.

Sur la base des expérimentations détaillées dans la section 4.4, on peut sélectionner l'algorithme d'apprentissage qui a obtenu les meilleurs performances. Cet algorithme a alors été directement implémenté et entraîné dans notre environnement numérique via l'API de Weka. Ainsi, les exigences extraites des documents prescriptifs sont classifiées automatiquement. Comme le montre la figure 4.13, les exigences qui ont été classifiées comme contenant au moins une référence transversale sont listées dans un tableau. L'analyste doit traiter les exigences une par une. Si l'exigence est un résultat faux positif, c'est à dire qu'elle ne contient pas de référence transversale, alors il appuie sur le bouton « *Remove* » pour l'exclure de la liste. Si l'exigence contient bien une référence vers un document externe et que ce document a déjà été traité, alors l'analyste le sélectionne via la liste déroulante et clique sur le bouton « *Process* » situé en face de l'exigence. Cette action lie l'exigence contenant la référence transversale avec les exigences prescrites par le document référencé. Techniquement, cela consiste à interroger la base de données pour : (1) récupérer le sommet « Spécification » correspondant au document référencé ; (2) récupérer les sommets « Exigence » correspondants aux exigences de ce document ; (3) récupérer l'exigence contenant la référence transversale ; (4) créer de nouvelle relations « Réfère » liant l'exigence qui fait référence au document avec les exigences prescrites par le document référencé. Il se peut que le document prescriptif référencé n'ait jamais été traité auparavant. Le cas échéant, il n'est pas listé dans la liste déroulante située au dessus du tableau. L'analyste doit alors soumettre le document à l'environnement numérique avant de pourvoir le sélectionner et cliquer sur le bouton « *Process* » permettant de lier les exigences.

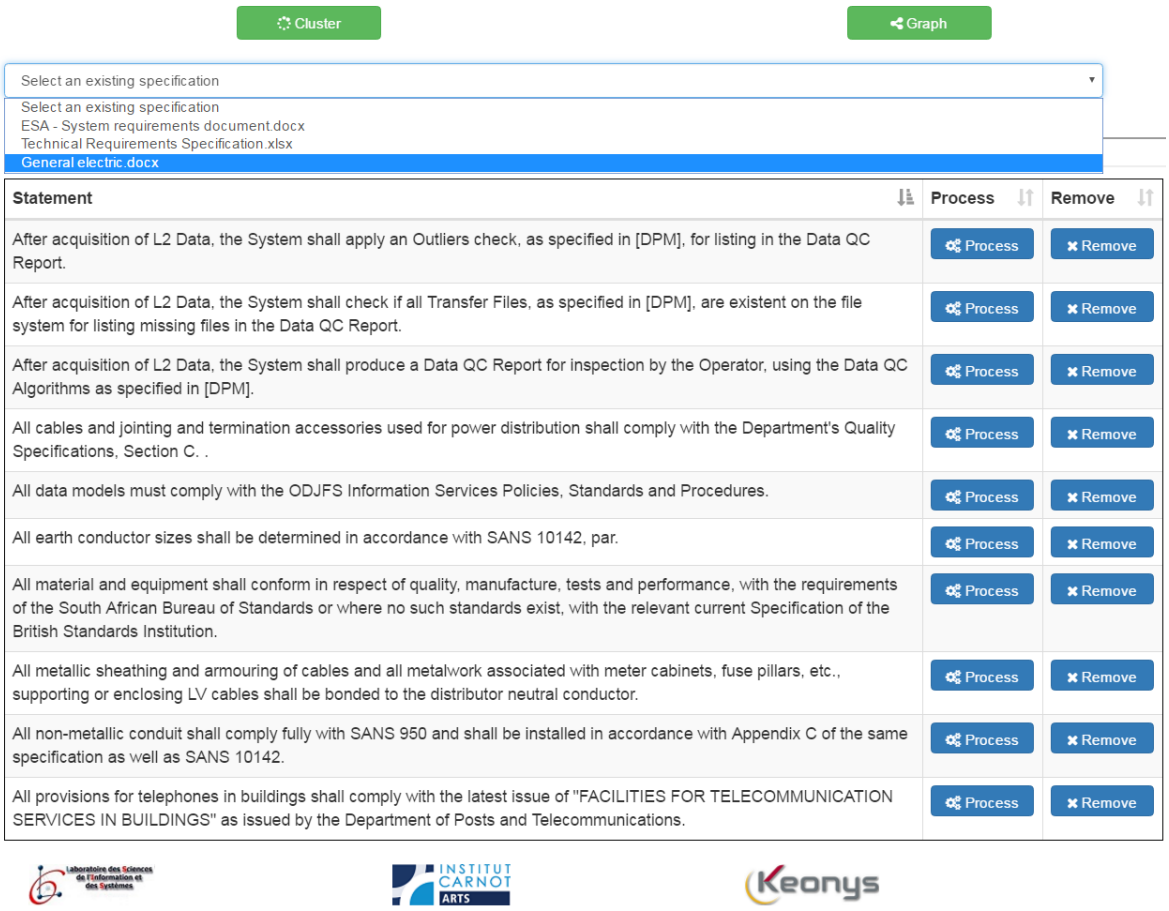


FIGURE 4.13 – Interface qui liste les exigences qui pontentiellement contiennent des références transversales. L'utilisateur peut alors créer les relations entre l'exigence qui fait référence à un document applicable et les exigences prescrites dans le document applicable.

4.4 Expérimentation

La vérification du module d'extraction des exigences se décompose en deux parties. D'une part il y a la vérification du chaînage numérique permettant l'extraction des exigences à partir de documents prescriptifs non structurés et semi-structurés. D'autre part, il faut vérifier que la fonction de classification basée sur l'apprentissage machine identifie bien les exigences qui contiennent des références transversales vers des documents externes applicables.

Expérimentation 1 - Vérification de l'extraction des exigences

Deux documents de spécification issus du milieu industriel sont utilisés pour vérifier le chaînage numérique qui extrait les exigences de documents prescriptifs.

La première spécification est un document de 106 pages disponible sur le site internet de l'Agence Spatiale Européenne. Ce document est au format PDF et a été originellement édité avec MS Word. Une revue manuelle a permis d'identifier 194 exigences qui spécifient un système d'informations pour observer des données mesurées par des capteurs.

La seconde spécification est un document de 74 pages disponible sur le site internet du ministère des travaux publics de l'état Sud-Africain. Ce document est au format MS Word et spécifie un ensemble d'exigences techniques générales pour les équipements, l'installation, les tests, la livraison et la maintenance d'installations électriques. Bien qu'il soit plus petit en nombre de pages, le document prescrit 10 fois plus d'exigences. En effet, la lecture de la spécification a permis d'en comptabiliser 1066.

L'évaluation d'un modèle de classification repose sur le calcul de quatre critères de performance : le pourcentage de classifications correctes, la précision, le rappel et la F-mesure. Pour calculer ces critères, il faut collectionner les entrées d'une table de contingence (Tab. 4.2) qui synthétise les résultats :

– « vrais positifs » (VP) les objets pertinents retournés,
– « faux positifs » (FP) les objets non pertinents retournés,
– « faux négatifs » (FN) les objets pertinents non retournés, et
– « vrais négatifs » (VN) les objets non pertinents non retournés.

	Objets pertinents	**¬ Objets pertinents**
Objets retournés	VP	FP
¬ Objets retournés	FN	VN

TABLE 4.2 – Table de contingence.

Le pourcentage de classifications correctes est alors égal à $A = \frac{VP+VN}{VP+FP+FN+VN}$.

La précision P est la proportion d'objets pertinents parmi ceux qui sont retournés : $P = \frac{VP}{VP+FP}$.

Le rappel R est la proportion d'objets pertinents retournés parmi tous les objets pertinents : $R = \frac{VP}{VP+FN}$.

On peut aussi étudier l'équilibre entre la précision et le rappel. Il est facile d'obtenir un rappel de 1 en retournant tous les objets. En faisant une moyenne arithmétique on obtiendrait alors un score minimum de 0.5 en ayant une précision de 0 et un rappel de 1. Pour éviter ce genre de biais, on calcule une moyenne harmonique F_β qui combine la précision et le rappel tel que $F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P+R}$. On utilise généralement $\beta = 1$, soit $F_1 = \frac{2PR}{P+R}$. Ainsi, avec ce critère, si la précision ou le rappel est nul, alors F_1 est aussi nulle. Par ailleurs, pour que la F_1-mesure soit égale à 1 il faut que $P = R = 1$.

	Exigences	**Informations**
Énoncés classifiés comme des exigences	194	72
Énoncés classifiés comme des informations	0	?

TABLE 4.3 – Table de contingence pour le premier document prescriptif.

Les résultats obtenus lors de l'extraction automatique des exigences avec l'environnement numérique sont résumés dans les tables de contingence 4.3 et 4.4. Le lecteur remarquera que les résultats vrais négatifs, en bas à gauche, n'ont pas été comptabilisés. Si nous connaissions le nombre exact de phrases dans chaque document, alors nous pourrions calculer ces valeurs directement. Or, même en se basant sur les résultats obtenus par l'opération de segmentation du texte en phrases, il est très probable que le résultat ne soit pas 100 % précis. L'unique solution est donc de comptabiliser tous les énoncés informatifs, c'est-à-dire toutes les phrases qui ne sont pas des exigences. Cela requiert un temps considérable pour très peu d'informations complémentaires, si ce n'est le pourcentage de classifications correctes.

Selon la table 4.3, dans le premier document, le prototype a identifié la totalité des 194 exigences, ce qui résulte en un rappel de 1. Cependant, 72 énoncés informatifs ont été considérés comme des exigences, soit une précision de 0.73. Ce résultat est très satisfaisant car l'objectif est de collecter toutes les exigences.

Dans le second document, 1069 exigences ont été extraites sur un total de 1174. Parmi ces 1069 exigences il y a seulement 3 résultats faux positifs, soit une excellente précision de 0.99. Le prototype a néanmoins manqué 10 % des exigences, 108 exactement, ce qui permet d'obtenir un bon rappel de 0.9 étant donné que le nombre total d'exigences est important.

Si l'on fait la moyenne sur les deux documents, la précision et le rappel sont respectivement égal à 0.86 et 0.95. Même si la définition du concept d'exigence varie d'une recherche à l'autre, ces résultats sont nettement supérieur à ceux de l'état de l'art, et ce, avec un volume de donnés qui est beaucoup plus important.

	Exigences	Informations
Énoncés classifiés comme des exigences	1066	3
Énoncés classifiés comme des informations	108	?

TABLE 4.4 – Table de contingence pour le second document prescriptif.

Expérimentation 2 - Vérification de l'identification des références externes

Comme précédemment, la vérification de la fonction de classification qui identifie les exigences contenant des références vers des documents externes applicables nécessite de calculer des critères de performance pour chacun des algorithmes de classification préférés.

Les algorithmes d'apprentissage supervisé utilisés pour résoudre le problème de classification sont pré-implémentés dans l'outil Weka. Weka est un outil libre développé par l'université de Waikato qui met à disposition de nombreux algorithmes d'apprentissage supervisés et non supervisés. Il peut être utilisé indépendamment ou embarqué dans une application grâce aux APIs disponibles. Weka est préféré aux autres solutions plus populaires telles que R ou Python Scikit-Learn, car c'est une technologie développée en Java que l'on peut embarquer directement dans notre environnement numérique.

La construction d'une fonction de classification nécessite de faire diverses expériences pour tester plusieurs algorithmes d'apprentissage. Cette phase d'exploration a donc été réalisée avec l'outil Weka indépendamment de l'environnement numérique de synthèse des exigences. Les algorithmes les plus populaires précédemment introduits – modèle bayésien, régression logistique, machine à vecteurs supports et réseau de neurones – sont mis en œuvre par les implémentations fournies par Weka : Naïve Bayes (NB), J48, Logistic (LR), Sequential Minimal Optimization (SMO), et Voted Perceptron (VP).

Pour apprendre une fonction de classification, tous les modèles s'entraînent sur un corpus d'apprentissage de 500 exemples dont 270 exigences contenant au moins une référence transversale et 230 sans référence transversale. Ce corpus a été manuellement construit en collectant des exigences dans des spécifications industrielles et des normes diverses, puis en annotant chacune des exigences.

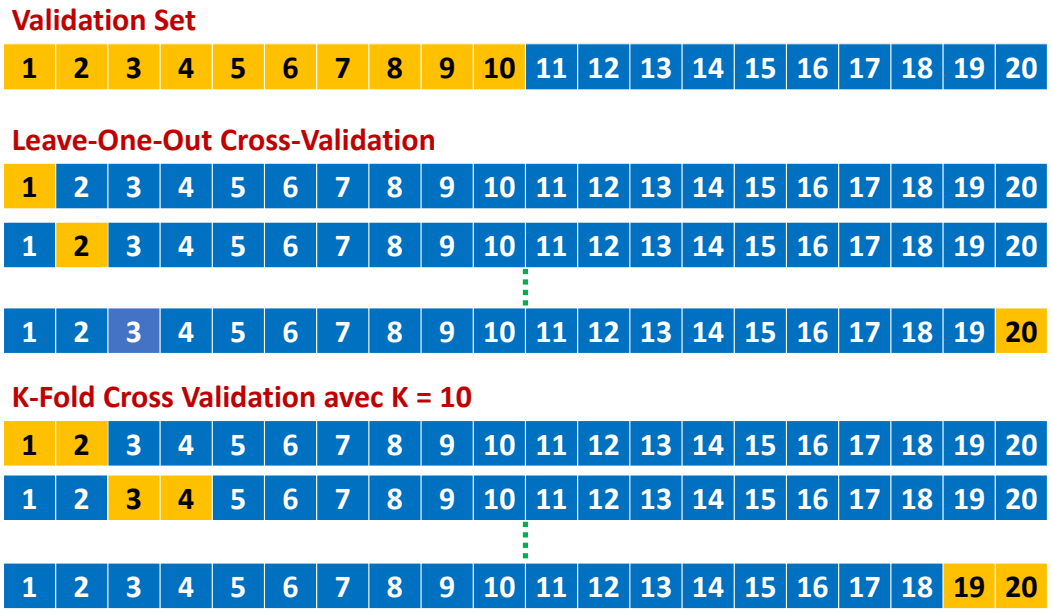


FIGURE 4.14 – Les trois méthodes de validation d'une fonction de classification.

Pour tester une fonction de classification, c'est-à-dire pour évaluer des critères de performance, il y a trois grandes méthodes (Fig. 4.14). La première consiste à diviser le corpus initial en deux sous-corpus : un corpus d'apprentissage et un corpus de test. L'apprentissage et l'évaluation se font alors sur deux corpus indépendants. Cette approche a deux désavantages. D'une part, les résultats peuvent varier

significativement selon le découpage. D'autre part, cette méthode de validation est adaptée quand on dispose de suffisamment d'exemples. Le nombre d'exemples nécessaires varie selon le problème. Dans un problème de classification binaire, une centaine d'exemples peut suffire, tandis qu'un problème de classification muti-classes peut nécessiter plusieurs milliers d'exemples. La seule règle non réfutable c'est que plus il y a d'exemples disponibles, meilleures (plus robustes) sont les performances !

Quand le nombre d'exemples est insuffisant, il faut s'assurer que le phénomène de sur-apprentissage (*overfitting*, en anglais) ne vienne pas fausser les résultats. En effet, les algorithmes ont tendance à mieux, voire trop apprendre quand il y a peu d'exemples. On dit qu'une fonction d'apprentissage est sur-apprise quand elle a obtenu de bonnes performances sur le corpus d'apprentissage pendant la phase d'entraînement, mais que ses performances se dégradent en phase de tests ou d'opération sur de nouvelles données inconnues. Autrement dit, en entraînant un algorithme d'apprentissage sur un ensemble d'exemples $\{(X_1, y_1), (X_2, y_2), ..., (X_n, y_n)\}$, on obtient une fonction de classification $\hat{f}$. On peut alors calculer $\hat{f}(X_1), \hat{f}(X_2), ..., \hat{f}(X_n)$ pour prévoir les réponses $\hat{y}_1, \hat{y}_2, ..., \hat{y}_n$. Si $\hat{y}_1 \approx y_1, \hat{y}_2 \approx y_2, ..., \hat{y}_n \approx y_n$, alors l'erreur d'apprentissage est minime, mais cela ne garantit pas que $\hat{f}(X_i) \approx \hat{y}_i$ pour un nouvel exemple X_i. D'un point de vue mathématique, l'optimisation des paramètres de l'algorithme d'apprentissage est trop dépendante des exemples. La figure 4.15 illustre différentes fonctions de classification apprises à partir d'algorithmes plus ou moins flexibles, c'est-à-dire capable d'approximer des formes plus ou moins variées. Par exemple, une fonction polynomiale de degrés 3 est beaucoup plus flexible qu'une fonction linéaire. Ainsi, si des tests montrent que l'erreur de classification tend à diminuer significativement avec une fonction moins complexe – une fonction linéaire, par exemple –, alors il est très probable que la fonction de classification fut initialement sur-apprise.

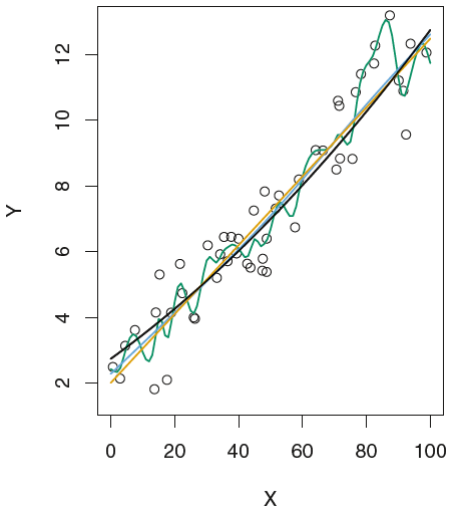


FIGURE 4.15 – 4 fonctions de classification estimées avec des algorithmes d'apprentissage de flexibilité différente. [James et al., 2013].

La validation croisée (*cross-validation*, en anglais) permet d'éviter ce phénomène. La première approche de validation croisée « *Leave-One-Out-Cross-Validation* » (LOOCV) consiste aussi à séparer le corpus d'apprentissage en deux. Cependant, ici, les deux corpus ne sont pas de taille comparable. Parmi les n exemples initiaux, un seul exemple est mis à l'écart. L'algorithme d'apprentissage apprend sur les $n - 1$ exemples restant avant de prévoir la réponse de l'exemple écarté. Le processus est répété n fois en écartant les exemples les uns après les autres. Les résultats obtenus avec LOOCV sont beaucoup plus robustes que ceux obtenus avec la méthode précédente parce que les performances ne sont pas dépendantes de la division du corpus initial. La méthode LOOCV est néanmoins onéreuse car il faut estimer n fonctions de classification. La troisième approche de validation « K*-Fold Cross-Validation* » est un compromis entre les deux. Le corps initial est divisé en K sous-corpus. Un corpus sert à estimer une fonction de classification, tandis que les K-1 autres servent à évaluer les performance. Comme avec LOOCV, le processus est répété K fois. La valeur k = 10, soit 10-Fold Cross Validation est habituellement préférée

pour obtenir des résultats suffisamment robustes avec un nombre acceptables d'apprentissages.

Étant donné que notre corpus initial contient un nombre limité mais acceptable de 500 exemples, la méthode 10-Fold Cross Validation est préférée pour évaluer les performances des fonctions de classification. Les résultats obtenus sont présentés dans la table 4.5. Premièrement, on peut constater que toutes les fonctions obtiennent des résultats très satisfaisants avec un pourcentage d'exemples bien classifiés compris entre 90 +/- 1.5%. Autrement dit, en moyenne, la fonction de classification se trompe seulement 1 fois sur 10. Par ailleurs, les scores de rappel et de précision sont tous équilibrés ce qui se reflète dans les valeurs de la F-mesure. Cela montre que la précision n'a pas été dopée au détriment du rappel ou *vice-versa*. Dans cet exercice de classification, c'est l'algorithme de machine à vecteurs supports (SMO) qui a obtenu les meilleurs résultats. Par défaut, SMO applique une fonction linéaire, ce qui est adapté pour les problèmes de classification binaire. En plus de la validation croisée, pour vérifier que la fonction de classification n'est pas sur-apprise, l'algorithme SMO a été entraîné en utilisant une fonction noyau polynomiale d'ordre 2 et 3. Le pourcentage d'exigences bien classifiées est respectivement égal à 92 % et 91.4 %. Il est donc peu probable que la fonction de classification soit sur-estimée, car les résultats obtenus avec la fonction linéaire ne sont pas significativement plus faibles que ceux obtenus avec les deux fonctions polynomiales. Par ailleurs, l'utilisation de fonctions plus flexibles n'améliore pas significativement les performances, les données sont donc linéairement séparables et ne nécessitent pas de modèle plus sophistiqué.

	NB	**J48**	**LR**	**SMO**	**VP**
Observations bien classifiées (%)	88.6	90.8	90.6	91.2	89.4
Taux de vrai-positif	0.886	0.908	0.906	0.912	0.894
Taux de faux-positif	0.125	0.093	0.097	0.089	0.112
Précision	0.890	0.908	0.906	0.912	0.895
Rappel	0.886	0.908	0.906	0.912	0.894
F-mesure	0.885	0.908	0.906	0.912	0.894
MCC	0.774	0.815	0.811	0.823	0.787
Surface ROC	0.968	0.925	0.970	0.911	0.921

TABLE 4.5 – Critères de performance obtenus pour chaque algorithme d'apprentissage.

4.5 Synthèse

Bilan de notre méthode d'extraction des exigences

Dans ce chapitre, nous cherchons une méthode qui puisse permettre à un analyste : (1) d'extraire les exigences de documents prescriptifs structurés et semi-structurés, et (2) identifier les exigences qui font référence à un document externe potentiellement prescriptif et applicable.

Dans un premier temps, la méthode consiste à identifier le format du document afin d'utiliser le parseur adéquat. Les exigences prescrites par des documents semi-structurées ReqIF ou SysML sont extraites avec un parseur XML. Les documents non structurés, eux, sont convertis au format HTML afin d'analyser leur structure. Le texte des documents non structurés est ensuite envoyé dans un processus de traitement du langage naturel pour le segmenter en phrases. Parmi ces phrases, les exigences sont distinguées grâce à une liste de verbes prescriptifs. Enfin, une fonction de classification estimée au moyen des techniques d'apprentissage machine permet d'identifier les exigences qui font référence à des documents externes qui sont potentiellement prescriptifs et applicables. L'analyste peut alors soumettre les documents prescriptifs référencés afin d'extraire les exigences et d'établir des références transversales entre l'exigence référence et les exigences référencées. La figure 4.16 est un diagramme de flux d'informations qui résume la méthode que nous proposons pour extraire les exigences.

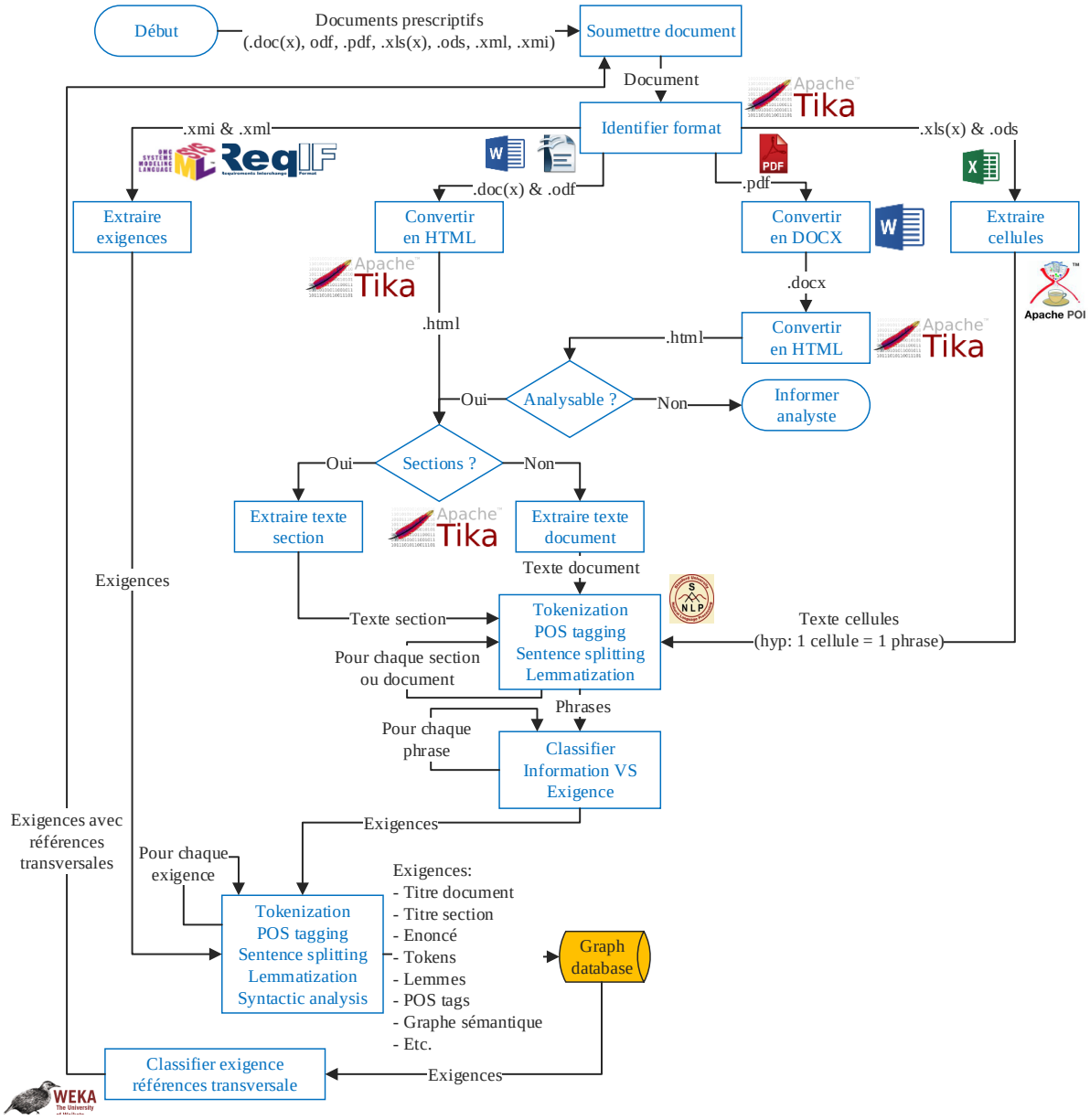


FIGURE 4.16 – Diagramme de flux d'informations qui résume la méthode d'extraction des exigences.

Deux expérimentations ont été utilisées pour tester ces deux propositions. La première a permis de montrer que la méthode d'extraction des exigences était capable d'extraire les exigences avec un rappel et une précision moyenne de 0.95 et 0.86. La fonction de classification automatique permettant d'identifier les exigences contenant une référence vers un document externe, elle, a réalisé 91.2 % de classifications correctes sur un corpus de 500 exemples et une validation croisée en 10 plis.

Qu'est-ce qui est nouveau ou différent ?

L'état de l'art a montré que les fonctions techniques étudiées dans cette première partie avaient été plus ou moins étudiées auparavant.

Néanmoins, pour la première fonction d'extraction des exigences, les travaux existants sont superficiels. Les outils utilisés et les problèmes liés aux formats des documents ne sont par exemple jamais discutés. Ici, le passage vers un format HTML nous permet de mieux appréhender l'extraction des phrases en considérant la structure du document. Par ailleurs, les documents utilisés pour tester sont habituelle-

ment courts – une dizaine d'exigences –, en nombre très limité et rarement mis à disposition des autres chercheurs. Ici, nous proposons un premier cadre bien formaté avec les outils utilisés, les formats supportés, et un libre accès aux documents prescriptifs sur lesquels le prototype a été testé. Par ailleurs, nous ne nous limitons pas à des définitions strictes – présence d'un verbe modal ou légal, structure syntaxique particulière – du concept d'exigence.

Ce qui est relativement nouveau, c'est la mise en œuvre de méthodes d'apprentissage supervisé pour identifier les exigences qui font référence à des documents externes. En effet, les solutions existantes sont principalement basées sur des règles de connaissance limitées aux textes légaux et règlementaires. Ici, nous proposons un ensemble de caractéristiques utiles à l'identification des références transversales, ainsi qu'une comparaison des performances obtenues avec les principaux algorithmes d'apprentissage supervisé. Par ailleurs, un corpus d'entrainement constitué de 500 exemples a été construit manuellement. Pour des raisons de propriété intellectuelle et de compétitivité de notre partenaire industriel, ce corpus n'est actuellement pas en libre accès, mais pourra potentiellement être partagé sous certaines contraintes d'exploitation.

Quelles sont les limites et les perspectives d'amélioration ?

Les résultats faux-positifs obtenus dans l'analyse du premier document sont principalement dues au fait que la première moitié du texte est constituée de sections informatives, lesquelles contiennent quelques termes prescriptifs qui n'ont rien à voir avec les exigences du produit. Bien que certains chapitres soient ignorés sur la base de mots clés (*introduction*, *scope*, etc.), ces derniers ne couvrent pas toute l'étendue des possibilités offertes par la flexibilité du langage naturel. Par ailleurs, si les chapitres sont rédigés sans utiliser les fonctionnalités de Word, alors ils ne sont pas identifiables car il n'y a pas les balises HTML correspondantes. Néanmoins, habituellement, les premiers énoncés d'un document sont des informations décrivant le cadre d'application. La position de chaque exigence par rapport au début du document pourrait donc permettre de calculer un critère de confiance, c'est-à-dire une probabilité que l'énoncé est informatif ou prescriptif selon sa position dans le texte. Ce critère de confiance pourrait aussi être utile pour signaler les résultats qui sont très probablement des faux positifs dont l'analyse nécessite ne intervention de l'analyste. La définition d'un seuil de confiance nécessite de mener une étude sur de nombreux documents prescriptifs afin de calculer une distribution statistique pour déterminer, par exemple, le nombre moyen de phrases qui précèdent la première exigence. En dernier recours, si cette approche n'est pas pertinente, avant le traitement automatique, l'analyste pourrait ajouter un tag textuel ou graphique sur le document pour distinguer les sections informatives et descriptives.

L'utilisation de listes pour énumérer des conditions ou des prescriptions est la principale raison qui fait que certaines exigences passent à travers le modèle de classification. En effet, si les listes ne sont pas définies avec les fonctionnalités proposées par Word, alors ce sont de simples caractères tels que des espaces, des lettres, des chiffres arabes ou romains, des parenthèses, des crochets, des points, et des retours à la ligne. Le cas échéant, il n'y a pas de balises HTML permettant d'identifier les listes. Chaque énumération est alors concaténée dans une grande chaîne de caractères avec le texte environnant. Néanmoins, aujourd'hui, les nouveaux éditeurs de texte automatisent la construction d'une liste dès que l'utilisateur initie une énumération. On peut donc espérer que cette limite devienne obsolète avec les nouveaux traitements de texte.

Certaines exigences ne sont pas que textuelles, mais font référence à des graphiques, des croquis, des tableaux, etc. À ce jour, notre méthode d'extraction des exigences se limite à la partie textuelle, une minorité d'entre elles sont donc incomplètes. Il est possible d'extraire les images et de faire l'hypothèse que celle-ci réfère à l'exigence qui la précède, mais ceci n'est qu'une supposition dérivée d'un constat : on rédige l'exigence et on ajoute les éléments complémentaires en dessous.

Concernant l'identification des exigences qui font référence à des documents externes, il n'y a pas de limite majeure. Il serait néanmoins utile d'ajouter une étape de sélection des caractéristiques (*features selection*, en anglais) précédent la phase d'apprentissage. La sélection de caractéristiques exclut les caractéristiques redondantes ou impertinentes ou les deux. Travailler avec un sous-ensemble de caractéristiques non redondantes et pertinentes peut améliorer les performance de la fonction de classification

et diminuer le temps d'entraînement. Ici, les performances sont très bonnes et le temps d'entraînement est dérisoire (moins d'une minute). L'ensemble des caractéristiques a donc été conservé, mais cela ne signifie pas que l'étude soit sans intérêt.

Chapitre 5

FIABILISER LES EXIGENCES

5.1 Introduction

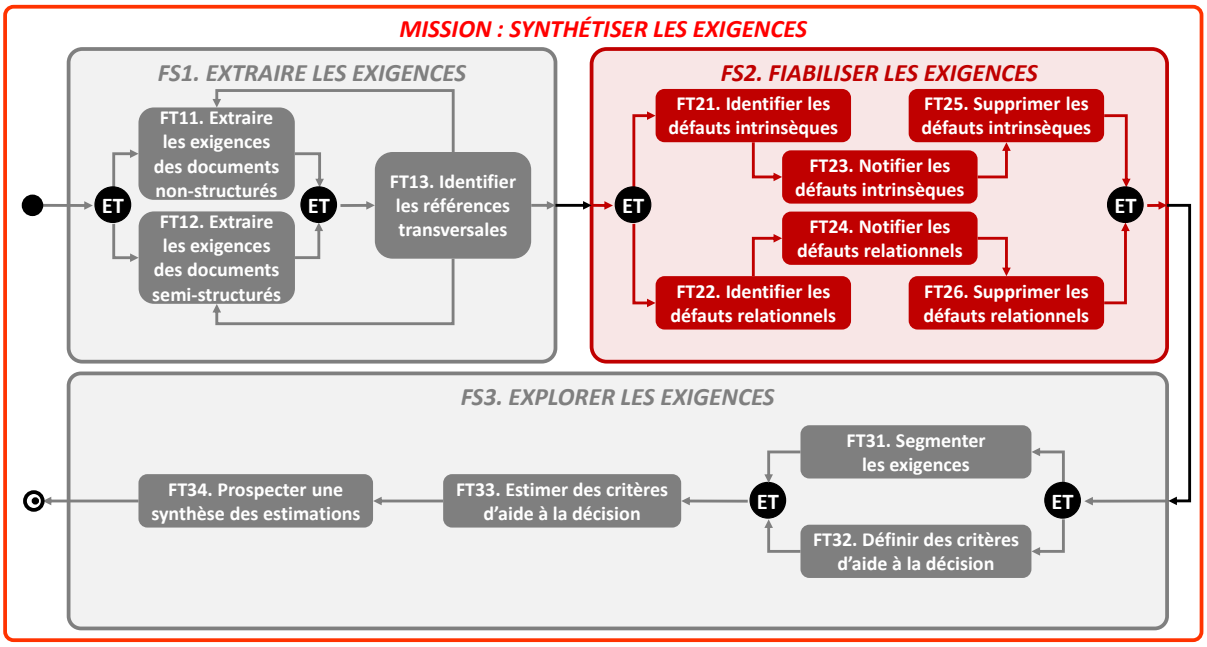


FIGURE 5.1 – FS2 - Fiabiliser les exigences.

Les exigences extraites des documents prescriptifs sont ambiguës. Un énoncé est ambigu s'il peut être interprété différemment par un ou plusieurs agents. En ingénierie des exigences, ces ambiguïtés ont des effets néfastes sur le reste du cycle de vie du produit. Elles peuvent par exemple mener à l'insatisfaction du client, à la non certification du produit, ou à des dépenses imprévues.

Dans notre cas, nous cherchons à minimiser le champ des interprétations afin que les estimations ultérieures des critères d'aide à la décision soient objectives. Nous insistons sur le fait que notre but est de fiabiliser l'interprétation d'une exigence et non pas de la rendre non ambiguë, et ce, parce que l'ambiguïté est inévitable pour diverses raisons qui seront discutées dans l'état de l'art.

Dans la suite de ce chapitre, la section 5.2 est un état de l'art des différentes approches potentiellement utiles pour fiabiliser l'interprétation d'une exigence.

Dans la section 5.3 nous proposons un diagramme d'influence qui synthétise les critères susceptibles d'influencer le degré d'ambigüité d'une exigence. Nous proposons alors un ensemble de règles qui, au moyen des techniques de traitement du langage naturel, permettent d'identifier les défauts de qualité intra- et inter-exigences. Enfin, nous proposons diverses représentations graphiques interactives pour

nettoyer les défauts de qualité d'une exigence.

La section 5.4 présente des résultats expérimentaux menés sur des documents de spécifications issus de l'industrie.

Enfin, la section 5.5 synthétise les éléments nouveaux et différents, ainsi que les limites de nos propositions.

5.2 État de l'art des solutions de fiabilisation des exigences

Cet état de l'art débute par une discussion autour du concept d'ambiguïté en ingénierie des exigences, ainsi que des modèles proposés pour évaluer la qualité d'une exigence et d'un ensemble d'exigences. Les autres parties synthétisent et critiquent les différentes solutions existantes qui contribuent à la fiabilisation de l'interprétation des exigences.

Des exigences inéluctablement ambiguës

Il existe diverses raisons qui font qu'une exigence est ambiguë. La raison la plus évidente trouve sa source dans une utilisation abondante du langage que nous utilisons quotidiennement pour communiquer à l'oral et à l'écrit : le langage naturel. Mich et al. [2004] ont estimé que 79% des spécifications dans le génie logiciel sont rédigées au moyen du langage naturel. Les ambiguïtés linguistiques sont lexicales, syntaxiques, ou sémantiques selon qu'on s'intéresse aux mots, à l'ordre des mots, ou au signifié.

Par ailleurs, un énoncé peut être non ambigu d'un point de vue linguistique, sans pour autant l'être d'un point de vue pragmatique [Berry et al., 2003]. En effet, même si les règles linguistiques sont rigoureusement respectées par l'émetteur, il se peut que les lacunes, linguistiques ou non, d'un récepteur fassent naître le doute quant à l'intention initiale de l'émetteur. Par exemple, l'ambiguïté est un phénomène qui est relatif à l'état des connaissances d'un récepteur. Si le récepteur est un expert sur le sujet, alors le champ des interprétations possibles sera restreint via l'exclusion naturelle des interprétations insensées. Pour illustrer ces propos, il suffit d'imaginer un concepteur en mécanique qui lit et interprète le jargon d'une exigence légale extraite d'un texte de loi et *vice-versa*.

Berry et al. [2003] nous indiquent aussi qu'un énoncé peut être volontairement ou involontairement ambigu. Effectivement, hormis le fait que l'ambigüité d'un énoncé soit, en règle générale, le résultat d'un acte inconscient, aussi surprenant que cela puisse paraître, une partie prenante peut volontairement spécifier des exigences ambiguës. Par exemple, les textes réglementaires sont volontairement ambigus afin que la décision finale de certification du produit demeure la propriété de l'autorité de régulation. Une telle pratique permet d'élargir le cadre d'application d'un texte règlementaire en s'affranchissant des particularités propres à chacune des industries ou législations. Quand cette situation se présente, le fournisseur initie un processus d'interprétation en deux temps. D'abord, il identifie les exigences règlementaires applicables. Ensuite, pour chaque exigence, il propose une interprétation à l'organisme de certification. Les grands donneurs d'ordres de l'industrie aéronautique soumettent leurs interprétations à l'European Aviation Sagety Agency (EASA). C'est également le cas dans l'industrie de l'énergie nucléaire [Sannier and Baudry, 2012].

Enfin, pour conclure, comme la récemment exposé Kasser [2012] au sommet de l'INCOSE : le manque de formation est l'une des raisons fondamentales qui fait que les exigences sont de mauvaise qualité. La plupart des analystes ne savent pas rédiger des exigences et cela se reflète dans la qualité des documents prescriptifs. Les exigences sont souvent incomplètes, parfois non singulières, elles contiennent des qualificatifs non mesurables et spécifient la solution plutôt que le besoin.

En partant du fait qu'une interprétation multiple est une condition suffisante pour faire naître l'ambiguïté, on peut conclure qu'à plus grande échelle, celle du document et du corpus, l'ambiguïté est inévitable [Berry et al., 2003]. Un document contiendra toujours quelques énoncés dont l'interprétation sera au moins double. Il ne faut donc pas se tromper d'objectif. L'objectif n'est pas d'éradiquer tous les signes qui caractérisent l'ambiguïté, mais de fiabiliser l'interprétation des exigences.

Une prolifération de modèles de qualité

L'ambiguïté est l'un des critères que l'on retrouve dans la plupart des modèles de qualité. En ingénierie des exigences, un modèle de qualité est un ensemble de caractéristiques qui permettent d'évaluer la qualité d'une ou plusieurs exigences. Par exemple, une exigence doit être nécessaire, complète, singulière, faisable, vérifiable, conforme, etc., tandis qu'un ensemble d'exigences doit être complet, consistant, faisable, compréhensible, validable, etc.

Il existe presque autant de modèles de qualité que de groupes de recherche en ingénierie des exigences. Parmi les plus populaires, on trouve ceux de Davis et al. [1993], Loucopoulos and Karakostas [1995], Wilson et al. [1997], Fabbrini et al. [2001], Swathi et al. [2011], Génova et al. [2013], Wiegers and Beatty [2013], Klaus Pohl [2015], Thitisathienkul and Prompoon [2015] et Fraga et al. [2015].

Cette prolifération de modèles de qualité dont les critères sont parfois mutuellement contradictoires ou redondants, souvent subjectifs et ambigus, engendrent le doute quant à leur valeur scientifique. En effet, pour se distinguer les uns des autres, tous les modèles, standardisés ou non, s'efforcent à, d'une part, définir des caractéristiques de plus en plus fines et, d'autre part, à les catégoriser via la définition d'une typologie. Ainsi, avec la multiplication des critères, les typologies s'apparentent plus à des catégories qui s'emboîtent comme des poupées russes qu'à une typologie rigoureuse [1]. Or, plus on raffine la définition des catégories, plus on trouve de contre-exemples permettant de réfuter (au sens de Karl Popper [1934]) certains de ces modèles. Saavedra et al. [2013] ont récemment pointé du doigt cette faiblesse lorsqu'ils ont mis en évidence le fait que tous les modèles de données ont oublié ou négligé l'influence, positive ou négative, que peut avoir une caractéristique de qualité sur une autre. Par exemple, dans le cas qui nous intéresse, celui de l'ambiguïté d'une exigence, les critères d'ambiguité, de complétude et de vérifiabilité que l'on retrouve dans tous les modèles de qualité ne peuvent pas former trois catégories indépendantes car ils sont mutuellement liés par des relations causales. Une exigence incomplète – sans intervalle mesurable, par exemple – est non seulement invérifiable, mais aussi ambiguë car on peut imaginer tous les intervalles possibles. Par ailleurs, une exigence qui est ambiguë d'un point de vue linguistique (lexical, syntaxique ou sémantique) est probablement vouée à être interprétée de différentes manières.

Une définition hiérarchique des critères n'est donc pas suffisante pour analyser la qualité d'une exigence ou d'un ensemble d'exigences, il faut aussi considérer les relations causales inter-critères. Même si l'initiative de Saavedra et al. [2013] est pertinente, son contexte d'application se limite à un ensemble d'exigences logicielles (*Software Requirements Specification*, en anglais), tandis que nous ne faisons pas d'hypothèse sur la nature du produit spécifié. Par ailleurs, nous étudions les défauts d'une exigence ou d'une paire d'exigences, mais pas ceux d'un ensemble d'exigences faisant office de spécification.

Les revues d'experts en ingénierie des exigences

La première approche pour fiabiliser l'interprétation d'une exigence c'est de monopoliser le savoir d'experts en ingénierie des exigences pour identifier et supprimer les défauts de qualité. Aussi surprenant que cela puisse paraître, même une vingtaine d'années après Hooks [1994], les revues manuelles sont toujours d'actualité [Lami, 2005, Ott, 2012, INCOSE, 2015a, Carlson and Laplante, 2013]. Ces approches sont même parfois mises en œuvre sans que la majorité des défauts ne soit identifiés [Firesmith, 2005].

Cette approche a de multiples désavantages. Premièrement, elle requiert des experts formés à la rédaction des exigences. Par ailleurs, en plus d'accroître les coûts fonctionnels de l'entreprise, cela nécessite des dépenses excessives compte tenu du temps nécessaire pour relire minutieusement chacune des centaines ou milliers d'exigences. Malgré ces désavantages, les revues d'experts ont néanmoins l'avantage de fournir de bonnes performances. Cependant, quand la taille des données devient trop importante, la revue manuelle ne suffit pas, en particulier pour identifier les redondances et les contradictions.

1. Une typologie est une partition mathématique d'un ensemble E tel que chaque élément de E appartient à une et une seule partition.

Les techniques inductives

L'ambiguïté d'une exigence ou d'un ensemble d'exigences peut aussi être réduite par des techniques inductives. Les techniques inductives consistent à capitaliser et réutiliser un ensemble de règles de bonnes pratiques pour rédiger des exigences. Par exemple, quand un expert fait une revue manuelle d'une spécification, il collectionne tous les défauts et rédige une note de recommandation destinée aux personnes en charge de la spécification. Ces consignes sont ensuite appliquées dans les projets ultérieurs. Une technique inductive populaire est le guide de rédaction des exigences de l'INCOSE [2015a].

Les techniques inductives, en particulier les guides de bonnes pratiques de rédaction des exigences, sont très utiles *ex ante*, mais rares sont les entreprises qui ont connaissance de tels guides ou les analystes qui maîtrisent ces règles.

Les techniques restrictives

Le champ des interprétations peut aussi être restreint par des techniques restrictives. La première technique restrictive correspond aux langages naturels contrôlés. Un langage naturel contrôlé est un langage naturel dont le lexique, la syntaxe et/ou la sémantique est contraint tout en préservant quelques propriétés naturelles [Kuhn, 2014]. La marge de manœuvre des analystes qui rédigent les énoncés est alors fortement réduite, et ce, pour deux objectifs : (1) simplifier la lecture par un être humain, et (2) automatiser le traitement avec une machine. Les langages « *ASD Simplified Technical English* » (ASD-STE), « *Standard Language* » (SLANG), « *Gellish English* », « *Attempto Controlled English* (ACE) », ou « *Common Logic Controlled English* » (CLCE) sont des exemples de langages naturels contrôlés.

La deuxième technique restrictive s'appuie sur des patrons (*template*, en anglais) d'exigence. Un patron d'exigence est un énoncé pré-formaté dont la syntaxe est figée, mais dont certaines locutions doivent être complétées par l'analyste. C'est la solution la plus restrictive. Divers patrons ont été proposés [Hull et al., 2011, AFIS, 2012a, INCOSE, 2015a]. Les énoncés ci-dessous sont deux exemples de patrons définis par l'INCOSE [2015a] :

- The **<subject clause>** shall, when **<condition clause>**, **<action verb clause> <optional qualifying clause>**.
- *The* ***Control_Subsystem*** *shall, when* ***the temperature of water in the Boiler is less than 85 °C, open the Inlet_Valve in less than 3 seconds***.
- The **<subject clause>** shall **<action verb clause> <object clause> <optional qualifying clause>**, when **<conditional clause>**.
- *The* ***Control_subsystem*** *shall o****pen the Inlet_Valve in less than 3 seconds****, when* ***the temperature of water in the Boiler is less than 85 °C***.

La troisième technique restrictive se base sur des locutions passe-partout (*boilerplate*, en anglais). La syntaxe de ces locutions est fixée, mais l'ordre dans lequel elles sont agencées au sein d'une proposition ne l'est pas. Par exemple, la condition d'applicabilité peut être mise avant la fonction – « *When the aircraft is on groud, the landing gear shall ...* » –, ou après la fonction – « *The landing gear shall ..., when the aircracft is on ground* ». Cette modularité rend la solution plus flexible que les patrons, car divers énoncés compliqués peuvent être obtenus en combinant des locutions passe-partout. Habituellement, chaque locution est catégorisée et a un but bien identifié. Par exemple, la locution « *The <system > shall <action>* » appartient à la catégorie des « Fonctions », tandis que la locution « *at least < quantity > times per < time unit>* » appartient à la catégorie des « Performances ». Hull et al. [2011] donnent une liste de locutions.

La suite logicielle proposée par The Reuse Company [Fraga et al., 2015] combine l'approche des locutions passe-partout et un vocabulaire contrôlé qui doit être manuellement créé et maintenu.

Les techniques restrictives sont tantôt approuvées tantôt désapprouvées. La définition d'un vocabulaire contrôlé est par exemple contraignante car elle nécessite l'intervention et un consensus d'experts. Le vocabulaire doit aussi être maintenu tout au long d'un ou plusieurs projets sans être forcément portable d'un contexte à l'autre. D'autre part, les locutions passe-partout et les patrons sont souvent perçus comme étant trop restrictifs. Une restriction lexicale et syntaxique a néanmoins l'avantage d'assister les

néophytes en ingénierie des exigences. Enfin, si les mots et/ou l'ordre des mots sont pré-définis, alors cela facilite la mise en œuvre des techniques analytiques présentées ci-après.

Les techniques analytiques

La dernière approche pour fiabiliser l'interprétation des exigences repose sur un panel de techniques analytiques. Les techniques analytiques exploitent des opérations de traitement du langage naturel et de fouille de données pour automatiser l'identification des défauts de qualité à l'échelle de l'exigence ou du document prescriptif. L'objectif idéal des techniques analytiques est de remplacer les fastidieuses revues manuelles. Pour cela, les algorithmes implémentent des règles qui vérifient les critères de qualité définis dans les modèles de qualité. Ainsi, il existe presque autant d'outils analytiques que de modèles de qualité.

La majorité des solutions analytiques visent à détecter les défauts intrinsèques à une exigence ou évaluer la qualité d'un document de spécification. Comme précurseur, on trouve le logiciel Automated Requirements Measurement (ARMS) [Wilson et al., 1997]. ARMS a été développé par la NASA pour étudier la qualité d'une exigence et d'un ensemble d'exigences en se basant sur des indicateurs tels que le nombre de mots dans une exigence, le nombre d'exigences, le type de verbe modal utilisé, etc. Carlson and Laplante [2013] ont fait une rétro-ingénierie d'ARMS. Lami [2005] a proposé le Quality Analyzer for Requirements Specification (QuARS) pour évaluer l'expressivité, la complétude et la consistance d'un document prescriptif avec des techniques linguistiques qui traitent le niveau lexical et syntaxique. L'outil LEXical analysis for Improvement Of Requirements (LEXIOR) de Szczepaniak and Defarge [2006] est un environnement collaboratif qui diagnostique les défauts de qualité dans des documents prescriptifs hérités d'un donneur d'ordres. Les défauts sont alors stockés afin de dériver les bonnes pratiques qui serviront aux prochains projets. Kiyavitskaya et al. [2008] suggèrent une approche en deux temps pour, d'une part, identifier les phrases ambiguës et, d'autre part, diagnostiquer la racine des ambiguïtés. La solution repose aussi sur des opérations de traitement du langage naturel. Pour chaque exigence, l'outil analytique Tool to InGest and Elucidate Requirements (TIGER) de Kasser [2006b] mesure un ensemble d'indicateurs de qualité avant de fournir des résumés sur un tableau de bord. Lamar [2009] s'appuie sur des patrons d'exigences et des techniques de traitement du langage naturel pour identifier les exigences incomplètes. Le logiciel Requirements Engineering Specification Improver [körner and Brumm, 2009, Körner and Brumm, 2010] exploite des ontologies, ainsi que des techniques de traitement du langage naturel pour assister les analystes en identifiant et signalant les exigences ambiguës, défectueuses ou imprécises. Yang et al. [2011] proposent une approche pour identifier les ambigüités anaphoriques - c'est-à-dire les ambiguïtés qui apparaissent quand les lecteurs sont en désaccord vis-à-vis de l'interprétation des pronoms tels que « *it* », « *then* » et « *their* ». Christophe et al. [2011] suggèrent de désambiguïser les exigences en agissant sur la grammaire, la sélection des mots et la description du contexte en réutilisant le Recursive Object Model de Wang and Zeng [2009] et l'atlas sémantique de Ploux et al. [2010]. Si la réponse aux questions permettant de clarifier les mots clés n'est pas automatisée, alors elle n'est pas applicable en pratique car cela demande trop de travail. Moser et al. [2011] proposent une solution technologique sémantique pour automatiser la gestion des exigences et le reporting basé sur l'ontologie OntRep, laquelle facilite la catégorisation automatique des exigences et l'identification des conflits. L'outil REquirements Analysis aSSISTANT – REAssistant [Rago et al., 2012] supporte le processus de découverte des défauts dans les exigences textuelles en s'appuyant sur le framework de traitement du lange naturel UIMA et les technologies basées sur EMF. Mokammel et al. [2013] collectent des exigences en lisant des fichier XML. Les exigences sont ensuite analysées par un outil de traitement du langage naturel pour évaluer divers indicateurs tels que des termes vagues (*easy*, *strong*, ...), subjectifs (*similar*, *as X as possible*, ...), optionnels (*possibly*, *eventually*, ...), etc. Ici, nous retrouvons le problème du à une catégorisation non stricte des indicateurs. Par exemple, le mot optionnel *possibly* est aussi vague. Quel est donc l'intérêt de faire ces catégories ? Le logiciel ReqUirements BoileRplate sanIty Checker (RUBRIC) [Arora et al., 2013b] sert à vérifier si un énoncé d'exigence est conforme à une grammaire pré-définie. L'utilisateur peut ainsi vérifier si son énoncé est conforme à un patron. Génova et al. [2013] proposent l'outil Requirements Quality Analyzer (RQA) qui, lui aussi,

vérifie la qualité d'une exigence et d'un ensemble d'exigences à partir d'indicateurs tels que le nombre de mots, la lisibilité, la ponctuation, les acronymes, les termes vagues, etc. Ce qui est intéressant dans leurs travaux, c'est l'utilisation de trois fonctions – convexe, croissante, décroissante – de qualité. Par exemple, plus une exigence est longue ou courte – fonction concave – plus sa qualité est mauvaise. Plus, le nombre d'acronymes est important – fonction croissante – plus l'exigence est de mauvaise qualité. Pour récupérer les données nécessaires, l'outil se greffe comme un plug-in à une base de données comme IBM DOORS. Kang and Saint-Dizier [2015a] proposent une méthode basée sur le traitement du langage naturel en apprenant des corrections réalisées par les auteurs. Enfin, Kang and Saint-Dizier [2015b] proposent une mise en œuvre de l'outil LELIE qui correspond à la version académique de la solution commerciale Semios de Prometil. le moteur d'analyse linguistique de LELIE s'appui sur des règles très, voire même trop spécifiques qui nécessitent d'être personnalisées avant chaque analyse. Avant de traiter un document, il faut donc commencer à l'étudier manuellement pour configurer l'outil.

Beaucoup moins de travaux s'intéressent à la détection de défauts relationnels entre les exigences. Dans leur étude de faisabilité, Natt och Dag et al. [2002] ont étudié trois métriques de similarité textuelle – Jaccard, Dice et Cosine – pour identifier les exigences redondantes. Leurs expérimentations concluent que les métriques Dice et Cosine donnent de meilleurs résultats que la fonction Jaccard. Cependant, leurs implémentations n'exploitent aucune ressource de synonymie, on peut donc remettre en cause la robustesse de leur solution dans un cas pratique. Ilyas and Küng [2009] ont aussi étudié le comportement de ces trois fonctions de similarité et ont trouvé que la métrique Cosine est plus performante que les deux autres, mais là aussi la synonymie n'est pas considérée. Mokammel et al. [2013] proposent une nouvelle métrique qui considère les mots composés et la synonymie pour mesurer la similarité entre deux exigences. Coatanéa et al. [2013] ont proposé quelques algorithmes pour identifier les contradictions entre les exigences. Selon les auteurs il y a six types de contradictions. Les contradictions liées à : des (1) antonymes, des (2) négations, ou des (3) valeurs numériques. Il y a aussi les contradictions dues à des connaissances (4) générales, (5) lexicales ou (6) structurelles. Les auteurs avancent avoir trouvé des solutions pour identifier les trois premiers types de contradictions. Cependant, quasiment aucune description explicative n'est associée aux algorithmes proposés, il est donc difficile d'accepter la proposition qui est faite. Par ailleurs, la lecture des diagrammes laisse entrevoir des limites majeures. Par exemple, la détection de contradictions dues à des antonymes se limitent aux noms (Fig. 5.2). Or, les contradictions peuvent aussi être issues d'un verbe « *increase* vs. *decrease* », d'un adverbe « *fast* vs. *slow* » ou d'un adjectif « *green* vs. *red* ». Idem, pour l'algorithme qui identifie les contradictions numériques en se limitant aux noms et aux valeurs numériques quelconques. Les contre-exemples sont nombreux. Par exemple, deux exigences qui contiennent une référence à une norme « ISO 9000 » et « CS 25 » contiennent des valeurs numériques mais ne sont forcement contradictoires. Enfin, le troisième algorithme d'identification des contradictions dues à des négations se limitent aussi aux noms. Or, les négations peuvent être liées à des verbes « *increase* vs. *not increase* », des adverbes, etc. Par ailleurs, le système semble être totalement déterministe en utilisant des règles. On peut alors s'interroger à propos de la robustesse de ces règles inconnues dès lors qu'elles sont appliquées au langage naturel non structuré peuplé de synonymes et d'homonymes, et dont la syntaxe change considérablement d'un énoncé à l'autre. Tous ces éléments nous emmènent à conclure que la solution est très incomplète, voire inefficace. Enfin, Fraga et al. [2015] identifient les exigences similaires, donc potentiellement redondantes. Pour cela, à partir d'un vocabulaire contrôlé pré-défini, ils construisent deux graphes pour chaque paire d'exigences. La comparaison des graphes permet alors de mesurer le niveau de similarité entre les deux exigences. Ici, l'exercice est simplifié car le vocabulaire et la grammaire sont restreints et connus à l'avance.

Les solutions analytiques présentées ci-dessus pourraient être combinées à des méthodes de visualisations pour fiabiliser l'interprétation des exigences. La visualisation analytique consiste à concevoir des interfaces visuelles interactives pour faciliter le raisonnement analytique [Thomas and Cook, 2005]. Reddivari et al. [2014] ont dérivé le terme de visualisation analytique d'exigences (*visual requirements analytics*, en anglais) pour désigner l'application des techniques de visualisation analytique à l'ingénierie des exigences. Par exemple, Heim et al. [2008] permettent à un analyste de visualiser les redondances et les contradictions dans un graphe d'exigences. Ces visualisations doivent néanmoins être définies ma-

nuellement car l'outil n'a aucun moyen de traitement du langage naturel.

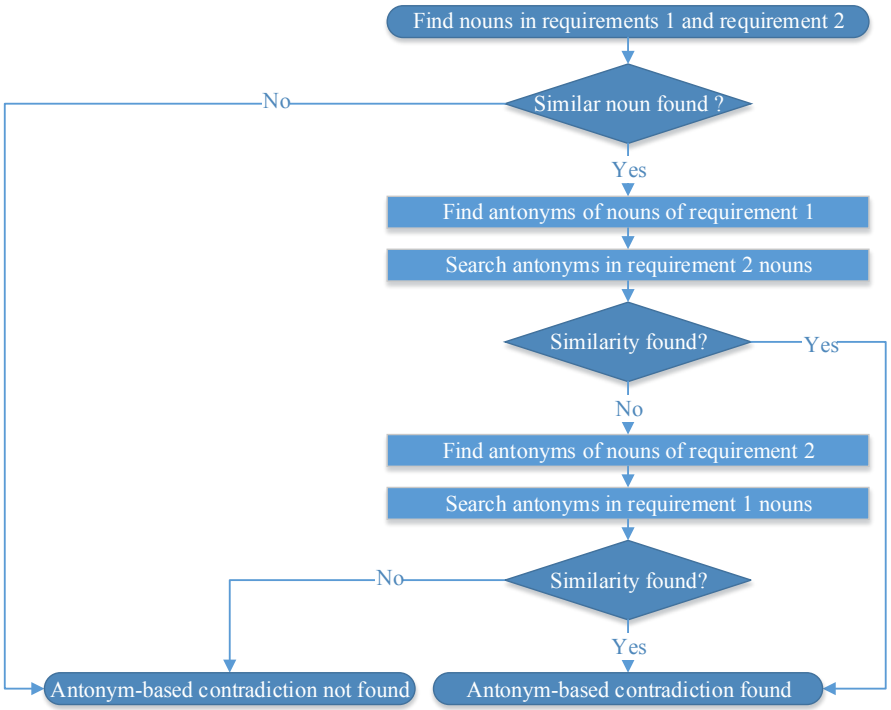


FIGURE 5.2 – Algorithme de détection des contradictions dues à des antonymes selon Coatanéa et al. [2013].

Synthèse des méthodes de fiabilisation des exigences

Il est étonnant de constater que les exigences soient encore de mauvaise qualité quand on observe toute l'étendue des travaux de recherche qui ont été menés. Nous identifions plusieurs raisons pour expliquer cette situation.

La première raison c'est qu'il y a un nombre démesuré de modèles de qualité outillés ou non. En effet, de multiples critères de qualité et indicateurs de mesure sont proposés par chacun des groupes de recherche. Des outils sont alors développés pour automatiser le diagnostic des exigences. Cependant, comme cela a été expliqué précédemment, il n'y a aucun consensus sur ces critères. Bien que certains soient normés, la plupart d'entre eux résultent d'une analyse subjective.

Par ailleurs, la majorité des approches se concentrent sur la détection des défauts intrinsèques à une exigence. Il est possible qu'un expert puisse identifier des redondances ou des contradictions entre les exigences lors d'une revue manuelle. Cependant, quand les exigences sont réparties dans plusieurs documents prescriptifs qui spécifient des centaines ou des milliers d'exigences, il est quasiment impossible qu'il puisse identifier les défauts relationnels. Les solutions analytiques proposées n'ont toujours pas apporté de solution à la détection des contradictions entre exigences.

Enfin, beaucoup d'outils mettent en œuvre des techniques d'analyses telles que le traitement du langage naturel, la fouille de textes, les ontologies, etc. sans se soucier de l'aspect graphique permettant l'interaction avec l'utilisateur. Inversement, les quelques travaux qui proposent des représentations graphiques sont indépendants des solutions de traitement. Il manque donc une intégration des deux perspectives dans un environnement unique.

5.3 Proposition

Avant d'exposer une proposition de solution pour le problème des interprétations multiples, il faut dégager les caractéristiques essentielles d'une solution « équilibrée » permettant de fiabiliser l'interprétation des exigences.

Quelle approche pour fiabiliser l'interprétation ?

Toutes les propositions discutées ci-dessus diffèrent selon 5 oppositions qui permettent d'identifier les caractéristiques qu'une solution équilibrée doit posséder :

- **A priori *VS.* A posteriori.** Les techniques a priori sont celles qui traitent la cause des ambiguïtés. Les techniques a priori sont appliquées avant ou pendant la rédaction des exigences avec comme ambition : *Right the first time !*. Les patrons, les locutions passe-partout, et les vocabulaires contrôlés sont des solutions restrictives a priori. Les guides de bonnes pratiques sont des solutions inductives a priori. Inversement, hormis quelques exceptions, les méthodes analytiques sont des solutions a posteriori qui traitent les symptômes. En effet, elles servent à détecter les défauts après que les exigences aient été spécifiées. Bien que les techniques a priori semblent être un bon remède, le fournisseur n'a aucun moyen pour agir sur la qualité du document prescriptif qu'un client lui fourni. Il ne peut pas lui imposer des règles de rédaction, ni le forcer à suivre une formation en ingénierie des exigences. Enfin, comme cela a été expliqué précédemment, toutes les exigences légales et règlementaires qui sont au cœur du problème sont volontairement ambiguës, donc les méthodes a posteriori sont indispensables. Une solution équilibrée doit donc proposer des fonctionnalités a priori et a posteriori. Les méthodes a priori servent à assister les utilisateurs qui ne sont pas familiers avec les bonnes pratiques de rédaction des exigences, tandis que les méthodes a posteriori diagnostiquent les défauts dans les exigences héritées du donneur d'ordres.
- **Manuelle *VS.* Automatique.** Les revues manuelles sont particulièrement chronophages et monopolisent des experts. De plus, les revues sont des solutions a-posteriori réalisées après que les exigences ait été spécifiées ce qui augmente le temps et les coûts relatifs à l'ingénierie des exigences. Avec la prolifération des exigences, ce type d'approche tend à devenir impraticable, mais elles reste néanmoins un gage de qualité non négligeable. À l'opposé, l'automatisation des techniques inductives – la vérification des bonnes pratiques, par exemple – permet de limiter les délais et les dépenses. Malgré l'investissement des chercheurs pour essayer de rendre les machines « intelligentes », ces dernières n'ont toujours pas atteint les facultés humaines. Par exemple, l'identification de certains types de contradictions entre deux énoncés textuels reste un problème fondamental en traitement automatique du langage naturel, alors que c'est trivial pour un humain. Par conséquent, une solution équilibrée doit être semi-automatique, c'est-à-dire une combinaison de tâches dont certaines sont réalisées par la machine et d'autres par l'humain. Plus précisément, la machine doit mécaniser les opérations fastidieuses avant de restituer les informations pertinentes nécessaires à l'humaine qui exploite ses connaissances et ses intuitions pour décider.
- **Calcul *VS.* Graphique.** L'état de l'art a montré que la majorité des propositions se concentrent sur la mécanique du calcul, du traitement. Or, toutes les composantes des sciences des données partagent une logique commune qui consiste à non seulement traiter les données pour extraire les informations clés, mais aussi à les restituer graphiquement. Dans certains cas, les représentations graphiques peuvent permettre d'observer des informations qui été initialement invisibles. Il suffit de se référer au quartet d'Anscombe [1973] pour s'en convaincre. Par ailleurs, les méthodes analytiques s'appuient sur des techniques probabilistes – le traitement du langage naturel, par exemple –, les représentations graphiques permettent alors de notifier les résultats qui sont probablement des faux positifs. Ainsi, une solution équilibrée doit conjuguer des techniques de traitement et des représentations graphiques.
- **Structuré *VS.* (Semi-)Structuré.** La troisième comparaison oppose les solutions qui traitent des documents prescriptifs non structurés à celles qui se limitent aux (semi-)structurés. La majorité des méthodes analytiques viennent se greffer sur des bases de données relationnelles, ou ex-

ploitent des méthodes restrictives – des vocabulaires contrôlés ou des patrons, par exemple –, ce qui facilite amplement l'analyse des données. En pratique, les documents prescriptifs non structurés sont néanmoins plus populaires que les (semi-)structurés. De plus, les exigences ne sont habituellement pas rédigées avec des méthodes restrictives, et ce, pour plusieurs raisons : les experts ne partagent pas un glossaire commun ; la définition et la maintenance d'un vocabulaire contrôlé est trop laborieuse et non portative d'un contexte (projet, entreprise, etc.) à l'autre ; les patrons d'exigences sont trop restrictifs ; etc. Une solution équilibrée doit se concentrer sur le cas le plus défavorable et le plus populaire qui est celui de l'analyse de documents et d'exigences non structurés.

– **Intrinsèque *VS.* Relationnel.** Enfin, alors que l'identification de défauts intrinsèques à une exigence ou un document prescriptif est l'objet de nombreux travaux, seules quelques propositions s'attaquent à la détection de défauts relationnels entre les exigences. Des fonctions de similarité ont été appliquées pour détecter les redondances, mais elles ne considèrent pas la synonymie des termes. Enfin, comme cela a été discuté précédemment, les solutions proposées pour identifier les contradictions entre exigences sont très limitées. Une solution équilibrée doit identifier les défauts intrinsèques et relationnels, avec un zoom sur le problème irrésolu des contradictions.

Une solution équilibrée doit posséder toutes les caractéristiques discutées ci-dessus en s'appuyant sur un ensemble de causes qui influent sur l'ambiguïté d'une exigence.

Un diagramme d'influence pour fiabiliser l'interprétation

Pour toutes les raisons énoncées dans l'état de l'art, l'objectif n'est pas de proposer un énième modèle de qualité. Ce qui importe c'est d'identifier un ensemble de causes qui tendent soit à fiabiliser l'interprétation d'une exigence – les effets positifs –, soit à accroître leur ambigüité – les effets négatifs. La figure 5.3 illustre un diagramme d'influence qu'on peut lire ainsi :

- Si le nombre de défauts intrinsèques à une exigence diminue alors l'ambiguïté de cette exigence diminue, et pour cela on peut :
  - détecter, notifier et supprimer les exigences incomplètes.
  - détecter, notifier et supprimer les exigences qui contiennent des termes vagues.
  - détecter, notifier et supprimer les exigences qui contiennent des connecteurs.
- Si le nombre de défauts relationnels à une exigence diminue, alors l'ambiguïté de cette exigence diminue, et pour cela on peut :
  - détecter, notifier et supprimer les contradictions.
  - détecter, notifier et supprimer les redondances.
- Si le contexte d'une exigence est plus riche (dans une certaine limite), alors l'ambiguïté de cette exigence diminue, et pour cela on peut :
  - mettre en évidence les exigences voisines reliées par des interdépendances transversales.
  - mettre en évidence les exigences voisines reliées par des interdépendances sémantiques.
  - mettre en évidence les exigences voisines reliées par des interdépendances conceptuelles.

Dans la suite de ce chapitre nous présentons différents moyens pour détecter les défauts intrinsèques et relationnels qui doivent être supprimés par l'analyste. Les détails de la troisième solution « Contexte », laquelle consiste à restituer chaque exigence dans un contexte qui facilite l'interprétation, ne sont pas présentés dans ce chapitre. Ils ne rendent pas service aux analystes qui sont chargés de supprimer les défauts de qualité, mais aux experts lorsqu'ils estiment les critères d'aide à la décision associés aux exigences. Nous reviendrons donc sur cette contribution dans la section 6.3 du chapitre suivant.

Identifier les défauts intrinsèques (FT21)

Dans la catégorie des défauts intrinsèques, la première cause qui influe sur l'ambigüité d'une exigence est la présence de connecteurs. Les connecteurs sont des termes qui introduisent une conjonction ou une disjonction entre deux locutions. Les connecteurs que nous avons identifiés sont : *and*, *and/or*, *as well as*, *both*, *but*, *but also*, *however*, *meanwhile*, *on the other hand*, *or*, *otherwise*, *then*, *unless*, *whereas*,

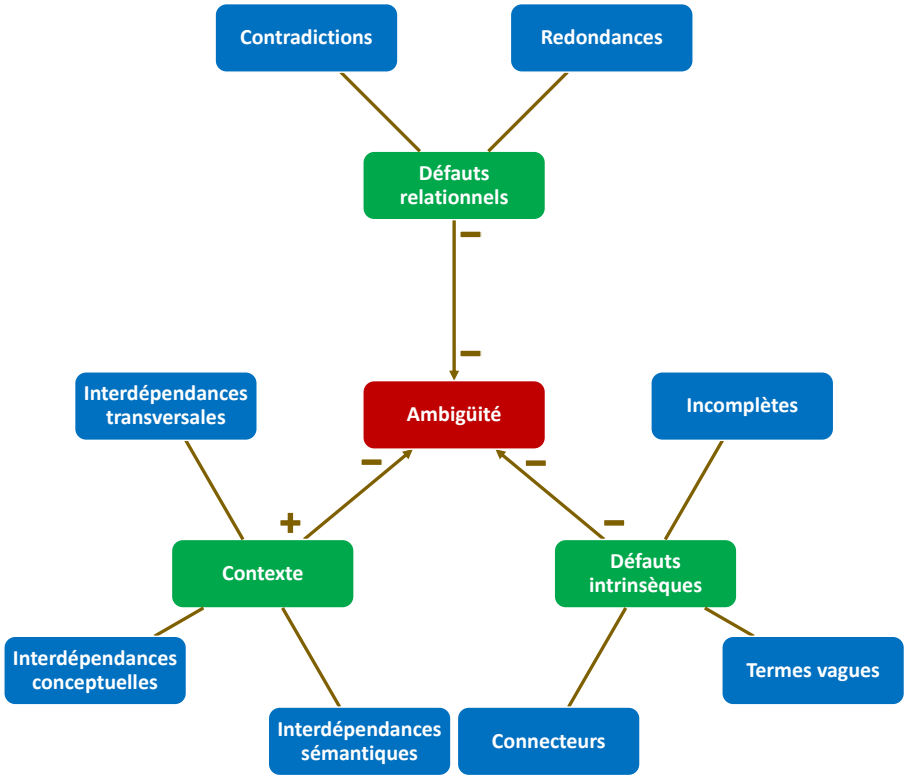


FIGURE 5.3 – Diagramme d'influence des causes qui influent sur l'ambigüité d'une exigence.

whether, et le symbole /. Pour vérifier si une exigence contient un connecteur, nous parcourons la liste des lemmes dans chaque objet « Exigence ».

Toujours dans la catégorie des défauts intrinsèques, la deuxième cause qui influe sur l'ambigüité d'une exigence est la présence de termes vagues. Les termes vagues sont des mots qui ne sont pas précis laissant encore place à de multiples interprétations. Pour les identifier, nous utilisons des expressions régulières et un dictionnaire qui contient environ 200 termes vagues incluant :

- les articles indéfinis tels que *a* et *an*,
- les pronoms tels que *it*, *they*, [...], *them* utilisés dans les anaphores,
- les unités lexicales universelles telles que *any*, *all*, [...], *fully*,
- les mots ou locutions qualitatives telles que *a few*, *a lot of*, *acceptable*, *accurate*, *as appropriate*, *as much as possible*, *easily*, [...], *frequently*.

Les deux causes précédentes sont relativement bien traitées dans la littérature et n'apportent pas de nouveaux éléments. La troisième cause d'ambigüité, les exigences incomplètes, n'a pas été étudiée, sauf dans les solutions qui analysent les exigences dont la rédaction s'appuie sur des structures syntaxiques pré-définies comme les patrons. En effet, il est possible de vérifier qu'une exigence est conforme à une grammaire pré-définie. Certains auteurs utilisent des grammaires formelles – Extended Backus-Naur Form (EBNF), par exemple – [Arora et al., 2013b], tandis que d'autres exploitent un vocabulaire contrôlé pré-annoté [Fraga et al., 2015].

Pour être complète, nous postulons qu'une exigence ne doit pas nécessairement être conforme à une grammaire spécifique. Un même énoncé peut être syntaxiquement complet mais l'ordre des mots peut varier. Ainsi, ici, nous n'imposons aucune structure syntaxique. Nous utilisons la notion de complétude qui a été formellement définie dans le concept théorique de Property Based Requirement (PBR) développé par Micouin [2008]. Pour être complète, une exigence doit être constituée de quatre éléments : une condition C, un objet O, une propriété P et un domaine D tel que :

$$When\ C \longrightarrow Val(O.P) \in D \tag{5.1}$$

Une PBR se lit alors « Quand la condition C est satisfaite, la propriété P de l'objet O doit appartenir au domaine D ». Par exemple, la PBR ci-dessous se lit « Quand l'avion (AC) est situé entre 0 et 5000 pieds, l'erreur absolue pour l'altitude indiquée (Indicated_Alt) par l'indicateur d'altitude (ADC) doit être inférieure à 25 pieds ».

$$When\ AC.Alt \in [0, 5000] \longrightarrow\ val(ADC.Indicated\_Alt - AC.Alt) < 25ft \tag{5.2}$$

Dans certains cas, la condition est optionnelle. Par exemple, l'exigence « Le nombre de passagers dans l'hélicoptère doit être supérieur à 10 » se traduit formellement par la PBR suivante :

$$val(HC.Passenger\_seat\_number > 10) \tag{5.3}$$

D'un point de vue syntaxique, une exigence ne peut avoir que quatre structures (Tab. 5.1). Parmi ces quatre structures, si la condition est absente, alors on ne peut rien conclure car on ne sais pas si elle est indispensable ou non à l'interprétation de l'exigence (Ex. 5.2 et Ex. 5.3). Ainsi, ci-après, nous proposons un moyen de vérifier que chaque exigence possède une condition, mais cette vérification n'est pas impérative.

Structure 1 : «Prescriptive» «Domain» «Condition» – $\mathcal{PDC}$
The Control_Subsystem shall open the Inlet_Valve in less than 3 seconds when the temperature of water in the Boiler is less than 85 °C.
Structure 2 : «Prescriptive» «Condition» «Domain» – $\mathcal{PCD}$
The Control_Subsystem shall, when the temperature of water in the Boiler is less than 85 °C, open the Inlet_Valve in less than 3 seconds.
Structure 3 : «Condition» «Prescriptive» «Domain» – $\mathcal{CPD}$
When the temperature of water in the Boiler is less than 85 °C the Control_Subsystem shall open the Inlet_Valve in less than 3 seconds.
Structure 4 : «Prescriptive» «Domain» – $\mathcal{PD}$
The Control_Subsystem shall open the Inlet_Valve in less than 3 seconds.

TABLE 5.1 – Les quatre structures syntaxiques d'une exigence.

Si une exigence prescrit une propriété physique[2] (masse, vitesse, énergie, taille, force, tension, puissance, etc.), alors il est indispensable de vérifier qu'elle possède un domaine mesurable auquel la propriété doit appartenir. Si aucun domaine mesurable n'est détecté, alors l'exigence est annotée comme incomplète. Comme pour la présence ou non d'une condition, il se peut qu'une exigence soit complète sans posséder de domaine mesurable. Par exemple, l'exigence « Le système doit être conforme à la norme CS 25 » est complète et ne possède pourtant pas de domaine mesurable. Cependant, en pratique, hormis quelques exceptions, toutes les exigences doivent prescrire une propriété mesurable avec un domaine de tolérance sans quoi sinon n'importe quelle solution satisferait les exigences.

Pour vérifier qu'une exigence est complète, la première étape consiste à construire son graphe sémantique. Le groupe de recherche de traitement du langage naturel de Stanford a développé un parseur de dépendances basé sur un réseau de neurones préalablement entraîné sur un corpus d'apprentissage [Chen and Manning, 2014]. Comme le montre la figure 5.4, un graphe sémantique est un ensemble d'arrêtes annotées qui modélisent les dépendances entre les tokens.

Dans chaque graphe sémantique, nous recherchons les dépendances de type *nummod* qui signifient *numerical modifier*. Une dépendance *nummod* modélise la relation entre une valeur numérique et le token qu'elle modifie. Comme on le voit sur la figure 5.4, chaque valeur numérique est stockée dans le nœud cible, tandis que le token modifié est stocké dans le nœud source. En utilisant un dictionnaire de dimensions physiques – e.g. pressure – et les unités associées – bar, Pa, etc. –, on peut vérifier que le token modifié est une propriété physique. Pour chaque exigence, nous obtenons alors une liste de

2. Le concept « d'exigence de performance » prédomine dans la littérature.

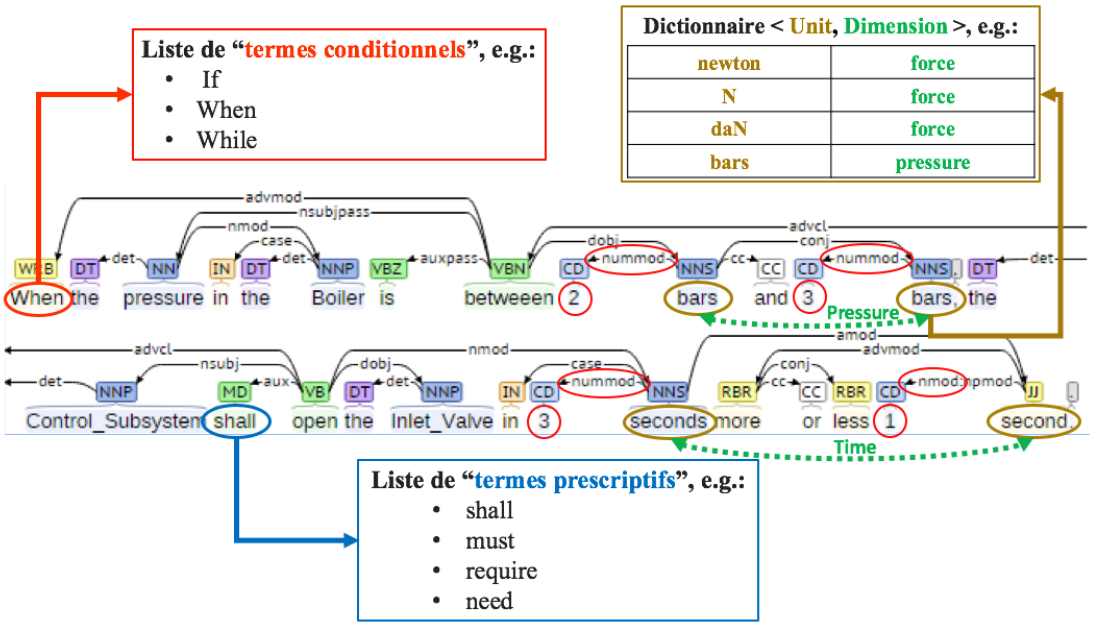


FIGURE 5.4 – Graphe sémantique d'une exigence.

couples <valeur numérique, unité physique>. Dans l'exemple de la figure 5.4 la liste de couples est : <2, bars>, <3, bars>, <3, seconds> et <1, second>.

Règle 1 :	**IF** $P < PND < C$ **THEN** *Structure 1*
Règle 2 :	**IF** $P < C < PND$ **THEN** *Structure 2*
Règle 3 :	**IF** $C < P < PND$ **THEN** *Structure 3*
Règle 4 :	**IF** $C = \emptyset$ **AND** $P < PND$ **THEN** *Structure 4*

TABLE 5.2 – Règles pour identifier les quatre structures syntaxiques d'une exigence complète.

Par ailleurs, pour chaque exigence, nous connaissons la position du terme prescriptif – *shall* (Fig. 5.4) –, ainsi que celle du terme conditionnel tel que *if*, *when* ou *while* – *when* (Fig. 5.4) – qui sert à introduire la condition. Puisqu'une exigence complète ne peut avoir que quatre structures syntaxiques, nous vérifions que l'exigence est potentiellement complète en appliquant quatre règles (Tab. 5.2), lesquelles peuvent être traduites ainsi :

- Si le terme prescriptif P est placé avant une dépendance physique numérique PND qui, elle même, est placée avant le terme conditionnel C, alors l'exigence est potentiellement complète et conforme à la structure 1.
- Si le terme prescriptif P est placé avant le terme conditionnel C qui, lui même, est placé avant une dépendance physique numérique PND, alors l'exigence est potentiellement complète et conforme à la structure 2.
- Si le terme conditionnel C est placé avant le terme prescriptif P qui, lui même, est placé avant une dépendance physique numérique PND, alors l'exigence est potentiellement complète et conforme à la structure 3.
- Si il n'y a pas de terme conditionnel C et que le terme prescriptif P est placé avant une dépendance physique numérique PND, alors l'exigence est potentiellement complète et conforme à la structure 4.

Notifier (FT23) et supprimer (FT25) les défauts intrinsèques

Dans un premier temps, les défauts intrinsèques sont représentés par des communautés (Fig. 5.5) dont la couleur correspond à un type de défaut. La taille d'une communauté est proportionnelle au

nombre d'exigences qu'elle contient. Habituellement, l'échelle est définie via une fonction quadratique, racine carrée ou logarithmique. Ici, ce n'est pas le cas, toutes les communautés sont elles mêmes contenues dans une communauté invisible de plus haut niveau dont le diamètre est fixé selon la taille de l'écran. Le regroupement des exigences selon les types de défauts donne un premier aperçu des défauts intrinsèques les plus fréquents. Dans l'exemple ci-dessus, on constate que les défauts sont dus à la présence de connecteurs (*combinators*, en anglais) et à l'incomplétude des exigences : pas de condition, pas d'unité physique, pas de domaine mesurable, etc. Cela permet aussi à l'analyste d'exclure les catégories dont il ne souhaite pas se soucier parce qu'il juge qu'elles ne sont pas véritablement critiques ou parce que les données ne sont pas en adéquation avec les défauts recherchés. Par exemple, les défauts liés à l'utilisation de la voie passive ou l'utilisation de parenthèses ne sont pas véritablement critiques. D'autre part, si aucune exigence ne possède une condition d'applicabilité, alors les communautés associées « *No condition* », « *Condition not quantified* » peuvent être également exclues par l'analyste.

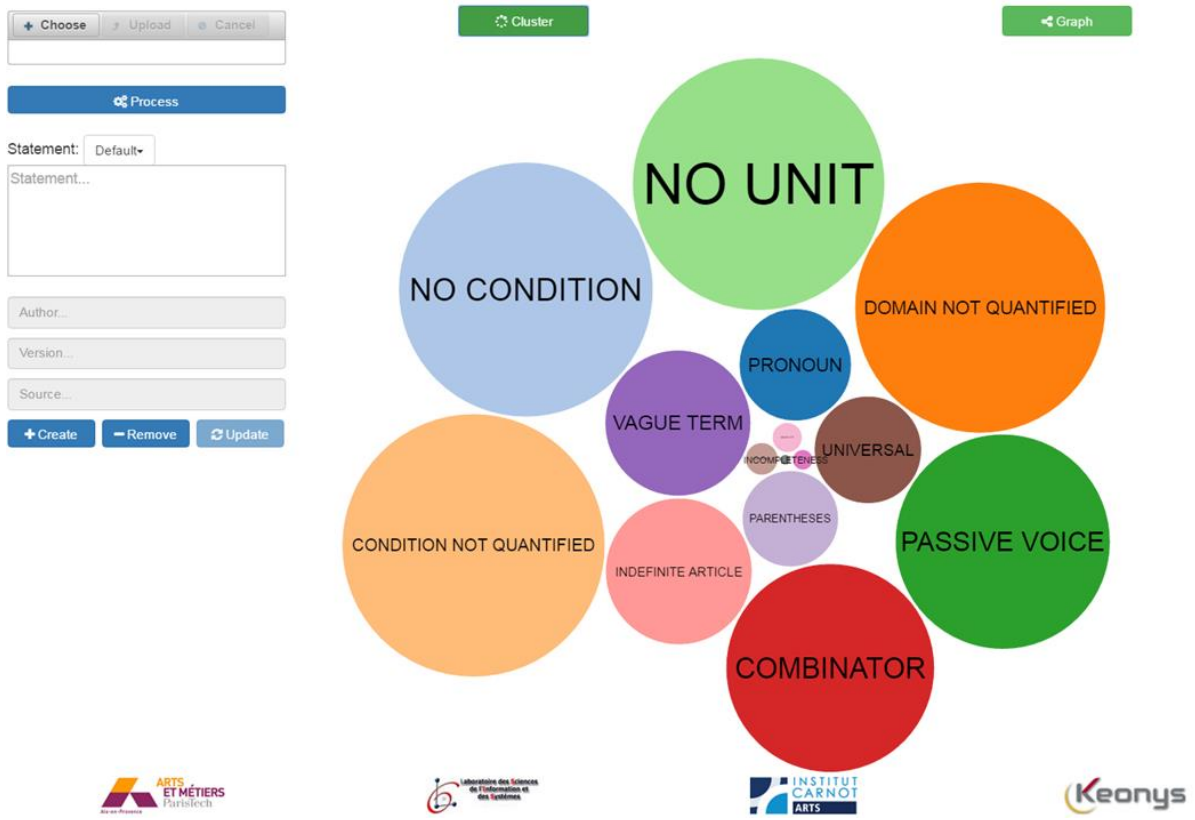


FIGURE 5.5 – Communautés de défauts intrinsèques.

Cette vue synthétise les défauts intrinsèques, mais ne permet pas d'améliorer les exigences. Pour accéder à une liste d'exigences (Fig. 5.6), l'analyste doit cliquer sur l'une des communautés. Une analyse manuelle doit alors être conduite de manière linéaire. Pour chaque exigence, l'analyste clique sur l'énoncé, les défauts sont alors surlignés dans le champ d'édition à partir duquel il peut modifier l'exigence et la mettre à jour dans la base de données. Si le résultat est un faux positif alors il peut ôter l'exigence de la liste via le bouton « *Remove* ».

Enfin, même si ce n'est pas l'objectif de l'environnement numérique, l'analyste peut spécifier de nouvelles exigences en utilisant l'éditeur. Ce dernier signale alors les défauts pendant la rédaction en surlignant les termes proscrits. Des patrons de rédaction sont aussi mis à disposition des utilisateurs qui ne sont pas familiers avec les bonnes pratiques de rédaction.

Identifier les défauts relationnels (FT2.2)

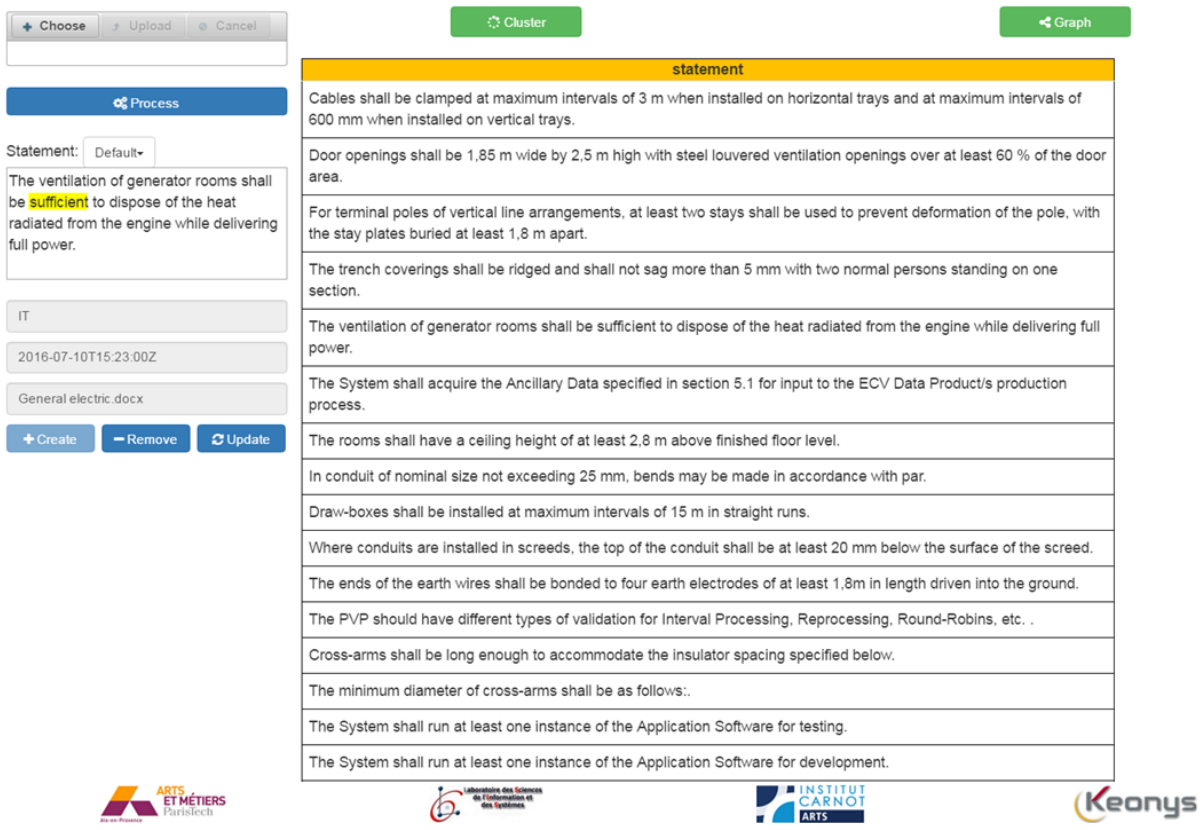


FIGURE 5.6 – Liste de défauts intrinsèques appartenant à la communauté des termes vagues.

L'état de l'art a montré que les propositions faites pour identifier les redondances sont soit basées sur une métrique de similarité qui ne tient pas compte de la synonymie des mots [Natt och Dag et al., 2002, Ilyas and Küng, 2009], soit sur un vocabulaire pré-défini [Fraga et al., 2015] facilitant l'analyse. Par ailleurs, la seule solution [Coatanéa et al., 2013] qui vise à identifier les exigences contradictoires s'appuie sur un processus d'analyse qui est purement déterministe et incomplet.

Il existe sept types de contradictions (Fig. 5.7) dans le langage naturel [de Marneffe et al., 2008]. Selon l'auteur, elles peuvent être regroupées dans deux catégories selon qu'elles soient faciles ou difficiles à détecter. Les contradictions dues à une négation, un antonyme ou une valeur numérique sont jugées comme étant facile à détecter. Les contradictions autres sont jugées difficiles à détecter.

Les trois premières sont jugées faciles à détecter car elles ne nécessitent pas une compréhension totale de la phrase. Il « suffit » par exemple d'identifier deux termes antonymes et que les deux phrases soient relativement similaires pour qu'il y ait une contradiction due à un antonyme. Comme nous le verrons ci-dessous, nous avons eu la même intuition. Cependant, dans un domaine technique comme l'ingénierie, nous montrerons qu'il est souvent très difficile de savoir si oui ou non deux énoncés sont contradictoires sans avoir les connaissances métiers nécessaires.

Ici, nous nous limitons à la détection des contradictions qui sont dues à une négation, un antonyme, ou une valeur numérique. Nous faisons trois hypothèses pour que deux énoncés R_1 et R_2 soient contradictoires.

Hypothèse 1. Si R_1 contient une négation et R_2 ne contient pas de négation, ou *vice-versa*, et si $\text{Sim}(R_1,R_2)$ [3] > X avec X une valeur seuil, alors il existe une contradiction de négation entre R_1 et R_2 ; e.g.

– R_1 : *The car shall* ***be*** *made of steel.*
– R_2 : *The car shall* ***not be*** *made of steel.*

Hypothèse 2. Si R_1 contient un nom, un verbe, un adjectif, ou un adverbe qui est un antonyme d'un

3. $\text{Sim}(R_1,R_2)$ est la similarité entre l'exigence R_1 et l'exigence R_2

ID	Type	Text	Hypothesis
1	Antonym	Capital punishment is a catalyst for more crime.	Capital punishment is a deterrent to crime.
2	Negation	A closely divided Supreme Court said that juries and not judges must impose a death sentence.	The Supreme Court decided that only judges can impose the death sentence.
3	Numeric	The tragedy of the explosion in Qana that killed more than 50 civilians has presented Israel with a dilemma.	An investigation into the strike in Qana found 28 confirmed dead thus far.
4	Factive	Prime Minister John Howard says he will not be swayed by a warning that Australia faces more terrorism attacks unless it withdraws its troops from Iraq.	Australia withdraws from Iraq.
5	Factive	The bombers had not managed to enter the embassy.	The bombers entered the embassy.
6	Structure	Jacques Santer succeeded Jacques Delors as president of the European Commission in 1995.	Delors succeeded Santer in the presidency of the European Commission.
7	Structure	The Channel Tunnel stretches from England to France. It is the second-longest rail tunnel in the world, the longest being a tunnel in Japan.	The Channel Tunnel connects France and Japan.
8	Lexical	The Canadian parliament's Ethics Commission said former immigration minister, Judy Sgro, did nothing wrong and her staff had put her into a conflict of interest.	The Canadian parliament's Ethics Commission accuses Judy Sgro.
9	Lexical	In the election, Bush called for U.S. troops to be withdrawn from the peacekeeping mission in the Balkans.	He cites such missions as an example of how America must "stay the course."
10	WK	Microsoft Israel, one of the first Microsoft branches outside the USA, was founded in 1989.	Microsoft was established in 1989.

FIGURE 5.7 – Les différents types de contradictions linguistiques [de Marneffe et al., 2008].

nom, d'un verbe, d'un adjectif, ou d'un adverbe de R_2, ou *vice-versa*, et si Sim(R_1,R_2) > X avec X une valeur seuil, alors il existe une contradiction d'antonyme entre R_1 et R_2 ; e.g.

– R_1 : *The car shall be **fast**.*
– R_2 : *The car shall be **slow**.*

Hypothèse 3. Si R_1 et R_2 contiennent des valeurs numériques qui appartiennent à la même dimension physique, et si Sim(R_1,R_2) > X avec X une valeur seuil, alors il existe une contradiction numérique entre R_1 et R_2 ; e.g.

– R_1 : *The structure shall withstand **40 daN**.*
– R_2 : *The structure shall withstand **600 N**.*

Ainsi, un degré de similarité élevé est une condition nécessaire mais pas suffisante pour que deux exigences soient contradictoires. La présence d'une négation, d'un antonyme ou de valeurs numériques sert alors à distinguer le type de contradiction. Si la contradiction n'appartient à aucun de ces trois types mais que les deux exigences sont très similaires, alors elles sont potentiellement redondantes.

La première étape doit donc permettre de mesurer la similarité entre chaque paire d'exigences. Il existe une multitude de métriques de similarité appliquées au domaine du traitement du langage naturel et de la recherche d'informations [Gomaa and Fahmy, 2013]. Cosinus (*Cosine*, en anglais) est néanmoins l'une des fonctions les plus utilisées. Cette dernière s'est également avérée être la plus pertinente en ingénierie des exigences [Natt och Dag et al., 2002, Ilyas and Küng, 2009], nous l'avons donc préférée pour nos travaux. Habituellement, avant d'appliquer la fonction de similarité Cosinus à chacune des paires d'exigences, il faut construire un espace métrique. Un espace métrique est un couple (X, d) formé d'un ensemble X et d'une distance d sur X tel que :

– $\forall x, y \in X, d(x, y) = 0$ si et seulement si $x = y$ (séparation)
– $\forall x, y \in X, d(x, y) = d(y, x)$ (symétrie)
– $\forall x, y, z \in X, d(x, z) \leq d(x, y) + d(y, z)$ (inégalité triangulaire)

Pour représenter un texte dans un espace métrique on utilise le modèle vectoriel (*vector space model*, en anglais). Un modèle vectoriel modélise le texte comme une matrice qui est communément connue sous le nom de *Term-Document Matrix* (TDM). Dans une matrice TDM, chaque ligne correspond à un terme distinct, tandis que chaque colonne correspond à un document. Le concept de document sert

de dénominateur commun à un livre, une page Web, un e-mail, une phrase, etc. Ici, les documents correspondent aux exigences, tandis que les termes correspondent aux lemmes. On utilise donc une matrice Exigences-Lemmes dans laquelle chaque ligne est une exigence et chaque colonne est un lemme. Pour construire cette matrice, il faut commencer par créer un sac de mots (*bag of words*, en anglais). Ici, le sac de mot est l'union [4] des vecteurs de lemmes clés stockés dans chaque objet « Exigence ». Par exemple, si l'on considère deux exigences R_1 et R_2, et les deux vecteurs de lemmes clés $\vec{L}(R_1)$ et $\vec{L}(R_2)$ associés, alors l'union $\vec{L}(R_1) \cup \vec{L}(R_2)$ est :

- R_1 = The aircraft shall be inspected every 30 hours.
- $\vec{L}(R_1)$ = {aircraft, shall, be, inspect, every, hour}
- R_2 = The team shall check the airplane every 50 hours.
- $\vec{L}(R_2)$ = {team, shall, check, airplane, every, hour}
- $\vec{L}(R_2) \cup \vec{L}(R_2)$ = **{aircraft, shall, be, inspect, every, hour, team, check, airplane}**

La taille du sac de mots est alors égale au nombre de lemmes clés uniques contenus dans tous les documents soumis à l'environnement numérique. Dans la suite du texte, le sac de mot est appelé vocabulaire. Ainsi, la dimension de l'espace vectoriel engendré est égale à la taille du vocabulaire.

Ce vocabulaire sert à construire la matrice Exigences-Lemmes. En parcourant la liste des exigences, si une exigence R_i contient le lemme L_i, alors l'entrée (i, j) de la matrice est mise à jour avec la valeur 1. La matrice Exigences-Lemmes de l'exemple précédent est la suivante :

$$\begin{bmatrix} & aircraft & shall & be & inspect & every & hour & team & check & airplane \\ R_1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ R_2 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{5.4}$$

En pratique, une seule grosse matrice Exigences-Lemmes de dimension $M * N$ est construite, avec M le nombre d'exigences et N la taille du vocabulaire. Le cas échéant, la similarité entre deux exigences R_i et R_j est calculée comme le Cosinus des deux vecteurs associés $\vec{L}(R_i)$ et $\vec{L}(R_j)$. Par exemple, la similarité entre R_1 et R_2 est :

$$Sim(R_1, R_2) = \cos(\vec{L}(R_1), \vec{L}(R_2)) = \frac{\vec{L}(R_1) \cdot \vec{L}(R_2)}{\|\vec{L}(R_1)\|_2 \, \|\vec{L}(R_2)\|_2} = \frac{3}{\sqrt{6}\,\sqrt{6}} = 0.5 \tag{5.5}$$

Les deux exigences R_1 et R_2 ont un score de similarité faible (50 %), alors même qu'elles sont très similaires, et ce, parce que le concept de synonymie entre *aircraft* et *airplane*, ainsi qu'entre *inspect* et *check* n'est pas considéré.

Une solution pour contourner ce problème est d'enrichir chacun des vecteurs de lemmes clés avec les synonymes issus d'un thésaurus. Par exemple, dans l'exemple précédent, les deux exigences R_1 et R_2 sont transformées comme suit :

- R_1 = The aircraft shall be inspected every 30 hours.
- $\vec{L}(R_1)$ = {aircraft, **airplane**, shall, be, inspect, **check**, every, hour}
- R_2 = The team shall check the airplane every 50 hours.
- $\vec{L}(R_2)$ = {team, shall, check, **inspect**, airplane, **aircraft**, every, hour}

Le nouveau score de similarité $Sim(R_1, R_2)$ est alors égal à :

$$Sim(R_1, R_2) = \cos(\vec{L}(R_1), \vec{L}(R_2)) = \frac{\vec{L}(R_1) \cdot \vec{L}(R_2)}{||\vec{L}(R_1)|| \; ||\vec{L}(R_2)|} = \frac{7}{\sqrt{8}\,\sqrt{8}} = 0.875 \tag{5.6}$$

Il est important de ne rajouter que les synonymes qui sont présents dans l'une ou l'autre des exigences. En effet, si l'on ajoute l'ensemble des synonymes de chaque lemme clé, alors on introduit du bruit qui dégrade la qualité de la mesure. Par exemple, si le terme *organization* qui est un synonyme de *system* dans l'exigence R_1 est ajouté dans le vecteur $\vec{L}(R_1)$ alors qu'il n'est pas dans R_2, la similarité serait dégradée comme suit :

4. On rappelle que l'union $E_1 \cup E_2$ de deux ensembles $E_1 = \{A, C, D\}$ et $E_2 = \{A, B, D\}$ est $\{A, B, C, D\}$.

– R_1 = The system shall be inspected every 30 hours.
– $\vec{L}(R_1)$ = {system, **organisation**, shall, be, inspect, **check**, every, hour}
– R_2 = The team shall check the airplane every 50 hours.
– $\vec{L}(R_2)$ = {team, shall, check, **inspect**, airplane, **aircraft**, every, hour}

$$Sim(R_1, R_2) = \cos(\vec{L}(R_1), \vec{L}(R_2)) = \frac{\vec{L}(R_1) \cdot \vec{L}(R_2)}{||\vec{L}(R_1)||\ ||\vec{L}(R_2)|} = \frac{5}{\sqrt{8}\ \sqrt{8}} = 0.625 \tag{5.7}$$

Le fait d'étendre chacun des lemmes avec les synonymes qui ne sont pas présents dans la seconde exigence a non seulement un impact sur le produit scalaire, mais aussi sur la norme car un lemme clé peut avoir un seul synonyme, tandis qu'un autre lemme clé peut en avoir une dizaine. Cela nous oblige à construire un modèle vectoriel pour chaque paire d'exigences, plutôt qu'un seul gros modèle vectoriel comme c'est généralement le cas.

Tous les synonymes sont extraits du thésaurus WordNet en tenant compte de la catégorie syntaxique du lemme clé recherché, car le même lemme peut avoir la même catégorie – un nom ou un verbe, par exemple –, et donc avoir des synonymes différents (Fig. 5.8).

Noun

- S: (n) **bank** (sloping land (especially the slope beside a body of water)) *"they pulled the canoe up on the bank"; "he sat on the bank of the river and watched the currents"*
- S: (n) depository financial institution, **bank**, banking concern, banking company (a financial institution that accepts deposits and channels the money into lending activities) *"he cashed a check at the bank"; "that bank holds the mortgage on my home"*
- S: (n) **bank** (a long ridge or pile) *"a huge bank of earth"*
- S: (n) **bank** (an arrangement of similar objects in a row or in tiers) *"he operated a bank of switches"*
- S: (n) **bank** (a supply or stock held in reserve for future use (especially in emergencies))
- S: (n) **bank** (the funds held by a gambling house or the dealer in some gambling games) *"he tried to break the bank at Monte Carlo"*
- S: (n) **bank**, cant, camber (a slope in the turn of a road or track; the outside is higher than the inside in order to reduce the effects of centrifugal force)
- S: (n) savings bank, coin bank, money box, **bank** (a container (usually with a slot in the top) for keeping money at home) *"the coin bank was empty"*
- S: (n) **bank**, bank building (a building in which the business of banking transacted) *"the bank is on the corner of Nassau and Witherspoon"*
- S: (n) **bank** (a flight maneuver; aircraft tips laterally about its longitudinal axis (especially in turning)) *"the plane went into a steep bank"*

Verb

- S: (v) **bank** (tip laterally) *"the pilot had to bank the aircraft"*
- S: (v) **bank** (enclose with a bank) *"bank roads"*
- S: (v) **bank** (do business with a bank or keep an account at a bank) *"Where do you bank in this town?"*
- S: (v) **bank** (act as the banker in a game or in gambling)
- S: (v) **bank** (be in the banking business)
- S: (v) deposit, **bank** (put into a bank account) *"She deposits her paycheck every month"*
- S: (v) **bank** (cover with ashes so to control the rate of burning) *"bank a fire"*
- S: (v) count, bet, depend, swear, rely, **bank**, look, calculate, reckon (have faith or confidence in) *"you can count on me to help you any time"; "Look to your friends for support"; "You can bet on that!"; "Depend on your family in times of crisis"*

FIGURE 5.8 – Ensembles de synonymes pour le terme *bank*.

Après avoir calculé le score de similarité pour les $\frac{n*(n-1)}{2}$ paires d'exigences, il reste à déterminer si chacune des paires est une contradiction ou non et, si elle en est une, de quel type de contradiction il s'agit.

Le thésaurus WordNet fournit les antonymes d'un lemme, lesquels sont nécessaires à l'identification des contradictions dues à des antonymes. Ainsi, étant donné :

- une paire d'exigences (R_i, R_j),
- les vecteurs étendus $\vec{L}(R_i)$ et $\vec{L}(R_j)$ de lemmes clés $L_{i?}$ et $L_{j?}$ associés aux exigences R_i et R_j :
  - $\vec{L}(R_i) = \{L_{i1}, L_{i2}, ..., L_{im}\}$
  - $\vec{L}(R_j) = \{L_{j1}, L_{j2}, ..., L_{jn}\}$
- les vecteurs étendus $\vec{A}(R_i)$ et $\vec{A}(R_j)$ des antonymes clés $A_{i?}$ et $A_{j?}$ associés aux vecteurs $\vec{V}(R_i)$ et $\vec{V}(R_j)$:
  - $\vec{L}(R_i) = \{L_{i1}, L_{i2}, ..., L_{im}\}$
  - $\vec{A}(R_i) = \{A_{i1}, A_{i2}, ..., A_{ik}\}$
  - $\vec{L}(R_j) = \{L_{j1}, L_{j2}, ..., L_{jn}\}$
  - $\vec{A}(R_j) = \{A_{j1}, A_{j2}, ..., A_{jl}\}$

Si un antonyme appartenant à $\vec{A}(R_i)$, respectivement $\vec{A}(R_j)$, appartient aussi à $\vec{L}(R_j)$, respectivement $\vec{L}(R_i)$, alors les deux exigences sont potentiellement contradictoires. On peut illustrer l'opération avec l'exemple suivant :

- R_1 = The car shall be fast.
- $\vec{L}(R_1)$ = {car, shall, be, **fast**}
- $\vec{A}(R_1)$ = {..., **slow**, ...}
- R_2 = The car shall be slow.
- $\vec{L}(R_2)$ = {car, shall, be, **slow**}
- $\vec{A}(R_2)$ = {..., **fast**, ...}

Dans cet exemple, le lemme *fast* est un antonyme de *slow*, donc il y a potentiellement une contradiction due à un antonyme. Il est nécessaire de rechercher les antonymes pour chacune des exigences d'une paire car le thésaurus WordNet n'est pas symétrique. Par exemple, *fast* peut être un antonyme de *slow* sans que *slow* soit un antonyme de *fast*.

Pour savoir si une exigence contient ou non une négation, nous analysons le graphe sémantique de chaque exigence. Par exemple, le graphe sémantique de la figure 5.9 contient une dépendance de type *neg* correspondant à une négation. Ainsi, si pour une paire d'exigences donnée le score de similarité est très élevé, et qu'une des deux exigences contient une négation, alors il y a potentiellement une contradiction due à une négation.

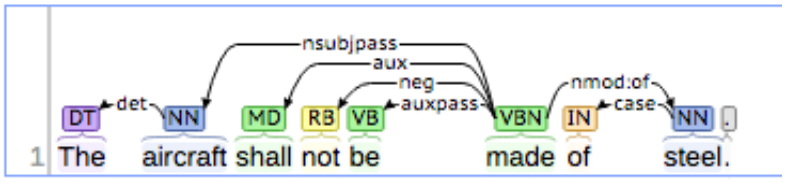


FIGURE 5.9 – Dépendance de négation « neg » dans un graphe sémantique.

L'identification des contradictions dues à des valeurs numériques sont traitées de la même manière. Si pour une paire d'exigences donnée le score de similarité est très élevé, et que les deux exigences contiennent une valeur numérique, alors il y a potentiellement une contradiction numérique. Les valeurs numériques sont identifiables via l'annotation syntaxique « CD » (Fig. 5.4). Pour savoir si les valeurs numériques sont des grandeurs physiques, nous récupérons la valeur du nœud source et interrogeons un dictionnaire d'unités physiques. Par exemple, nous savons que la valeur *bars* dans l'exemple de la figure 5.4 stockée dans le nœud source de la dépendance *num* correspond à une unité physique.

Notifier (FT24) et supprimer (FT26) les défauts relationnels

Une fois identifiés, les défauts inter-exigences sont représentés sous la forme d'un graphe non connexe dans lequel les sommets correspondent aux exigences et les arêtes aux défauts relationnels

(Fig. 5.10).

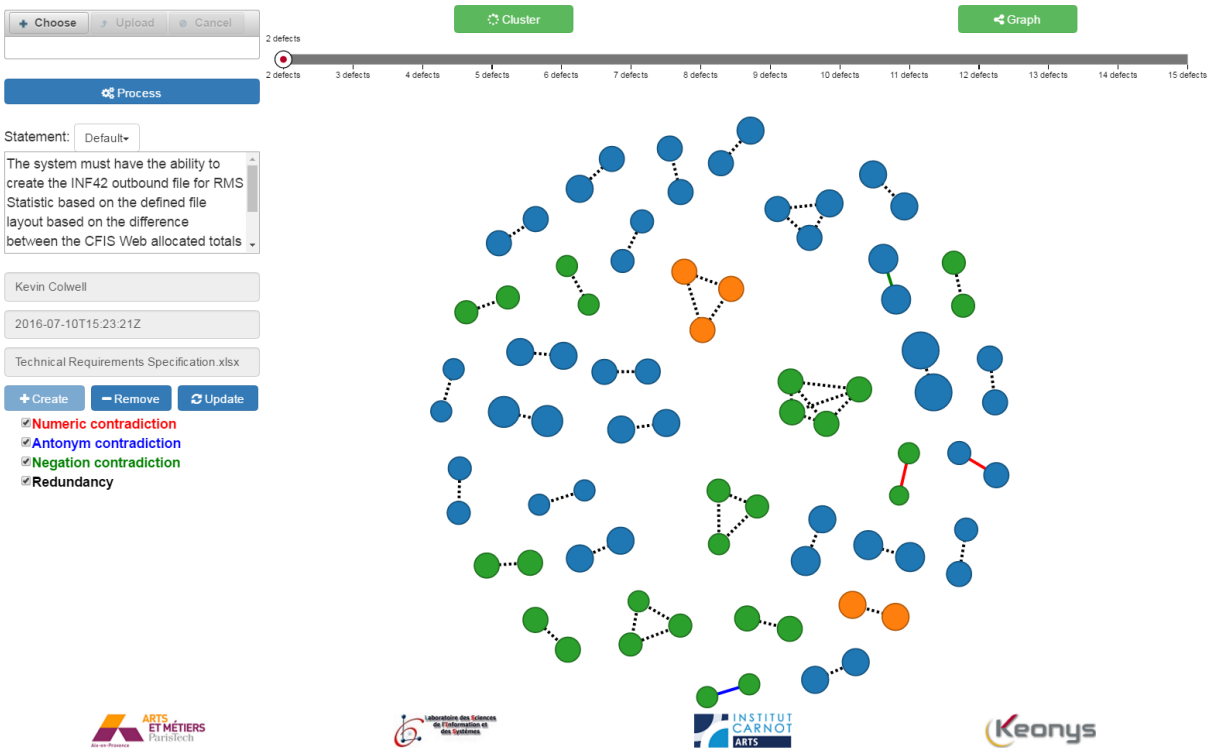


FIGURE 5.10 – Graphe non connexe qui représente les défauts inter-exigences.

D'un point de vue métaphorique, la couleur des sommets correspond à un document prescriptif, tandis que la couleur des arêtes, multiples ou non, représente les différents types de contradictions et les redondances. Enfin, le diamètre des sommets illustre le nombre de défauts inhérents à une exigence. Plus le diamètre est grand, plus il y a de défauts. Si le nombre de défauts relationnels est trop important, alors l'analyste peut filtrer les données selon deux entrées. D'abord, il peut filtrer selon le type de défaut relationnel en cochant une ou plusieurs *checkbox*. De plus, il peut filtrer selon le nombre de défauts intrinsèques à chaque exigence en déplaçant le curseur vers la droite. Le dessin du graphe utilise un algorithme de modèle de force dirigée basé sur la simulation physique. Dans le modèle de force, les sommets sont considérés comme des particules chargées, tandis que les arêtes sont analogues à des ressorts mécaniques. Ainsi, on peut calculer la force de répulsion F_{rep} et la force d'attraction F_{attr} qui sont appliquées à chaque sommet. Si on déplace les sommets de manière itérative d'une distance δ qui est une fonction de F_{rep} et F_{attr}, alors on peut trouver une position d'équilibre statique pour laquelle les nœuds voisins son proches les uns des autres avec un minimum d'intersections [Kaufmann and Wagner, 2001]. L'algorithme de force dirigée que nous utilisons est celui implémenté dans la bibliothèque implémenté dans la bibliothèque JavaScript de visualisation des données D3.js.

5.4 Expérimentation

Pour évaluer les performances des deux modules destinés à identifier les défauts de qualité intrinsèques et relationnels, nous avons réalisé deux expérimentations avec des données industrielles.

Vérification de l'identification des défauts intrinsèques

La première chose à vérifier c'est le module d'identification des défauts intrinsèques susceptibles de rendre une exigence ambigüe. Les règles implémentées doivent permettre d'identifier trois types de

défauts intrinsèques : les exigences incomplètes, celles qui contiennent un connecteur tel que *and* ou *or* qui fait que l'exigence n'est pas singulière, et les exigences qui contiennent un terme vague.

Bien que l'évaluation de la détection des termes vagues et des connecteurs soit faisable, elle n'a pas été réalisée car nous n'obtiendrions aucune information constructive. En effet, une validation empirique reviendrait à vérifier que les règles permettant de détecter la présence de mots proscrits stockés dans un dictionnaire ont été correctement implémentées. Si les mots sont dans le dictionnaire, alors il n'y a pas de surprise, les résultats sont bons ; s'ils sont absents du dictionnaire, alors on les y rajoute. Quelques auteurs mènent ce genre de tests, mais cela n'a pas beaucoup d'intérêt pour ce type d'application, si ce n'est pour révéler d'éventuels oublis lors de la construction des dictionnaires de termes à proscrire.

Nous avons néanmoins souhaité évalué notre proposition pour détecter les exigences incomplètes. Pour cela, nous avons utilisé une spécification de la mission spatiale ROSETTA, un document PDF disponible en ligne sur le site de l'agence spatiale européenne dont les exigences spécifient la navette spatiale et ses interfaces. L'analyse automatique a permis d'extraire 1073 exigences.

Parmi les 1073 exigences, le prototype a détecté que 1017 d'entre elles ne contiennent pas de condition d'applicabilité. Ceci n'est pas une surprise car, en pratique, les personnes qui sont en charge de spécifier les produits n'explicitent pas la condition dans laquelle l'exigence est applicable. Ainsi, toutes les exigences sont applicables pour tous les scénarios imaginables. Parmi les 56 exigences régies par une condition d'applicabilité, seulement 7 d'entre elles sont des résultats faux positifs, lesquels sont principalement dus à la présence du terme *even if* et à des informations qui ont été malencontreusement classifiées comme des exigences.

Concernant la présence ou non d'un domaine de tolérance mesurable, parmi les 1073 exigences, le prototype a détecté que seulement 81 d'entre elles en possèdent un. Comme pour la condition d'applicabilité, ceci n'est pas stupéfiant, le constat est le même pour la trentaine de spécifications techniques dont nous disposons : les analystes prescrivent rarement des exigences avec une propriété quantitative. Dans certains cas comme dans l'exigence « *The autonomy implemented in software shall be designed using deterministic algorithmic techniques.* » cela ne pose pas de problème, car la vérification se fait au moyen d'une inspection et ne nécessite pas de mesure pour être vérifiée. Néanmoins on peut rester dubitatif quand 90 % des exigences d'un système spatial ne sont pas mesurables, ce qui rend les exigences ambiguës comme celle ci : « *Pure force thrusters for orbit manoeuvres shall have minimum residual torques.* ». Même si ce type d'exigence est compréhensible, cela ne veut pas dire que son interprétation est unique. Est-ce que *minimum* c'est 10, 100, 1000, 10 000 N.m ? Dans ce cas là, il est très difficile de planifier les coûts et les délais requis pour développer la solution associée sans critère de performance.

Vérification de l'identification des défauts relationnels

Après avoir testé notre proposition d'identification des défauts intrinsèques, nous avons évalué celle permettant de détecter les redondances et les contradictions potentielles entre exigences.

Les redondances et les contradictions sont extrêmement rares quand le document prescriptif a été rédigé par une seule personne et qu'il prescrit seulement quelques dizaines ou une centaine d'exigences. Pour qu'il y ait des défauts relationnels entre les exigences, il faut que les conditions soient propices : plusieurs centaines ou milliers d'exigences ; plusieurs agents pour spécifier les exigences ; toutes les exigences doivent spécifier le même produit ; etc. Malheureusement, nous ne disposons pas d'un corpus de documents prescriptifs qui puisse être représentatif d'une telle situation.

Pour évaluer la qualité de notre proposition, nous ne mesurons pas le rappel et la précision à partir de documents prescriptifs préalablement analysés à la main. Pour un corpus de 1000 exigences de taille suffisamment raisonnable pour espérer trouver des défauts relationnels, cela nécessiterait de faire $\frac{1000(1000-1)}{2} = 499\,500$ comparaisons manuelles ! Ainsi, nous proposons de faire le travail inverse, nous appliquons notre proposition à un ensemble d'exigences, après quoi nous vérifions que chaque paire d'exigences retournée est probablement redondantes ou contradictoires.

Les données utilisées sont identiques à celles de l'expérimentation 4.4, soit deux documents prescriptifs non structurés auxquels nous avons ajouté une feuille de calcul Excel. Au total, parmi les 1618

exigences comparées, notre prototype a retourné 96 défauts relationnels dont 6 contradictions et 90 redondances.

Numérique :
Req src : The MTBF of the File Store Data Archive shall be more than 1 month. **Req tgt** : The MTBF of the On-line Data Archive shall be more than 1 month.
Req src : In the case of switchboards with uninsulated conductors accessible from the back, a clear space of at least 1,2 m shall be provided between the back and sides of the board and the wall. **Req tgt** : In the case of switchboards with uninsulated conductors which are exposed and accessible from the back a clear space of at least 1,2 m shall be provided at the back.
Négation
Req src : Where sizes are not specified, the largest bolt or screw that will fit into the hole shall be used. **Req tgt** : Where holes exist in equipment to be fixed, bolts and fixing screws as specified shall be used.
Req src : Where conductor sizes are not specified, the minimum conductor sizes shall be used. **Req tgt** : The minimum conductor size to be used shall be 4 mm2.
Req src : This earth conductor shall not be installed external to the flexible conduit but within the conduit with the other conductors. **Req tgt** : An earth conductor shall be installed in all non-metallic flexible conduit.
Antonyme
Req src : The System shall have interfaces with the following systems for automatic acquisition of L2 Data : a) WARP for METOP-A,B,C ASCAT (see 4.1.3), b) LPRM for SSM/I, Windsat (see 4.1.3). **Req tgt** : The System shall have interfaces with the following systems for manual acquisition of L2 Data : a) WARP for METOP-A,B,C ASCAT (see 4.1.3), b) LPRM for SSM/I, Windsat (see 4.1.3).

TABLE 5.3 – Contradictions potentielles dues à des valeurs numériques physiques, des négations et des antonymes.

Parmi les 6 contradictions (Tab. 5.3), 2 d'entre elles sont dues à des valeurs numériques physiques, 3 à des négations et 1 à un antonyme. Parmi les contradictions numériques, les exigences sont bien similaires et associées à des grandeurs mesurables : une durée en mois et une longueur en mètres. Dans la première contradiction numérique on constate que le sujet (*File Store Data Archive*) de la première exigence est différent de celui de la première exigence (*On-line Data Archive*), on pourrait donc être tenté de s'assurer que deux exigences sont contradictoires si elles ont le même sujet. Or sans vocabulaire contrôlé, rien ne nous assure que deux agents utiliseront le même sujet dans des documents différents. Les contradictions potentielles dues à des négations sont aussi pertinentes. Les paires d'exigences sont relativement similaires et au moins une des exigences contient une négation. Enfin, les deux dernières exigences sont très similaires et le nom *automatic* est un antonyme de *manual* il est donc probable que ces deux exigences soient contradictoires, même si, ici, la probabilité semble très faible puisqu'on se doute que l'auteur a voulu spécifier les deux modes d'acquisition des données.

Un extrait aléatoire des 90 redondances potentielles est donné dans le tableau 5.4. En lisant chacune des paires, on constate que la métrique de similarité retourne des résultats pertinents pour des longueurs de phrases variables. Une revue manuelle des 90 redondances a permis de s'assurer qu'il n'y avait aucune contradiction due à une négation, une valeur numérique ou un antonyme parmi les résultats obtenus. Cela ne nous garantit pas qu'il n'y a aucune contradiction. Par exemple, il peut y avoir des contradictions dont la détection nécessite des connaissances spécifiques au produit, au projet, voire au monde en général. En effet, quand nous lisons la liste des contradictions et redondances potentielles, dans la plupart des cas, nous sommes incapables d'affirmer à 100 % que les paires retournées sont bien conflictuelles. Cela illustre toute la difficulté de l'exercice. Par exemple, il est très probable que la redondance entre les exigences « *Brass screws bolts and nuts shall be used to fix galvanised equipment.* » et « *Where holes*

exist in equipment to be fixed, bolts and fixing screws as specified shall be used. » ne sont probablement pas contradictoires et cela se reflète dans le score de similarité qui est égal à 0.76. Au contraire, on ne peut pas être certain que les exigences « *The system must allow for the automatic creation of sample rosters for each RMS Service Location based on the roster from the prior sample period (r 4.2.2).* » et « *The system must allow the creation and maintenance of sample rosters for each RMS Service Location and sample period.* » sont redondantes, même si, ici, on peut espérer qu'elles ne le sont pas puisque les documents ont été rédigés par la même personne. Au final, la revue manuelle nous incite à conclure que les 96 paires d'exigences signalées comme étant potentiellement contradictoires ou redondantes méritent d'être révisées manuellement par un analyste qui possède les connaissances contextuelles et techniques nécessaires à son analyse.

Req src : The System shall produce the Merged Active-only ECV Data Product in accordance with the Product Specification, as specified in [PSD], for dissemination to Data Users. **Req tgt** : The System shall produce the Merged Active-Passive ECV Data Product in accordance with the Product Specification, as specified in [PSD], for dissemination to Data Users.
Req src : The contractor shall supply and install all additional supporting timbers in the roof space as required. **Req tgt** : The Contractor shall supply and install all additional supporting timbers required.
Req src : Conduit ends shall be threaded and fitted with a coupling and brass plug. **Req tgt** : The conduit ends shall be threaded and fitted with a coupling and brass plug.
Req src : He shall further liaise with the Main Contractor for the provision of holes and access through the structure and finishes. **Req tgt** : He shall further liaise with the Main contractor to verify the position of holes and access routes through the structure and finishes.
Req src : Conductors installed in vertical wire-ways shall be secured at intervals not exceeding 5m to support the weight of the conductors. **Req tgt** : Where vertical duct lengths exceed 5m, conductors installed in the channels shall be secured at intervals not exceeding 5m to support the weight of the conductors.
Req src : The system must allow the Service Location RMS coordinator role to set the Service Location specific Random Moment Sample (RMS) selection variables for a minimum of the following values : sample hit counts per cost pool , control member percentage (r 4.3.1), sample response deadline (r.4.3.7), pop-up timer (r 4.3.13), pre-moment review time (r 4.3.4). **Req tgt** : The system must allow the state RMS coordination role to set the statewide Random Moment Sample (RMS) selection variables for a minimum of the following values : control member percentage (r 4.3.1), sample response deadline (r.4.3.8), pop-up timer (r 4.3.12).
Req src : The system must allow for the automatic creation of sample rosters for each RMS Service Location based on the roster from the prior sample period (r 4.2.2). **Req tgt** : The system must allow the creation and maintenance of sample rosters for each RMS Service Location and sample period.

TABLE 5.4 – Extrait des redondances potentielles entre les exigences.

5.5 Synthèse

Bilan de notre méthode de fiabilisation des exigences

Dans ce chapitre, nous cherchons une méthode qui puisse permettre à un analyste de fiabiliser l'interprétation des exigences afin que les estimations réalisées par les experts et les décisions stratégiques prises par le manageur soient respectivement objectives et rationnelles.

Nous avons identifié trois solutions pour améliorer l'interprétation des exigences. Les deux premières consistent à supprimer les défauts intrinsèques aux exigences et les défauts relationnels entre plusieurs exigences. Parmi les défauts intrinsèques nous considérons la présence de termes vagues et de connecteurs logiques, ainsi que les exigences incomplètes. Les termes vagues et les connecteurs logiques sont identifiés grâce à un dictionnaire de termes proscrits, tandis que la complétude d'une exigence est vérifiée par un ensemble de règles appliquées à un graphe sémantique. La figure 5.11 résume la méthode mise en œuvre pour détecter les défauts intrinsèques et relationnels. La troisième solution pour fiabiliser les exigences est d'utiliser un contexte d'exigences voisines. Cette dernière solution est présentée dans le chapitre suivant car elle fournit un service à l'expert qui réalise les estimations, tandis qu'ici on s'intéresse au service fourni à l'analyste.

Lors des tests menés pour évaluer notre proposition de détection des défauts intrinsèques nous avons constaté que la plupart des exigences ne sont pas conformes à la notion de complétude telle qu'elle est

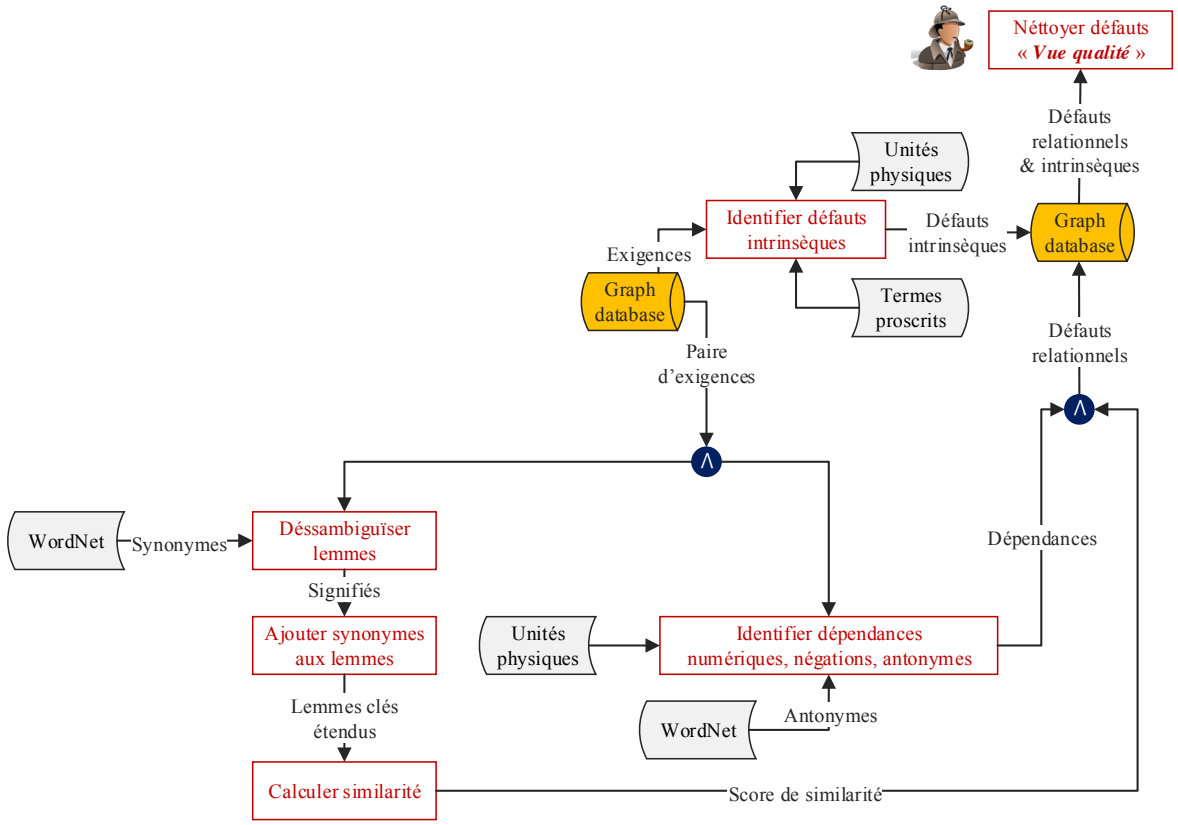


FIGURE 5.11 – Diagramme de flux d'informations qui résume la méthode de fiabilisation des exigences.

définie par le concept de Property Based Requirements. En effet, 90 % d'exigences collectées n'ont pas de domaine de tolérance mesurable. On peut alors s'interroger quant à l'utilité d'une telle analyse alors même que les documents prescriptifs sont majoritairement de mauvaise qualité. Cette fonctionnalité doit être réservée pour analyser des documents prescriptifs spécifiant des sous-systèmes dont le niveau de détail est assez fin pour que toutes les exigences soient quantifiées. La détection des redondances et des contradictions, elle, donne des résultats encourageants car toutes les paires d'exigences retournées sont potentiellement conflictuelles. Nous insistons sur l'aspect potentiel, et ce, pour deux raisons. D'abord, parce que la liste des paires d'exigences conflictuelles peut contenir des résultats faux positifs. Enfin, parce que nous sommes incapables d'affirmer que chaque contradiction ou redondance en est réellement une, car cela nécessite non seulement une connaissance du contexte, mais aussi des connaissances techniques dont nous ne disposons pas.

Qu'est-ce qui est nouveau ou différent ?

La littérature relative à l'analyse des défauts de qualité est très riche, en particulier pour l'analyse analytique des défauts intrinsèques à une exigence.

La première chose qui est différente dans notre méthode c'est la manière de détecter les défauts relationnels. D'une part, la fonction de similarité exploite le thésaurus WordNet qui nous permet de considérer la synonymie entre les termes. D'autre part, pour identifier les contradictions, nous analysons le graphe sémantique associé à chaque exigence afin d'identifier des négations et des valeurs numériques qui peuvent potentiellement induire une contradiction. Nous sommes aussi capables de vérifier qu'une exigence est complète selon la notion de Property Based Requirements, c'est-à-dire qu'elle possède au moins un domaine de tolérance mesurable et, si nécessaire, une condition d'applicabilité.

Nous proposons aussi une couche visuelle permettant d'avoir une vue synthétique des défauts de qualité, alors que les solutions de l'état de l'art se concentrent principalement sur l'analyse et un peu sur

la visualisation, mais n'intègrent jamais les deux dans un même environnement.

Quelles sont les limites et les perspectives d'amélioration ?

Aujourd'hui, nous sommes capables d'identifier trois types de contradictions potentielles – négation, antonyme et numérique –, mais il en reste encore quatre selon la typologie de de Marneffe et al. [2008]. Nous n'avons pas de recommandation spécifique pour les identifier, cela demeure un problème fondamental dans la compréhension de la langue. Il est néanmoins indéniable qu'il faille mettre en place une couche de connaissances. La définition de cette dernière peut être manuelle en construisant des ontologies, ou automatique en exploitant les techniques d'apprentissage profond alimentés par les données libres et celles qui sont stockées sur les serveurs des entreprises. En effet, la compréhension d'un texte technique passera par l'analyse de données internes qui reflètent le vocabulaire et les connaissances manipulées par une entreprise.

Pour améliorer notre proposition, en particulier notre fonction de similarité, il serait pertinent de tester les derniers modèles de représentation de textes [Mikolov et al., 2013a,b, Pennington et al., 2014] adaptés à des énoncés textuels courts. En effet, les métriques de similarités classiques utilisées en recherche d'informations n'exploitent pas les co-occurrences entre les termes. Plus généralement, elles ne se soucient pas du contexte qui environne chacun des termes clés qui servent à mesurer la similarité entre des énoncés.

Chapitre 6

EXPLORER LES EXIGENCES

6.1 Introduction

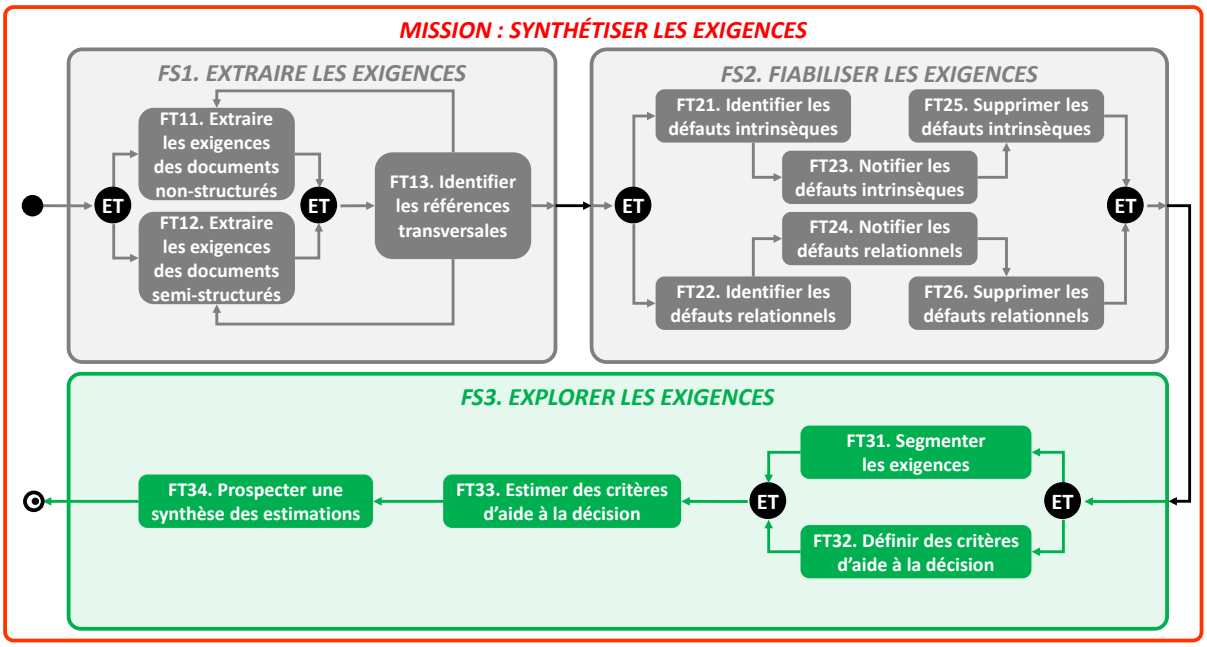


FIGURE 6.1 – FS3 - Explorer les exigences.

Les exigences dénuées des principaux défauts de qualité engendrent un espace prescriptif à partir duquel un manageur doit prendre des décisions stratégiques informées. Par exemple, une décision de « GO vs NO GO » lors d'une réponse à un appel d'offre. Ici, nous cherchons donc une solution pour que le manageur puisse s'informer vi-à-vis de l'espace prescriptif dans l'objectif de prendre des décisions stratégiques avisées.

La première section 6.2 de ce chapitre synthétise et critique les solutions proposées pour supporter la prise de décisions en ingénierie des exigences. Les propositions sont classifiées selon 3 catégories. Premièrement, il y a les techniques d'aide à la décision mono- et multi-critères. Deuxièmement, on trouve les méthodes de résolution d'un problème sous contraintes. Enfin, il y a les techniques de fouille de données.

La section 6.3 propose un ensemble d'éléments qui permettent d'explorer l'espace prescriptif en vue de prendre des décisions informées. Pour commencer, une nouvelle fonction de classification basée sur l'apprentissage machine classifie les exigences dans des catégories qui correspondent à des métiers (mécanique, électronique, etc.). Le thésaurus WordNet et l'ontologie ConceptNet 5 sont ensuite exploités

pour enrichir le graphe d'exigences avec des relations sémantiques et conceptuelles. Ce graphe enrichi sert à segmenter les exigences en communautés pour assurer le passage à l'échelle. Une plateforme collaborative de sondage est mise à disposition des experts pour estimer les critères d'aide à la décision qui sont associés à chacune des communautés d'exigences. Enfin, un tableau de bord permet au manageur de prospecter l'espace prescriptif via des résumés statistiques qui synthétisent les estimations réalisées par les experts.

Diverses expérimentations nécessaires à la section d'une technique de segmentation des exigences sont présentées en section 6.4.

Enfin, la section 6.5 synthétise les différences et les nouveautés de notre approche, ainsi que les limites et les perspectives d'amélioration.

6.2 État de l'art des solutions d'exploration des exigences

L'activité incontournable de décision en ingénierie des exigences est la priorisation. Prioriser c'est ordonner un ensemble d'objets par priorité. En ingénierie des exigences, on priorise les exigences. En effet, quand les exigences d'un client sont trop nombreuses, le fournisseur doit sélectionner un sous-ensemble dont l'implémentation maximise la satisfaction du client selon les budgets et délais impartis.

La récente revue de littérature faite part Achimugu et al. [2014a] énumère pas moins de 49 techniques de priorisation des exigences, chacune avec une popularité plus ou moins établie. La figure 6.2 résume les 15 méthodes les plus populaires selon le nombre de citations.

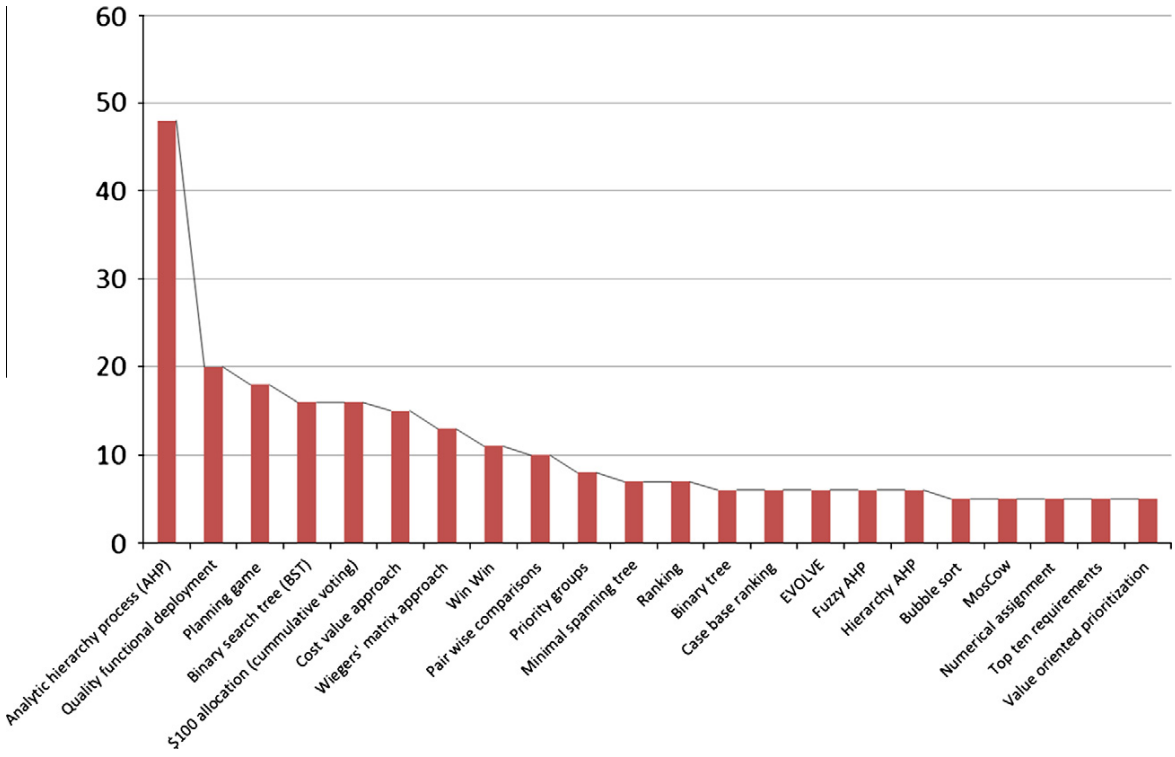


FIGURE 6.2 – Les 15 techniques de priorisation d'exigences les plus populaires en termes de citations [Achimugu et al., 2014a].

Toutes ces techniques de priorisation peuvent être rangées au sein de trois grandes approches : (1) les techniques d'aide à la décision mono- et multi-critères, (2) les techniques d'optimisation, et (3) les techniques de fouille de données.

Les techniques d'aide à la décision

La plupart des techniques d'aide à la décision ont servi à prioriser un ensemble d'exigences. La méthode la plus élémentaire, *In or Out*, est employée par des agents qui passent en revue chacune des exigences et décide celles qui doivent être développées pour la prochaine version du produit [Wiegers and Beatty, 2013]. Si la liste se limite à 10 exigences par parties prenantes, alors on parle de : *Top-Ten Requirements* [Berander and Andrews, 2005].

Une autre solution, celle qui est implémentée dans tous les outils de gestion des exigences, consiste à attribuer une priorité qualitative ordinale (*low*, *medium*, *high*) à chacune des exigences. Pour objectiver l'estimation, l'échelle à trois niveaux (*Three-level scale*, en anglais) raffine la définition de la priorité en conjuguant deux critères : l'importance – le client a ou n'a pas besoin de cette exigence –, et l'urgence – le client peut ou ne peut pas attendre la prochaine version du produit [Wiegers and Beatty, 2013]. Ainsi, par exemple, si une exigence est importante et urgente, alors elle a une priorité très élevée. Ces techniques peuvent servir à ordonner les exigences, mais leur subjectivité ne facilite pas la prise de décisions stratégiques. Une autre alternative est d'évaluer le coût et la valeur d'une exigences afin de les représenter dans un diagramme coût-valeur (Fig. 6.3).

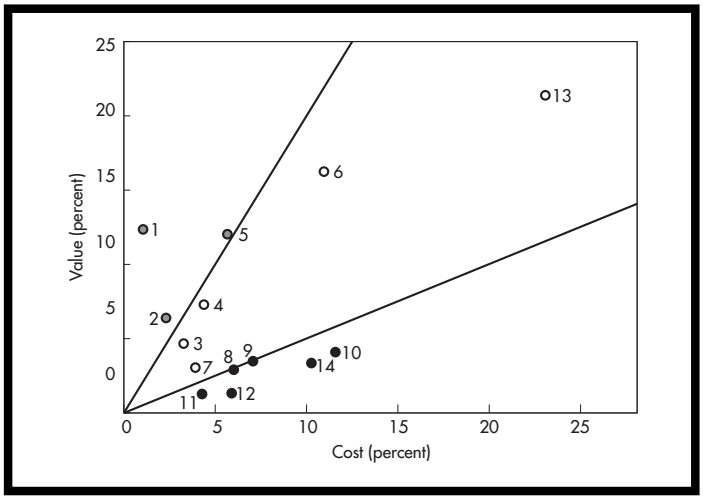


FIGURE 6.3 – Diagramme coût-valeur pour prioriser les exigences [Karlsson, 1996].

Quand il est trop difficile de sélectionner ou d'ordonner directement les exigences, on peut avoir recours à des comparaisons par paires (*pairwise comparison*, en anglais). Si un ordre de préférence est établi pour chacune des $\frac{n(n-1)}{2}$ paires d'exigences, alors les exigences sont ordonnées par ordre de priorité. Les techniques basées sur les comparaisons par paires sont appréciées, car elles donnent des résultats robustes [Karlsson, 1996] et sont faciles à appliquer. Le tri à bulles (*Bubblesort*, en anglais) nécessite aussi $\frac{n(n-1)}{2}$ comparaisons [Karlsson et al., 1998], tandis qu'un arbre binaire de recherche (*Binary Search Tree*, en anglais) est un peu moins lent à exécuter $O(n\ log\ n)$, mais la priorisation d'un ensemble de 1000 exigences nécessite approximativement 10 000 comparaisons [Laurent et al., 2007]. L'une des méthodes les plus populaires, *Analytic Hierarchy Process* (AHP) [Saaty, 1987], exploite la comparaison par paires dans un cadre analytique rigoureux qui minimise la subjectivité. Sa particularité d'être construite sur des fondamentaux mathématiques assure un minimum de robustesse [Karlsson et al., 1998] et établit une confiance avec son utilisateur. La méthode d'analyse multi-critères AHP a servi dans de nombreux domaine, dont le PLM [Zhang et al., 2013b], en particulier pour prioriser les exigences [Perini et al., 2009, Karlsson et al., 2004]. La méthode AHP reste toutefois non extensible, difficile et lente à mettre en œuvre [Ahl, 2005, Karlsson et al., 2007]. Malgré ces défauts, la comparaison par paires reste un élément fondamental pour de nombreuses méthodes [Achimugu et al., 2014c], lesquelles sont applicables sur quelques dizaines d'exigences.

D'autres alternatives analytiques ont été proposées pour s'affranchir des laborieuses comparaisons

par paires. La matrice *Quality Function Deployment* (QFD) de Cohen [1995] et la matrice de priorisation de Wiegers and Beatty [2013] sont fréquemment utilisées pour décider en ingénierie des exigences. Cette dernière propose un cadre de priorisation des exigences selon trois critères – le coût, le risque, et la valeur – associés à chaque exigence. Ramzan et al. [2011] proposent de regrouper les clients et les experts dans la méthode de priorisation *Value Based Intelligent Requirement Prioritization*, laquelle se base sur la logique floue. Enfin, Liaskos et al. [2011] proposent une méthode orientée vers les buts pour modéliser les préférences et prioriser les alternatives. Ces méthodes sont moins laborieuses que celles basées sur des comparaisons par paires, mais elles sont toujours limitées à une petite centaine d'exigences.

En plus des techniques d'aide à la décision mono- ou multi-critères, il y a les techniques de négociation. Par exemple, dans la méthode des 100 $, aussi appelée vote cumulatif (*cumulative voting*, en anglais), chacune des parties prenantes dispose d'une somme fictive de 100 $ dont elle alloue une partie à chacune des exigences. C'est l'une des méthodes les plus simples et rapides à mettre en œuvre sur un nombre très limité d'exigences [Ahl, 2005]. Chatzipetrou et al. [2010] vont un peu plus loin en proposant une plate-forme d'analyse statistique multivariée *Compositional Data Analysis* (CoDA) pour explorer les données d'un vote cumulatif. Enfin, le procédé de négociation *WinWin* [Ruhe et al., 2002, 2003, Boehm and Kitapci, 2006] permet de prioriser les exigences en tenant non seulement compte des conditions qui font qu'une partie prenante est gagnante, mais aussi des conflits et des alternatives de solution.

Ces techniques d'analyse mono- ou multi-critères d'aide à la décision ont divers avantages et inconvénients. Étant donné qu'elles sont généralement peu sophistiquées, elles sont relativement simples à utiliser ou automatiser. Cependant, ces techniques ont plus d'inconvénients que d'avantages :

- **Non extensibles.** Ces méthodes ne permettent pas de prioriser plusieurs centaines ou milliers d'exigences [Perini et al., 2007, Tonella et al., 2013, McZara et al., 2014]. Elles ne sont donc pas directement exploitables dans notre contexte.
- **Non démontrées.** On constate aussi qu'il existe une pléthore de méthodes pour prioriser les exigences, mais très peu sont outillées [Achimugu et al., 2014c] et validées sur une volumétrie qui correspond aux pratiques industrielles.
- **Pas de critères universels.** Les techniques d'aide à la décision nécessitent de définir un ou plusieurs critères permettant d'estimer qualitativement ou quantitativement la priorité des exigences. Il n'y a pas de critères universels. Bien que le coût, la valeur, le risque et le temps soient quasi-omniprésents, la plupart d'entre-eux sont définis selon le contexte. Par exemple, Azar et al. [2007] utilisent des critères dérivés du monde des affaires tels que les ventes, le marketing, la concurrence, la stratégie, etc. Les critères diffèrent donc selon l'entreprise, le projet, le produit, etc. Dans leur état de l'art, Riegel and Doerr [2015] ont identifié pas moins de 280 critères utilisés pour prioriser les exigences. Rares sont donc les entreprises qui accepteront une solution rigide dont les critères sont pré-définis par le fournisseur de l'outil.
- **Subjectives.** Ces techniques d'aide à la décision font aussi face à des difficultés liées à l'incertain, à la mauvaise intuition humaine. Cela est encore plus vrai pour les méthodes qui n'utilisent pas de comparaisons par paires. Les estimations, souvent qualitatives, des exigences sont difficilement réalisables [Lehtola et al., 2004, Svensson et al., 2011] et relèvent alors plus de l'intuition, de l'opinion. De plus, quand la méthode devient trop compliqué, les décideurs perdent confiance vis-à-vis des résultats [Lehtola and Kauppinen, 2004].
- **Explosion combinatoire.** Pour pallier à cette subjectivité, diverses méthodes d'aide à la décision nécessitent de réaliser des comparaisons par paires. Cependant, ces techniques font face à une explosion combinatoire.
- **Non utilisables.** Pour conclure sur ces méthodes d'aide à la décision, on peut s'interroger quant à la capacité d'un expert à estimer objectivement une exigence sans avoir une vue contextuelle des exigences voisines telles que : les exigences prescrites par les documents externes applicables référencés, les exigences appartenant à la même thématique, etc. Par exemple, toutes les solutions fondées sur les comparaisons sont parfois inexploitables, car l'utilisateur peut être incapable de donner un ordre de préférence entre deux énoncés isolés de leur contexte.

Les techniques de résolution d'un problème sous contraintes

En informatique, le problème de la prochaine sortie (*Next Release Problem*, en anglais), cherche à maximiser la satisfaction d'un ou plusieurs clients tout en minimisant les ressources nécessaires au développement du produit. Autrement dit, l'objectif est d'implémenter un maximum d'exigences tout en limitant la consommation de ressources, en particulier les ressources financières. Pour résoudre ce problème dont les objectifs divergent, les solutions s'appuient sur les techniques d'optimisation multi-objectif. Par exemple, le problème d'optimisation formulé par Durillo et al. [2009] est le suivant :

- Conditions :
  - Un ensemble $C = \{c_1, c_1, ..., c_m\}$ de m clients.
  - Un ensemble $R = \{r_1, r_2, ..., r_n\}$ de n exigences.
  - Le coût r_j dérivé des ressources nécessaires pour satisfaire l'exigence j.
  - La valeur v_{ij} qui représente l'importance de l'exigence j pour le client i.
- Fonctions objectives :
  - Trouver $R^{'}$ un sous-ensemble d'exigences qui minimise le coût et maximise la satisfaction des clients, tel que :
  - minimise $f_1 = \sum_{r_i \in R^{'}} r_i$
  - maximise $f_2 = \sum_{i=1}^{n} c_i \sum_{r_j \in R^{'}} v_{ij}$

Alors que le résultat d'un problème d'optimisation avec une seule fonction objective est un point, ici, la résolution du problème multi-objectif permet d'identifier un ensemble de points dominants appelé front de Pareto (Fig. 6.4.)

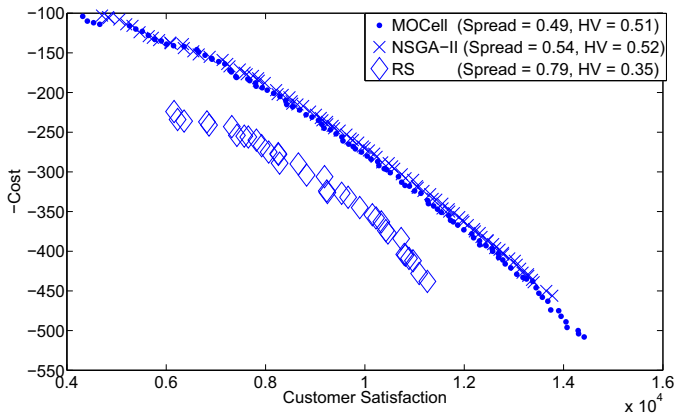


FIGURE 6.4 – Fronts de Pareto résultant de la recherche d'un sous-ensemble « optimisé » d'exigences [Durillo et al., 2009].

Divers auteurs ont testé différentes méthodes d'optimisation telles que les colonies de fourmis [Sagrado et al., 2015] ou les algorithmes génétiques [Zhang et al., 2007, Tonella et al., 2010, 2013] pour aider les utilisateurs à sélectionner un sous-ensemble « optimisé » d'exigences.

Ces techniques d'optimisation sont particulièrement adaptées quand les exigences sont décrites comme des objectifs à maximiser ou minimiser et non pas comme des intervalles de contraintes [Roda, 2013] auxquels les propriétés d'une solution doivent appartenir. Par exemple, plutôt que de choisir l'une des solutions appartenant à l'espace engendré par les exigences, l'approche de *Value Driven Design* [Cheung et al., 2010] cherche la solution optimale d'une fonction objective qui modélise la valeur (Fig. 6.5).

Outre les techniques d'optimisation, des propositions reformulent l'exercice de la priorisation comme un problème sous contraintes. Palma et al. [2011] combinent un solveur de *Satisfiability Modulo Theory* (SMT) et la technique des comparaisons par paires. Un solveur SMT sert à résoudre des problèmes

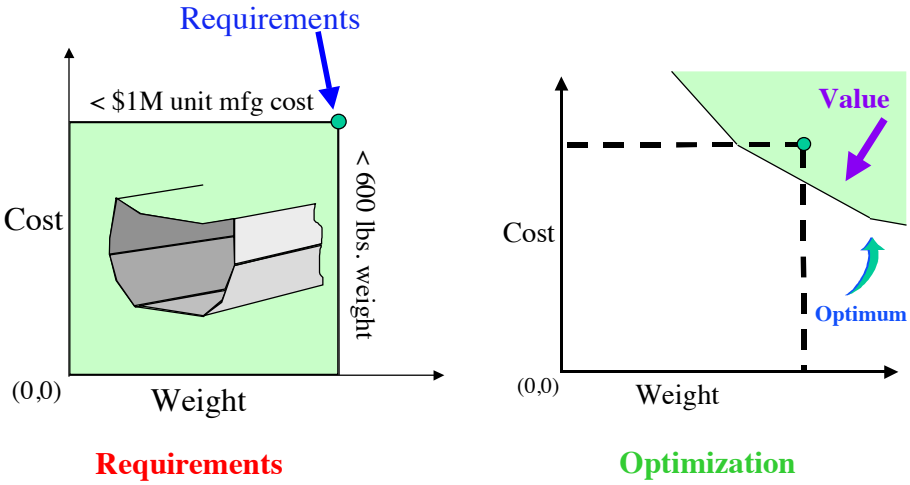


FIGURE 6.5 – Utilisations d'objectifs plutôt que d'exigences [Collopy, 2007].

de décision qui sont formulés à partir d'équations issues de la logique du premier ordre. Dans leur approche, les auteurs utilisent un graphe dont les sommets correspondent aux exigences, et les arêtes modélisent les relations de priorité qui dictent l'ordre d'implémentation des exigences. Ces contraintes d'implémentation sont ensuite exploitées par le solveur SMT qui établit une liste ordonnée d'exigences. Si plusieurs solutions satisfont les contraintes d'implémentation, alors les exigences conflictuelles sont manuellement priorisées par un utilisateur à l'aide de comparaisons par paires. Cette proposition vise à assurer le passage à l'échelle en évitant de comparer toutes les paires une par une.

McZara et al. [2014] prétendent aussi s'attaquer au problème de la volumétrie des exigences, mais leurs données de test varient entre 60 et 140 exigences, ce qui ne correspond pas aux problèmes pratiques de l'industrie. Leur solution combine un solveur de contraintes avec les techniques de traitement du langage naturel. Dans un premier temps, l'outil propose 50 mots clés extraits des exigences. L'utilisateur choisit quelques uns d'entre eux, lesquels servent à générer des communautés d'exigences via une métrique de similarité lexicale. Ainsi, plutôt que d'ordonner toutes les exigences, pour chacun des clusters, l'utilisateur priorise les exigences grâce à des comparaisons par paires. Ces contraintes de priorité alimentent un solveur SMT qui génère l'ordre final entre toutes les exigences.

Les techniques d'optimisation sont pertinentes quand les interdépendances entre les exigences – les priorités d'implémentation, par exemple – sont explicites. Dans les solutions qui sont bâties sur une base de données, si le processus d'ingénierie des exigences est rigoureusement mis en œuvre, alors certaines interdépendances doivent être identifiables. Néanmoins, même cette approche a ses limites. Comme le souligne Regnell et al. [2008], trouver les interdépendances parmi une dizaine d'exigences c'est possible, mais cela devient infaisable quand les quantités sont plus importantes. Enfin, dans notre étude de cas, l'identification des interdépendances à partir de documents textuels hérités d'un donneur d'ordres est encore plus difficile.

Les techniques de fouille de données

Au cours des dix dernières années, Perini et al. [2013] ont fourni un travail considérable pour adapter leur système d'aide à la priorisation basé sur les cas [Avesani et al., 2003] au problème de la priorisation des exigences. Leur solution *Case-Based Ranking (CBRank)* combine les ordres de préférence établis par les utilisateurs avec des techniques d'apprentissage machine qui approximent l'ordre final des exigences. Plus précisément, comme l'illustre la figure 6.6, CBRank est un processus itératif en trois étapes :

– Échantillonnage de paires : un échantillon de paires d'exigences est automatiquement généré.
– Priorisation de l'échantillon : le décideur établit un ordre de priorité pour chacune des paires de

l'échantillon.

– Apprentissage de la priorité : à partir de l'échantillon de paires d'exigences qui a été ordonné précédemment, une fonction de classement (*ranking function*, en anglais) approxime les préférences inconnues, ce qui permet d'obtenir une approximation de l'ordre pour toutes les exigences. La fonction de classement est apprise avec l'algorithme de classement RankBoost [Freund et al., 2003] qui est habituellement utilisé pour ordonner les résultats d'une requête dans un moteur de recherche.

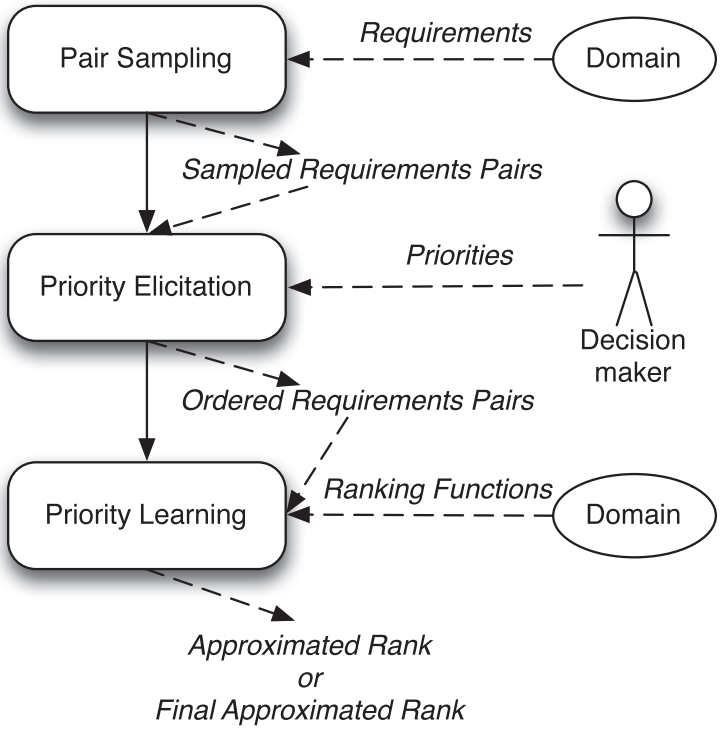


FIGURE 6.6 – Le processus de priorisation avec la solution d'apprentissage machine CBRank. [Perini et al., 2013].

Bien que cette proposition soit intéressante, les auteurs indiquent qu'avec un effort raisonnable, un utilisateur peut traiter jusqu'à 100 exigences avec CBRank. Encore une fois, un ensemble de 100 exigences ne correspond pas à la taille réelle des données industrielles.

Comme l'indique Regnell et al. [2008], le passage de la centaine au millier d'exigences nécessite de segmenter les exigences afin de travailler à un niveau d'abstraction plus élevé. Divers acteurs ont donc essayé de créer des communautés d'exigences.

Issu d'un long travail de recherche, l'outil Pirogov [Duan and Cleland-Huang, 2007a] permet de tracer et trier les exigences. Pour cela, les exigences sont segmentées en communautés (Fig. 6.7). Une exigence peut appartenir à une ou plusieurs communautés. Les clients évaluent la valeur de chaque communauté et pondèrent l'importance relative pour chaque paire de communautés. Enfin, une fonction objective génère une liste ordonnée des exigences. L'algorithme de segmentation agglomère de manière hiérarchique les exigences similaires. La notion de similarité est un calcul de probabilité basé sur la fréquence des termes clés d'une exigence. Par ailleurs, puisque les termes clés peuvent être exprimés différemment, une liste de synonymes et d'acronymes préalablement définie sert à étendre les mots clés originaux. Cette approche est intéressante, mais la définition des communautés se limite à l'aspect linguistique, en particulier la similarité lexicale entre les mots.

Sannier and Baudry [2012] et Sannier [2013] ont testé plusieurs méthodes de découpage en communautés pour identifier des thèmes dans un corpus de documents prescriptifs. Cependant, l'étude ne traite pas les exigences, mais se limite à l'analyse des sommaires des documents.

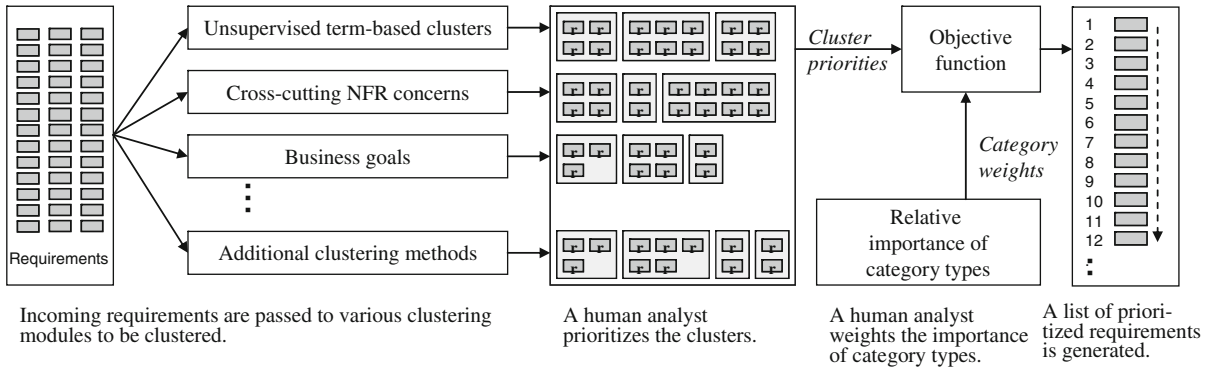


FIGURE 6.7 – Priorisation d'exigences basée sur des communautés [Duan and Cleland-Huang, 2007a].

Fitsilis et al. [2010] utilisent les techniques d'analyse de réseaux sociaux pour prioriser un ensemble d'exigences préalablement élucidées conformément à un modèle de données orienté graphe dans lequel les relations sont explicitement créées par les parties prenantes. Azmeh et al. [2014] s'appuient aussi sur l'historique des opérations d'un environnement collaboratif pour segmenter les parties prenantes et les exigences. L'utilisation de réseaux sociaux et de forums pour développer et gérer les exigences est une alternative aux documents. De tels environnements facilitent le travail collaboratif et la gestion de configuration selon une méthode de travail qui est presque toute aussi informelle que les méthodes basées sur les documents. L'analyse est néanmoins facilitée car les données sont modélisées dès le départ selon un formalise semi-structuré – e.g. graphe RDF – ou structuré – e.g. base de données (No)SQL.

Les méthodes de découpage en communautés on été utilisées pour d'autres objectifs. Par exemple, Azmeh et al. [2013] utilisent les langages du Web sémantique et le concept de treillis pour segmenter les utilisateurs finaux qui prescrivent leurs besoins dans un environnement collaboratif. L'identification de communautés d'utilisateurs a de multiples applications. Par exemple, le fournisseur peut prioriser les exigences en fonction de l'activité des parties prenantes.

Khan and Mahmood [2012] segmentent un graphe représentant un modèle d'exigences orienté vers les buts, tandis que Ferrari et al. [2013] segmentent les exigences d'un document prescriptif afin d'en améliorer la structure en réorganisant les exigences par groupes d'intérêts.

Tandis que les techniques de segmentation introduites ci-dessus cherchent à regrouper les exigences dans des communautés qui ne sont pas préalablement définies, d'autres appliquent des techniques qui classifient chaque exigence dans une catégorie prédéfinie. L'utilisateur peut alors explorer un gros volume de données en filtrant une ou plusieurs catégories.

De nombreux travaux ont permis de classifier les exigences selon qu'elles soient fonctionnelles ou non. Les propositions les plus simples s'appuient sur des patrons syntaxiques [Lamar, 2009], la détection de termes clés appartenant à une ontologie de domaine [Moser et al., 2011], à un schéma de facettes [Hochmüller, 1997], la fréquence des termes clés [Cleland-Huang et al., 2006], ou la similarité entre des mots clés [Mahmoud, 2015]. Les techniques d'apprentissage machine supervisé comme celles misent en œuvre dans la section 4.3 ont aussi fait leurs preuves dans cet exercice. Hussain et al. [2008] ont identifié un sous-ensemble de caractéristiques fonctionnelles et non fonctionnelles pour entraîner un arbre de décisions. Casamayor et al. [2010] ont proposé une solution semi-supervisée qui permet à l'utilisateur de confirmer et ou de modifier la classe prévue par la fonction de classification. Une fois classifiées, les observations servent alors à enrichir le corpus d'apprentissage afin d'améliorer les performances de la fonction de classification. Ainsi, avec une telle approche, la construction du corpus d'apprentissage nécessite beaucoup moins d'exemples annotés manuellement. Enfin, Rashwan et al. [2013] ont proposé une taxonomie d'exigences qui catégorise les différents types d'exigences fonctionnelles et non fonctionnelles (fiabilité, efficacité, sécurité, maintenance, etc.). Ces catégories correspondent aux classes parmi lesquelles les exigences sont automatiquement classifiées via une fonction approximée avec l'algorithme de machines à vecteurs supports.

En plus de distinguer les exigences fonctionnelles et non fonctionnelles, l'apprentissage machine a

aussi servi à classifier les exigences selon des métiers. Par exemple, Knauss [2011] a entraîné un modèle Bayésien pour identifier les exigences appartenant au métier de la sécurité. Par ailleurs, Ott [2013] a testé plusieurs algorithmes de classification pour catégoriser les exigences dans des sujets qui sont préalablement définis par une liste de mots clés. La particularité de cette étude est que les exigences qui spécifient les véhicules Mercedes-Benz sont issues de documents prescriptifs écrits en allemand. Dans leurs articles, les auteurs soulignent la difficulté qu'ils ont a regrouper un nombre suffisant d'exemples pour construire un corpus d'apprentissage. C'est pour cette dernière raison que Knauss and Ott [2014] ont étudié différentes approches d'apprentissage supervisé : automatique, semi-automatique et manuel. La première approche est réalisée par un agent qui classifie chaque exigence dans une ou plusieurs catégories. Dans l'approche semi-automatique, l'agent doit approuver ou modifier la révision faite par la fonction de classification. Les exigences revues manuellement servent alors à enrichir le corpus d'apprentissage. Enfin, l'approche automatique ne nécessite aucune intervention de la part de l'agent. Ces travaux se concentrent aussi sur les textes allemands.

Synthèse des méthodes d'exploration des exigences

Cet état de l'art a permis de faire trois constats. Le premier constat c'est que les méthodes d'aide à la décision multi-critères ne sont pas applicables à la volumétrie des exigences que l'on rencontre dans l'industrie. En effet, elles sont utiles pour explorer quelques dizaines d'alternatives, mais ne conviennent pas pour des centaines ou milliers d'exigences. C'est d'ailleurs pour cette raison qu'en pratique, les entreprises mettent en œuvre des méthodes ad-hoc pour prioriser les exigences qu'elles doivent développer [Karlsson et al., 2004].

Le deuxième constat c'est que les méthodes de fouille de données peuvent permettre de segmenter les exigences pour travailler à un niveau d'abstraction supérieur. Cependant, les techniques utilisées jusqu'ici ne se concentrent que sur l'aspect lexical (e.g. *reliable* $\rightarrow$ *safe*), mais aucune d'entre elles ne considère les interdépendances conceptuelles (e.g. *plane* $\rightarrow$ *crash*), ou les relations transversales qui font que deux exigences peuvent être interdépendantes, donc relativement proches, sans être similaires d'un point de vue sémantique.

Enfin, le troisième constat c'est que toutes les propositions relatives à l'exploration d'un gros volume d'exigences sont soit destinées à l'exercice de priorisation, soit à celui de traçabilité, mais aucune ne cherche à explorer un gros volume d'exigences pour mieux s'informer et décider. Or, ici l'objectif n'est ni d'ordonner ni de tracer les exigences, mais d'avoir un aperçu de l'espace prescriptif pour prendre des décisions stratégiques informées.

6.3 Proposition

Pour explorer un gros volume d'exigences, nous proposons de segmenter les exigences en communautés. Nous étudions trois alternatives de segmentation :

- **La classification.** Une fonction de classification est estimée à partir d'un algorithme d'apprentissage machine supervisé. Cette fonction catégorise les exigence dans des catégories métiers (mécanique, électronique, informatique, etc.) prédéfinies.
- **Le partitionnement.** Un algorithme de partitionnement de textes (*text clustering*, en anglais) est utilisé pour segmenter les exigences en communautés. Chaque communauté regroupe des exigences qui ont une certaine affinité sémantique.
- **La détection de communautés.** Issu de la théorie des graphes, un algorithme de détection de communautés est mis en œuvre pour segmenter le graphe engendré par les exigences.

Les communautés qui résultent de la segmentation sont caractérisées par un ensemble personnalisable de critères d'aide à la décision (coût, temps, risques, etc.). Ces critères sont alors estimés via un sondage collaboratif qui exploite et capitalise les connaissances tacites des experts dont le rôle est de représenter chacune des fonctions de l'entreprise.

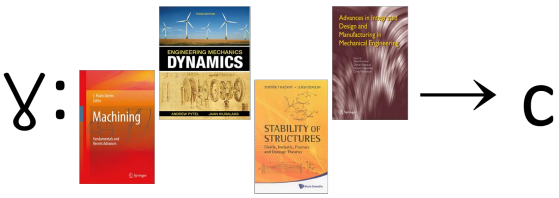
Enfin, un tableau de bord permet au manageur de prospecter l'espace prescriptif via des résumés statistiques qui synthétisent les estimations réalisées par les experts.

Segmenter les exigences (FT31) - alternative 1 : classification automatique

Pour résoudre le problème irrésolu du passage à l'échelle, notre première intuition a été de se dire que tous les experts n'ont pas besoin d'estimer chacune des exigences. En effet, à première vue, on peut s'interroger : pourquoi le responsable du marketing doit estimer les exigences qui concernent la mécanique du produit ? Si chacun des experts estime les exigences qui relèvent de son domaine d'expertise, alors la tâche est nettement moins laborieuse. Les travaux analogues de Ott [2013] et Knauss and Ott [2014] donnent des indices pour résoudre ce problème. En effet, ils ont montré qu'un algorithme d'apprentissage supervisé permet d'apprendre une fonction de classification qui assigne un sujet à chaque exigence. Cependant, leur méthode et leurs résultats ne sont pas portables car ils se concentrent sur des exigences rédigées en allemand.

La première tâche à réaliser est donc de construire un corpus d'apprentissage dont les exemples sont en anglais. Or, comme ces derniers le rappellent, la construction d'un corpus d'apprentissage est laborieux. Ce problème n'est pas inhérent à la classification des exigences, c'est un problème universel à l'application des techniques d'apprentissage machine [Manning et al., 2008]. Ainsi, plutôt que de rédiger ou de collecter un ensemble d'exigences, puis de les annoter avec la classe associée, nous avons cherché à générer automatiquement un corpus d'apprentissage.

Naïvement, on peut supposer qu'il existe une distribution de mots clés qui caractérise chacune des fonctions de l'entreprise. Les mots clés « force » et « déformation » sont par exemple plus fréquents en « mécanique » qu'en « électronique ». Si on connait les termes discriminants, alors on peut les utiliser pour entraîner un algorithme d'apprentissage supervisé.



Mechanics	100
Stress	40
Strain	38
Force	25
Acceleration	20
...	...

FIGURE 6.8 – Définition naïve du problème de classification des exigences dans des catégories correspondant à des fonctions de l'entreprise.

Pour identifier ces termes discriminants, nous supposons qu'un corpus de livres bien choisis (discipline, taille, généricité, etc.) contient les termes clés d'une catégorie (Fig. 6.8). Par exemple, une dizaine de livres qui traitent de la mécanique doit permettre d'identifier les termes qui caractérisent la mécanique. Ainsi, pour chaque métier, nous sélectionnons un ensemble de livres. Chacun des livres est traité afin d'extraire les phrases qu'il contient. Pour cela, nous réutilisons les fonctions de traitement du langage naturel développées précédemment. L'outil Apache Tika extrait le contenu textuel des livres, tandis que la boîte à outil Stanford CoreNLP segmente le texte en phrases. Chacune des phrases est automatiquement annotée avec la catégorie à laquelle elle appartient. Par exemple, toutes les phrases extraites du corpus de livres de mécanique sont annotées avec la catégorie mécanique. Ici, nous nous limitons à quatre fonctions de l'entreprise, soit quatre catégories : mécanique (ME), électronique (EE), informatique (CS) et fiabilité, disponibilité, maintenance et sécurité (RAMS). La méthode est néanmoins

extensible à d'autres fonctions, lesquelles doivent rester génériques au risque de dégrader les performances. Par exemple, il est beaucoup plus difficile de distinguer les énoncés qui traitent d'électronique de ceux qui relèvent de l'électrique.

Un livre appartenant au corpus de mécanique ne contient pas que des phrases de mécanique. Il y a de nombreuses phrases qui n'ont rien à voir avec l'une des quatre catégories définies. Certaines sont automatiquement supprimées à l'aide d'expression régulières. C'est par exemple le cas des phrases issues de la bibliographie ou celles qui sont peuplées de formules mathématiques. Le fait que d'autres phrases soient annotées avec une catégorie alors même qu'il n'y a aucune affinité entre elles n'est pas un problème puisque, *in fine*, nous sélectionnons un ensemble de termes discriminants.

L'extraction et l'annotation automatique des phrases donne naissance à un corpus d'apprentissage qui est constitué de 77 481 exemples, avec respectivement 18 135, 19 464, 20 183 et 19 699 pour la catégorie ME, EE, CS et RAMS.

Les phrases annotées ne sont pas directement exploitables par un algorithme d'apprentissage machine. Pour qu'une machine puisse apprendre, il faut la nourrir avec un ensemble de vecteurs numériques : les vecteurs caractéristiques. Nous ne parlerons donc plus de termes, mais de caractéristiques.

Pour calculer ces vecteurs de caractéristiques, il faut commencer par construire un modèle vectoriel $\mathbb{S}$ de dimension initiale $\mathbb{M}*\mathbb{N}$, avec $\mathbb{M}$ égal au nombre de caractéristiques et $\mathbb{N}$ égal au nombre de phrases (Tab. 6.1). Pour cela nous appliquons trois opérations de traitement du langage naturel incluant :

- la segmentation (*tokenization*, en anglais) pour découper les phrases en unités lexicales.
- la transformation des majuscules en minuscules (*case-folding*, en anglais) pour normaliser les termes et ainsi réduire la dimension de l'espace vectoriel.
- la suppression des mots vides de sens (*stop-words removal*, en anglais) tels que *the*, *a*, etc. Cela réduit aussi la dimension de l'espace vectoriel.

Il existe deux autres opérations de pré-traitement qui sont communément utilisées. L'opération de racinisation (*stemming*, en anglais) – e.g. *automate*, *automatic*, *automation* $\rightarrow$ *automat* – n'a généralement pas de valeur ajoutée, mais elle peut aider à compenser le fait que la matrice est creuse et diminuer le temps d'exécution en réduisant la taille du vocabulaire [Manning et al., 2008]. Cependant, l'expérience qui a été menée par Khoo et al. [2006] a montré que l'opération de lemmatisation – e.g. *required*, *requires* $\rightarrow$ *require* – dégrade les performances de la fonction de classification. La racinisation étant encore plus agressive que la lemmatisation, nous ne les avons donc pas appliquées dans notre étude. Après ce pré-traitement, le vocabulaire est néanmoins réduit de 40 729 caractéristiques à 32 885.

Terme 1	**Terme 2**	**...**	**Terme n**
$x_1^{(1)}$	$x_2^{(1)}$	...	$x_n^{(1)}$
0	1	...	1
$x_1^{(3)}$	$x_2^{(3)}$	...	$x_n^{(3)}$
...	...	...	...
$x_1^{(m)}$	$x_2^{(m)}$	...	$x_n^{(m)}$

TABLE 6.1 – Espace de caractéristiques modélisé comme un modèle vectoriel.

Ces opérations de pré-traitement permettent d'obtenir un ensemble de termes qui correspondent aux colonnes du modèle vectoriel. Chaque colonne est une caractéristique. Les lignes, elles, correspondent aux vecteurs de caractéristiques, lesquels représentent les phrases sous une forme numérique tel que : l'entrée $x_i^{(j)}$ est égale à 1 si le terme i est présent dans la phrase j, sinon l'entrée est égale à 0. Dans la plupart des cas, avec un document ou une page Web, la valeur n'est pas binaire, c'est la fréquence d'occurrence pondérée ou non. Cependant, ici, une phrase ne contient que très rarement, voire jamais, deux fois le même mot, donc une entrée binaire est suffisante. L'ensemble des vecteurs de caractéristiques engendrent l'espace de caractéristiques. Dans le domaine de la recherche d'informations, ce type de modèle est appelé *Term-Document Matrix* ou sa transposée *Document-Term Matrix*. La seule différence c'est qu'ici, l'échelle du document est la phrase.

Après cette phase de pré-traitement le modèle vectoriel est une matrice de dimension 77 481 x 32 885. Cet espace des caractéristiques contient beaucoup de termes qui ne sont pas discriminants. Par exemple, toutes les phrases qui ont été automatiquement annotées comme appartenant à une catégorie alors qu'elles ne possèdent aucun terme clés vis-à-vis des catégories ME, EE, CS et RAMS. Pour réduire la dimension de l'espace des caractéristiques on cherche et sélectionne un ensemble de caractéristiques discriminantes. Cette étape s'appelle la sélection de caractéristiques (*feature selection*, en anglais). en plus de supprimer les caractéristiques inutiles, la sélection de caractéristiques permet de supprimer celles qui occasionnent du bruit (*noise feature*, en anglais). Une caractéristique qui apporte du bruit est une caractéristique qui augmente le taux d'erreurs de classification. Par exemple, si le terme « vis » ne caractérise pas la classe électronique, mais que, par chance, toutes les phrases qui contiennent le terme « vis « appartiennent à la classe électronique, alors l'algorithme d'apprentissage peut approximer une fonction de classification qui se trompe lorsqu'elle classifie de nouveaux énoncés non annotés. Quand de telles généralisations se produisent on retrouve le phénomène de sur-apprentissage discuté dans la section 4.4. La sélection d'un sous-ensemble de caractéristiques doit donc supprimer les caractéristiques inutiles et réduire le bruit tout en conservant ou améliorant les performances de la fonction de classification.

La sélection de caractéristiques se fait selon deux grandes approches : les *filtres* et les *wrappers* [Witten et al., 2011]. Les méthodes dites *wrappers* enveloppent (*wrap*, en anglais) l'algorithme d'apprentissage pendant la phase de sélection de caractéristiques. En effet, si par exemple l'algorithme SVM est choisi, alors il sera entraîné autant de fois qu'il existe de combinaisons de caractéristiques. La recherche des différentes combinaisons de caractéristiques consistent à trouver le sous-ensemble de caractéristiques qui permet d'obtenir les meilleures performances de classification. Pour cela, on peut entraîner l'algorithme d'apprentissage en faisant une recherche en avant – on démarre avec l'ensemble vide et on ajoute une caractéristique à chaque entraînement jusqu'à ce que toutes les combinaisons soient réalisées –, en arrière – on démarre avec la totalité des caractéristiques et on en supprime une à chaque entraînement –, ou on utilise une recherche bidirectionnelle – on démarre avec un nombre initial de caractéristiques (la moitié par exemple) et on ajoute ou supprime une caractéristique selon les performances obtenues (amélioration ou dégradation) à chaque entraînement. Ces méthodes dites *wrapper* ont l'avantage de supprimer les caractéristiques qui sont non seulement inutiles mais aussi redondantes. Cependant, elles sont très gourmandes en temps de calcul et ne conviennent pas aux problèmes de grande dimension tels que la classification de textes. Les méthodes dites *filters*, elles, consistent à filtrer quelques caractéristiques avant d'entraîner l'algorithme d'apprentissage. Elles sont relativement simples à implémenter et rapides à exécuter. Elles conviennent donc aux problèmes de classification caractérisés par de nombreuses caractéristiques [Forman, 2007]. Le filtrage des caractéristiques se fait en mesurant une fonction d'utilité $A(t, c)$ pour chaque caractéristique t – ici une caractéristique est un terme – vis-à-vis de chaque catégorie c. Les mesures d'utilité servent alors a établir une liste qui ordonnent les caractéristiques les plus discriminantes en haut et les moins discriminantes en bas. Parmi cette liste ordonnée, l'utilisateur sélectionne un sous-ensemble de caractéristiques en fixant un nombre précis ou un seuil minimal. Ces techniques permettent d'exclure les caractéristiques inutiles, mais celles qui sont redondantes. Il existe plusieurs mesures d'utilité [Forman, 2003], mais les principales sont : la fréquence pondérée ou non, le test du X^2, et l'information mutuelle (*mutual information*, en anglais) qui est aussi connue comme le gain d'information (*information gain*, en anglais) [Manning et al., 2008].

La fréquence, pondérée ou non, consiste à calculer la fréquence d'apparition, pondérée ou non, de chaque terme t dans chaque catégorie c.

L'information mutuelle du terme t vis-à-vis de la catégorie c mesure avec quelle quantité d'information l'absence et la présence du terme t contribue à une bonne classification dans c. Autrement dit, l'information mutuelle mesure l'entropie quand la caractéristique est présente ($e_t = 1$) *versus* l'entropie quand la caractéristique ($e_t = 0$) est absente (Eq. 6.1).

$$I(U;C) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} P(U = e_t, C = e_c) \log \frac{P(U = e_t, C = e_c)}{P(U = e_t)P(C = e_c)} \tag{6.1}$$

Avec :

- $e_t = 1$ quand la phrase contient le terme t
- $e_t = 0$ quand la phrase ne contient pas le terme t
- $e_c = 1$ quand la phrase appartient à la catégorie C
- $e_c = 0$ quand la phrase n'appartient pas à la catégorie C

L'équation 6.1 peut être réécrite en replaçant chacun des termes $P(U = e_t, C = e_c)$ tels que :

- $P(U = 0, C = 0) = \frac{N_{00}}{N}$
- $P(U = 0, C = 1) = \frac{N_{01}}{N}$
- $P(U = 1, C = 0) = \frac{N_{10}}{N}$
- $P(U = 1, C = 1) = \frac{N_{11}}{N}$

Ainsi l'équation 6.1 se calcule de manière suivante :

$$\frac{N_{11}}{N} \log_2 \frac{NN_{11}}{N_{1.}N_{.1}} + \frac{N_{01}}{N} \log_2 \frac{NN_{01}}{N_{0.}N_{.1}} + \frac{N_{10}}{N} \log_2 \frac{NN_{10}}{N_{1.}N_{0.}} + \frac{N_{00}}{N} \log_2 \frac{NN_{00}}{N_{0.}N_{.0}} \tag{6.2}$$

Avec $N_{e_te_c}$ tel que :

- N_{10} : le nombre de phrases qui contiennent le terme t ($e_t = 1$) et qui n'appartiennent pas à la catégorie C ($e_c = 0$).
- N_{11} : le nombre de phrases qui contiennent le terme t ($e_t = 1$) et qui appartiennent à la catégorie C ($e_c = 1$).
- N_{01} : le nombre de phrases qui ne contiennent pas le terme t ($e_t = 0$) et qui appartiennent à la catégorie C ($e_c = 1$).
- N_{00} : le nombre de phrases qui ne contiennent pas le terme t ($e_t = 0$) et qui n'appartiennent pas à la catégorie C ($e_c = 0$).
- $N_{1.} = N_{10} + N_{11}$: le nombre de phrases qui contiennent le terme t ($e_t = 1$) indépendamment de l'appartenance à une classe ($e_c \in \{0, 1\}$).
- $N_{0.} = N_{10} + N_{11}$: le nombre de phrases qui ne contiennent pas le terme t ($e_t = 0$) indépendamment de l'appartenance à une classe ($e_c \in \{0, 1\}$).
- $N_{.1} = N_{10} + N_{11}$: le nombre de phrases qui appartiennent à la classe c ($e_c = 1$) indépendamment du terme t.
- $N_{.0} = N_{10} + N_{11}$: le nombre de phrases qui n'appartiennent pas à la classe c ($e_c = 0$) indépendamment du terme t.
- $N = N_{00} + N_{01} + N_{10} + N_{11}$: le nombre total de phrases.

Par exemple, avec un corpus d'apprentissage donné, si l'on considère la catégorie *mechanics* et le terme *voltage*, en calculant les quatre combinaisons possibles résumées dans la table 6.2, alors l'information mutuelle $I(U; C)$ du terme *voltage* vis-à-vis de la classe *mechanics* est environ égale à 0.0001105. Autrement dit, le terme *voltage* n'est pas une caractéristique discriminante pour la catégorie *mechanics*.

	$e_c = e_{mechanics} = 1$	$e_c = e_{mechanics} = 0$
$e_t = e_{voltage} = 1$	$N_{11} = 49$	$N_{10} = 27652$
$e_t = e_{voltage} = 0$	$N_{01} = 141$	$N_{50} = 774106$

TABLE 6.2 – Espace de caractéristiques modélisé comme un modèle vectoriel.

En statistique, le test X^2 est utilisé pour tester l'indépendance de deux évènements, c'est à dire $P(AB) = P(A)P(B)$. Dans l'exercice de la sélection de caractéristiques, les deux évènements sont l'occurrence d'un terme t et l'occurrence d'une catégorie c. X^2 mesure alors la déviation entre le nombre d'occurrences espérées E et le nombre d'occurences observées N. Le test du x^2 se calcule de la manière suivante :

$$X^2(\mathbb{P}, t, c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_te_c} - E_{e_te_c})^2}{E_{e_te_c}} \tag{6.3}$$

Avec :

– N la fréquence observée dans l'ensemble des phrases $\mathbb{P}$
– E la fréquence espérée, avec E_{11} la fréquence espérée que le terme t et la classe c se produisent en même temps dans une exigence supposant leur indépendance.
– $E_{11} = N * P(t) * P(c) = N * \frac{N_{11}+N_{10}}{N} * \frac{N_{11}+N_{01}}{N}$.

Notre problème de classification est de grande dimension – le nombre de caractéristiques est supérieur à 1000 –, nous utilisons donc la méthode de filtrage. Manning et al. [2008] soulignent le fait que les mesures d'utilité basées sur la fréquence d'apparition sont moins performantes que le test du X^2 et de l'information mutuelle. Par ailleurs, Joachims [1998] rappellent que l'information mutuelle est la mesure d'utilité qui fournit les meilleures performances. Nous l'avons donc utilisée pour sélectionner le sous-ensemble de caractéristiques discriminantes.

Pour définir le nombre de caractéristiques discriminantes à conserver, nous entraînons deux algorithmes d'apprentissage en faisant varier le nombre de caractéristiques, puis observons les performances de la fonction de classification en mesurant la F-mesure qui est égale à $\frac{2PR}{P+R}$ (Sec. 4.4 pour un rappel). Le temps d'entraînement étant d'environ deux heures, nous nous sommes limités à un modèle Bayésien (NB) et une machine à vecteurs supports (SVM). Un modèle Bayésien car c'est un algorithme qui est assez général pour être appliqué à presque tous les problèmes de classification, tandis que l'algorithme SVM est considéré comme étant le meilleur choix pour la classification de texte, ou, plus généralement, quand le nombre de caractéristiques est important – plus de 1000 caractéristiques en général – [Joachims, 1998]. Par ailleurs, ces deux algorithmes d'apprentissage servent les deux perspectives du compromis biais-variance. En effet, l'algorithme de NB n'est pas flexible, il a donc un biais élevé et une variance faible. Inversement, l'algorithme SVM a beaucoup plus de paramètres, il est donc flexible avec un biais faible et une variance élevée. Autrement dit, un modèle Bayésien va obtenir des résultats moyens mais plus ou moins identiques avec des données variées. Inversement, une machine à vecteurs supports va obtenir de très bons résultats dans un cas et des résultats nettement moins bons dans un autre cas.

En faisant varier le nombre de caractéristiques de 100 à 10 000 on observe que la F-mesure atteint un maximum égal à 0.719 lorsque les 1000 premières caractéristiques les plus discriminantes sont utilisées pour entraîner l'algorithme NB. Avec l'algorithme SVM, les performances sont sensiblement meilleures puisque nous obtenons une F-mesure égale à 0.737 lorsque le modèle est entraîné avec les 2000 caractéristiques les plus discriminantes. Les détails de l'expérience sont présentés dans la section 6.4.

Bien que cette fonction de classification puisse prévoir à quel métier une exigence appartient avec approximativement 3 erreurs sur 10, on peut s'interroger quant à la stratégie initiale. Au départ, l'objectif était d'assurer le passage à l'échelle en permettant à chaque expert d'estimer uniquement les exigences qui relèvent de son domaine d'expertise. Les mécaniciens estiment les exigences de la mécanique et les électroniciens se concentrent sur les exigences de l'électronique. Cependant, après une réflexion commune avec notre partenaire industriel, nous avons écarté cette solution parce que la plate-forme collaborative doit être un moyen de fédérer toutes les fonctions de l'entreprise. Pour qu'une décision soit objective, elle doit résulter d'un consensus des connaissances tacites des fonctions de l'entreprise. Or, si les experts sont les seuls agents à s'exprimer sur leurs propres intérêts, alors cela conduit à des décisions absolutistes d'un décideur souverain. Autrement dit, donner les moyens au chargé du marketing de s'exprimer sur des exigences de mécanique, c'est potentiellement un moyen de découvrir des opportunités commerciales ou de prévenir des risques liés à la concurrence. Toutes les fonctions de l'entreprise doivent estimer les critères d'aide à la décision.

Segmenter les exigences (FT31) - alternative 2 : partitionnement sémantique

Certains travaux présentés dans l'état de l'art, en particulier ceux de Duan and Cleland-Huang [2007a], ont opté pour une autre alternative : le partitionnement des exigences. Ainsi, plutôt que de prioriser des exigences, les auteurs travaillent à un niveau d'abstraction supérieur en ordonnant des partitions d'exigences.

Les techniques de partitionnement appartiennent à la catégorie des algorithmes d'apprentissage non supervisé (*clustering*, en anglais). Un algorithme d'apprentissage non supervisé sert à explorer un gros volume de données en cherchant des partitions de telle sorte que les objets dans une partition sont à

la fois très similaires entre eux et très différents de ceux contenus dans les autres partitions. Duan and Cleland-Huang [2007a] se limitent à l'aspect lexical en regroupant les exigences selon la similarité entre des mots clés. Mokammel et al. [2015] donnent quelques prémisses d'une décomposition en valeurs singulières (*Singular Value Decomposition*, en anglais) (SVD) couplée à l'algorithme des k-moyens (*k-means*, en anglais). Cette approche a deux inconvénients. D'une part, elle se limite à l'aspect linguistique et, d'autre part, elle nécessite de choisir *a priori* le nombre de partitions k. Sannier [2013] a déjà discuté l'inconvénient de devoir choisir le nombre de partitions. Cela requiert notamment de tester différentes valeurs pour chaque nouveau corpus de données. En effet, puisque le contenu n'est pas connu à l'avance, il faut tester $k = 2$, $k = 3$, $k = ...$ avant d'obtenir un résultat satisfaisant. Ainsi, si nous adoptions cette approche, le manageur devrait essayer plusieurs paramétrages de segmentation avant d'obtenir une « bonne » définition des partitions d'exigences.

Pour contourner le problème lié au choix du nombre de partitions nous avons testé l'algorithme de partitionnement sémantique Linlog [Osiński et al., 2004]. Linlog partitionne des énoncés textuels (phrases, pages Web, livres, etc.) en analysant le niveau lexical selon un processus en quatre étapes.

La première étape est un pré-traitement du texte. Le texte est d'abord segmenté en mots. Chaque mot est normalisé pour obtenir son radical, et tous les caractères non alphabétiques et les mots vides de sens (pronoms, articles, etc.) sont supprimés.

La seconde étape consiste à extraire les séquence de termes les plus fréquentes. Les séquences les plus fréquentes servent à baptiser les partitions. Par exemple, une partition des exigences spécifiant un avion pourrait être baptisée « aile résiste » pour représenter toutes les exigences relatives à la résistance des ailes. Pour qu'une séquence devienne candidate, elle doit remplir quatre conditions. Premièrement, il faut que sa fréquence d'apparition soit supérieure à un certain seuil. Deuxièmement, il faut que les termes de la séquence appartiennent à la même phrase, c'est-à-dire qu'ils ne soient pas à cheval sur deux phrases. Troisièmement, la séquence de termes doit être complète. Dans l'exemple 6.4, la séquence « bc » n'est pas complète à droite car le même caractère « d » suit toutes les séquences. Elle n'est pas complète à gauche non plus, car le caractère « a » précède toutes les séquences. Par contre, la séquence « bcd » est complète à droite parce que toutes les séquences sont suivies de caractères différents « e », « f » et « g ». Pour que la séquence « bcd » soit aussi complète à gauche, il faut qu'elle soit étendue et que le premier caractère « x » soit un « a » (Eqn. 6.5). On obtient alors la phrase complète « abcd ». Enfin, quatrièmement, pour qu'une séquence de termes soit une candidate, elle ne doit pas commencer ou se terminer par un mot vide de sens (article, pronom, etc.).

$$x\ a\ \mathbf{b}\ \mathbf{c}\ d\ e\ a\ \mathbf{b}\ \mathbf{c}\ d\ f\ a\ \mathbf{b}\ \mathbf{c}\ d\ g \tag{6.4}$$

$$x\ a\ \mathbf{b}\ \mathbf{c}\ \mathbf{d}\ e\ a\ \mathbf{b}\ \mathbf{c}\ \mathbf{d}\ f\ a\ \mathbf{b}\ \mathbf{c}\ \mathbf{d}\ g \tag{6.5}$$

La troisième étape sert à baptiser les partitions recherchées. Pour cela, LinLog exploite un modèle vectoriel représenté par une matrice termes-documents – termes-exigences dans notre cas – dont les entrées A_{ij} correspondent à la fréquence pondérée TF-IDF (*term frequency, inverse document frequency*, en anglais). TF-IDF considère qu'un terme est un meilleur indicateur pour un texte donné s'il est fréquent dans ce texte relativement à son nombre d'occurrences dans tous les textes. La décomposition en valeurs singulières de la matrice terme-document permet d'obtenir une base orthogonale qui est supposée représenter les concepts abstraits contenus dans chaque document. La décomposition en valeurs singulières (SVD) [Strang, 2006] est une technique d'algèbre qui permet de factoriser une matrice A en une matrice $\hat{A}$. $\hat{A}$ est alors dans un espace de plus petite dimension tel que la norme $\Delta = \|A - \hat{A}\|_2$ est minimisée. SVD est analogue à un problème de régression dans un espace de dimension 2 dans lequel on cherche à approximer un ensemble de points avec une droite de dimension 1. Avec SVD on projette le vecteur d'une exigence de dimension m correspondant à la taille totale du vocabulaire, dans un espace de dimension k avec $k <<< m$.

Enfin, la quatrième étape est d'assigner chaque document – exigence dans notre cas – à une ou plusieurs partitions. Pour cela, un énième modèle vectoriel est définit afin de calculer le cosinus entre

chaque document et chaque communauté. Le document est alors assigné à la communauté avec laquelle il partage le plus d'affinité sémantique.

Segmenter les exigences (FT31) - alternative 3 : détection de communautés

Les solutions de partitionnement de textes discutées ci-dessus segmentent les exigences en s'appuyant sur des fonctions qui mesurent la similarité linguistique, en particulier lexicale, entre deux énoncés textuels. Or, les affinités entre les exigences ne sont pas que sémantiques. Il existe de multiples interdépendances (sémantiques, conceptuelles, références transversales, dérivations, implémentations, etc.) entre les exigences qui ne sont pas considérées dans les solutions existantes. Ainsi, ici, sur la base de résultats récents en théorie des graphes, nous étudions une approche qui non seulement exploite l'aspect sémantique, mais aussi toutes les relations imaginables entre les exigences.

La segmentation d'un graphe consiste à analyser sa topologie afin d'identifier des sous-graphes particuliers. Par exemple, dans la figure 6.9, 6 sous-groupes particuliers sont identifiés dans le graphe (a) , lesquels donnent naissance à un nouveau graphe (b) de niveau conceptuel plus élevé.

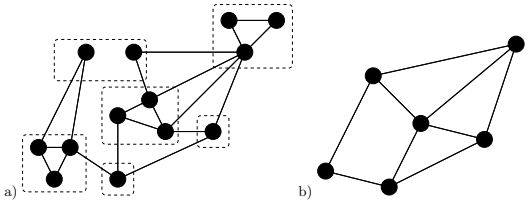


FIGURE 6.9 – Graphe segmenté [Kaufmann and Wagner, 2001].

La première méthode de segmentation consiste à trouver des sous-graphes particuliers locaux. On peut par exemple chercher les cliques, c'est-à-dire les sous-graphes complets [1]. Dans la figure 6.10, le sous graphe engendré par les sommets en rouge est une clique d'ordre 4.

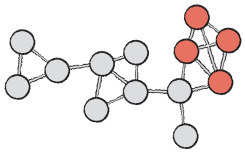


FIGURE 6.10 – Exemple de clique d'ordre 4 [Brath and Jonker, 2015].

Si les cliques sont trop contraignantes, alors on peut chercher des sous-graphes particuliers locaux moins restrictifs tels que les *k-plexes* ou les *k-cores* [Newman, 2010]. Une structure locale qui est souvent d'intérêt est la composante. Une composante d'un graphe G est un sous-graphe engendré connexe [2] maximal [3] de G. Il est parfois utile de trouver les k-composantes (Fig. 6.11) .

On peut aussi rechercher des composantes fortement connexes. Un graphe orienté G est fortement connexe si quels que soient deux sommets distincts x et y il existe un chemin allant de x à y. Ainsi, une composante fortement connexe de G est un sous-graphe engendré de G fortement connexe et maximal. Dans la figure 6.11, les sous-graphes entourés pas des traits discontinus sont des composantes fortement connexes.

Nous n'avons retenu aucune de ces notions car elles permettent d'identifier des sous-graphes particuliers de très petites tailles. L'application d'une de ces notions à un graphe constitué de plusieurs centaines ou milliers d'exigences résulterait en une multitudes de communautés et cela ne faciliterait pas la tache des experts.

1. Un graphe complet est un graphe simple tel que deux sommets distincts quelconques sont reliés par une arête.
2. Un graphe est connexe si quels que soient deux sommets ils sont reliés par une chaîne.
3. Un graphe G est maximal s'il n'existe pas de sous graphe engendré fortement connexe qui contient strictement ce sous-graphe.

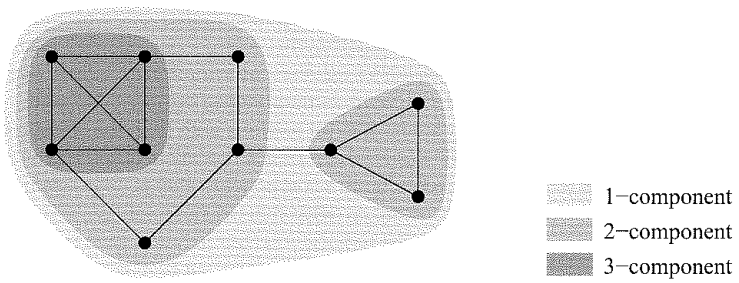


FIGURE 6.11 – K-composantes d'un graphe [Newman, 2010].

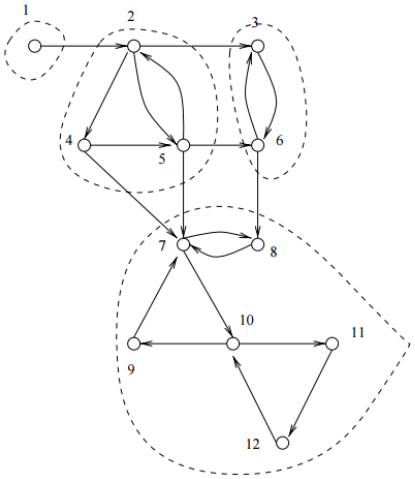


FIGURE 6.12 – Composantes fortement connexes d'un graphe.

La seconde méthode de segmentation consiste à partitionner le graphe (*graph partitioning*, en anglais), c'est à dire à le diviser en groupes distincts avec un nombre minimal d'arêtes entre chaque groupe. Le nombre d'arêtes entre chaque groupe est appelé la taille de coupe. Si aucune contrainte n'est imposée, alors il y a des solutions triviales. En effet, si l'on cherche deux communautés, alors on peut mettre tous les sommets dans une première communauté et un seul sommet dans une seconde communauté, la taille de la coupe est alors minimisée. De même si l'on cherche trois, quatre, ..., n communautés. Ainsi, la première caractéristique du partitionnement c'est que le nombre de communautés doit être fixé à l'avance. La seconde caractéristique c'est que la taille de chaque communauté doit aussi être plus ou moins spécifiée à l'avance. Le partitionnement d'un graphe est alors une opération difficile. La solution générale est de commencer par faire une bi-section, c'est-à-dire partitionner le graphe en deux, puis répéter l'opération sur chacune des parties. Ainsi, pour diviser un graphe en groupes distincts avec une coupe minimale on peut faire une recherche exhaustive en essayant toutes les coupes possibles, mais cela est trop coûteux en temps de calcul. Diverses heuristiques ont donc été proposées, notamment celles de l'algorithme de Kernighan-Lin et du partitionnement spectral [Newman, 2010]. Le partitionnement de graphe est pas exemple utile quand on souhaite distribuer un calcul sur un réseau de machines dont on connait le nombre de nœuds. Cependant, dans notre problématique, nous n'avons aucune idée *a priori* du nombre et de la taille des communautés. Par ailleurs, le graphe des exigences varie constamment selon les documents prescriptifs analysés, c'est pour cela que nous avons aussi exclu cette alternative de solution.

Enfin, la troisième méthode de segmentation d'un graphe est la détection de communautés (*community detection*, en anglais). Contrairement au partitionnement de graphe, la découpe en communautés ne nécessite aucune information initiale concernant la taille et le nombre de communautés. En effet, l'objectif est de trouver des divisions naturelles. Des divisions sont naturelles si il y a beaucoup d'arêtes à

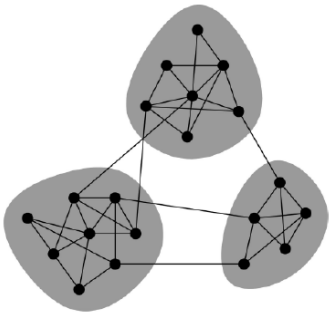


FIGURE 6.13 – Exemple de communautés dans un graphe [Newman, 2006].

l'intérieur de chaque communauté et peu d'arêtes entre les communautés (Fig. 6.13). Une bonne division d'un graphe n'est pas celle pour laquelle le nombre d'arêtes entre les communautés est minimum, mais celle pour laquelle il y a moins d'arêtes qu'espéré. L'idée que la structure du graphe est un arrangement d'arêtes qui est statistiquement surprenant peut être mesuré par ce que l'on appelle la modularité. La modularité (Eq. 6.6) est, à une constante multiplicative près, le nombre d'arêtes dans chaque groupes moins le nombre d'arêtes espérées dans un graphe équivalent dans lequel les arêtes sont placées aléatoirement.

$$Q = \frac{1}{4m} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2m})(s_i s_j + 1) \tag{6.6}$$

Avec :

- k_i (resp. k_j) le degré [4] du sommet i (resp. j).
- $m = \frac{1}{2} \sum_i k_i$ le nombre total d'arêtes dans le graphe.
- A_{ij} le nombre d'arêtes entre le sommet i et le sommet j.
- $\frac{k_i k_j}{2m}$ le nombre espéré d'arêtes entre le sommet i et le sommet j.
- $\frac{1}{2}(s_i s_j + 1) = 1$ si les sommets i et j appartiennent au même groupe, sinon 0.

L'objectif de l'exercice d'optimisation est alors de trouver les communautés en cherchant les divisions qui ont une modularité positive et, si possible, très élevée. Divers auteurs ont essayé de résoudre le problème de maximisation avec différents algorithmes ; ici, nous utilisons l'une des dernières heuristiques de la théorie des graphes : l'algorithme spectral de Newman [2006]. Une implémentation libre de cet algorithme est disponible dans l'API Jmod, une librairie Java développée à l'École Polytechnique Fédérale de Lausanne pour détecter les communautés dans un graphe d'interactions biologiques.

Nous avons préféré la détection de communautés comme approche de segmentation de graphe car la division est naturelle, elle ne nécessite aucun paramétrage. Avant de découper un ensemble d'exigences en communautés, il faut construire un graphe d'exigences. Initialement, notre graphe d'exigences $G(V, E)$ est engendré par un ensemble d'exigences V reliées par un ensemble d'arêtes E qui correspondent aux relations transversales identifiées dans la section 4.3. La taille de ce sous-graphe engendré n'est clairement pas suffisant pour découper les exigences en communautés. D'autres relations sont nécessaires pour induire un graphe contenant, autant que faire se peut, toutes les exigences.

Nous utilisons alors deux ressources de connaissances pour enrichir le graphe. La première ressource est le thésaurus WordNet. Nous avons déjà utilisé ce dernier pour identifier les redondances et les contradictions potentielles. Comme l'illustre la figure 6.14, WordNet est un graphe dans lequel chaque lemme est lié à d'autres lemmes via des relations sémantiques. Il existe plusieurs types de relations :

- « Hypernymy » : Airplane *is a kind of* ; e.g. *Airplane* ***is a kind of*** *Aircraft*
- « Hyponymy » : ... *is a kind of* Airplane ; e.g. *Bomber* ***is a kind of*** *Airplane*
- « Holonymy » : Wing *is a part of* ... ; e.g. *Wing* ***is a part of*** *Airplane*
- « Meronymy » : ... *is a part of* Airplane ; e.g. *Wing* ***is a part of*** *Airplane*
- « Synonymy » : Airplane *is a synonym of* ... ; e.g. *Airplane* ***is a synonym of*** *Plane*
- « Derivationally related terms » : Wing *is related to* ... ; e.g. *Wing* ***is related to*** *fly*

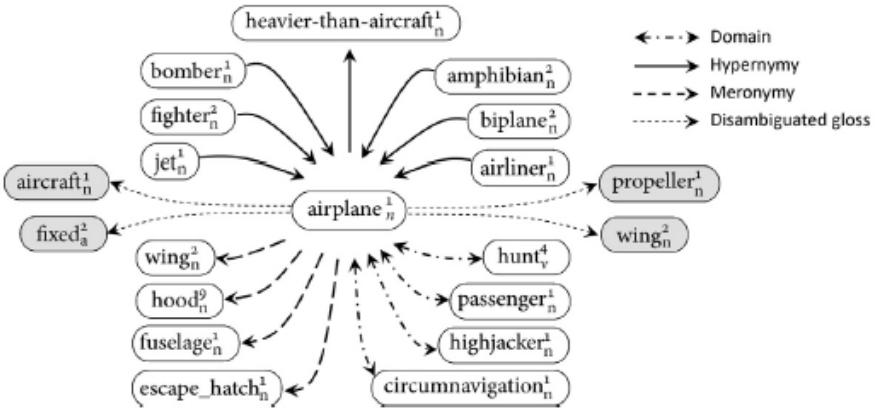
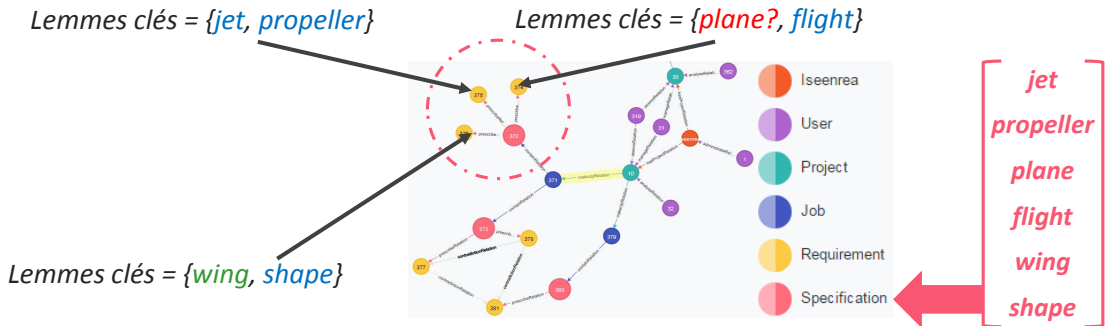


FIGURE 6.14 – Le champ lexical du mot *airplane* dans le thésaurus WordNet.

Pour chaque paire d'exigences (R_i, R_j) nous cherchons alors les relations sémantiques entre les lemmes clés. Néanmoins, comme on l'a vu précédemment (Fig. 5.8), dans le thésaurus WordNet, un lemme peut avoir plusieurs signifiés. Le champ lexical (hyponymes, méronymes, etc.) d'un lemme est différent pour chaque signifié, il est donc nécessaire de trouver le bon signifié de chaque lemme (*word sense disambiguation*, en anglais) avant d'exploiter son champ lexical. De nombreux travaux, parfois très sophistiqués, ont été réalisés dans ce sens, en particulier dans la théorie du traitement du langage naturel dont les principales techniques sont introduites par Manning and Schütze [1999].



The noun plane has 5 senses (first 3 from tagged texts)

1. (21) airplane, aeroplane, **plane** -- (an aircraft that has a fixed wing and is powered by propellers or jets; "the flight was delayed due to trouble with the airplane")
2. (16) **plane**, sheet -- ((mathematics) an unbounded two-dimensional shape; "we will refer to the plane of the graph as the X-Y plane"; "any line joining two points on a plane lies wholly on that plane")
3. (3) **plane** -- (a level of existence or development; "he lived on a worldly plane")
4. **plane**, planer, planing machine -- (a power tool for smoothing or shaping wood)
5. **plane**, carpenter's plane, woodworking plane -- (a carpenter's hand tool with an adjustable blade for smoothing or shaping wood; "the cabinetmaker used a plane for the finish work")

The verb plane has 3 senses (first 1 from tagged texts)

1. (2) **plane**, shave -- (cut or remove with or as if with a plane; "The machine shaved off fine layers from the piece of wood")
2. **plane**, skim -- (travel on the surface of water)
3. **plane** -- (make even or smooth, with or as with a carpenter's plane; "plane the top of the door")

FIGURE 6.15 – Désambiguïsation sémantique d'un lemme clé.

Dans notre cas, pour désambiguïser un lemme clé d'une exigence R_i extraite d'un document prescriptif D_j, nous supposons que le signifié du terme est celui qui est sémantiquement le plus proche du document prescriptif dont l'exigence est issue. Autrement dit, si la majorité des exigences d'un document prescriptif contiennent des termes appartenant au champ lexical de la banque, alors le terme *bank*

4. Le degré d'un sommet i est le nombre d'arêtes adjacentes à i.

signifie une institution financière. Au contraire, si la majorité des exigences contiennent des termes appartenant au champ lexical des travaux publics, ou de l'eau, alors le terme *bank* signifie la rive d'un cours d'eau. Ainsi, dans un premier temps, nous commençons par construire le sac de lemmes clés distincts du document prescriptif D_{ij}. Dans l'exemple de la figure 6.15, pour désambiguïser le lemme *plane*, nous accédons au document prescriptif associé $D1$, puis nous récupérons toutes les exigences qu'il contient afin de construire le sac de lemmes clés $\vec{L}_{D1}$:

$$\vec{L}_{D1} = \{jet, propeller, plane, flight, wing, shape\} \tag{6.7}$$

Ensuite nous faisons une requête dans le thésaurus WordNet en fonction de la catégorie syntaxique du lemme étudié. Dans cet exemple, le terme *plane* est un nom, donc nous collectons les signifiés associés au nom *plane*. Les signifiés sont ensuite traités pour récupérer la liste des synonymes et des lemmes clés de la définition associée à chaque signifié. Un sac de mots est alors construit pour chaque signifié. Par exemple, Pour les signifiés 1 et 2, les vecteurs de lemmes clés L_{S1} et L_{S2} sont :

$$\vec{L}_{S1} = \{airplane, aeroplane, plane, aircraft, fix, wing, power, propeller, jet, flight, delay, due, trouble\} \tag{6.8}$$

$$\vec{L}_{S2} = \{plane, sheet, unbound, two-dimensional, shape, will, refer, graph, line, joining, points, wholly\} \tag{6.9}$$

En calculant l'intersection entre le sac de lemmes clés associé à chaque signifié et le sac de lemmes clés du document prescriptif, nous identifions le signifié approprié. Par exemple, l'intersection $\vec{L}_{S1} \cap \vec{L}_{D1} = 5$, tandis que $\vec{L}_{S2} \cap \vec{L}_{D1} = 2$. Le signifié du terme *plane* dans ce contexte donné est donc le signifié (1). À partir de ce signifié nous faisons une requête pour récupérer le champ lexical associé.

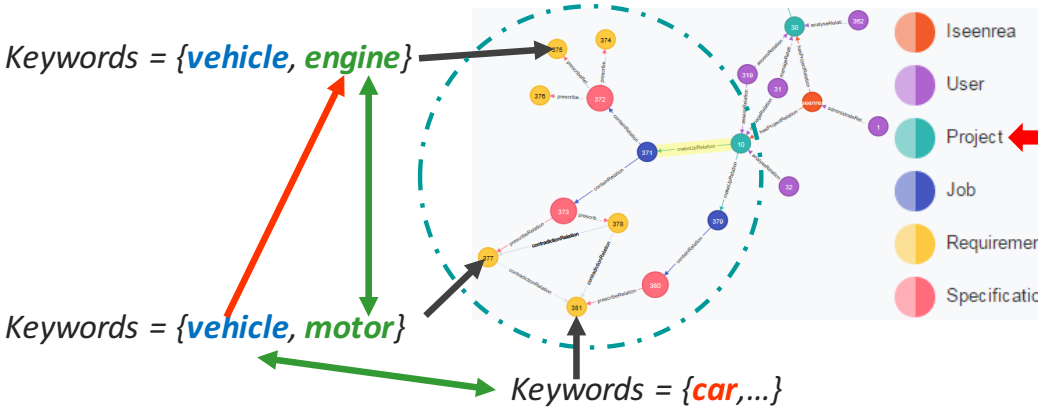


FIGURE 6.16 – Relations sémantiques (en vert) et conceptuelles (en rouge) entre les lemmes clés.

Comme le montre la figure 6.16, pour construire les relations sémantiques il faut parcourir les $\frac{n*(n-1)}{2}$ paires d'exigences dans le projet. Pour une paire d'exigence (R_i, R_j), nous parcourons chaque lemme clé de R_i (resp. R_j), nous récupérons le champ lexical associé et vérifions si l'un des lemmes connexes est contenu dans la liste de lemmes clés de l'exigence R_j (resp. R_i). Par exemple, le lemme clé *engine* a comme synonyme le lemme *motor*, donc une relation de synonymie est créée entre les deux exigences. De même pour les lemmes *vehicle* et *car*, tandis que *engine* et *vehicle* sont liés par une relation d'holonymie (*engine* ***is a part of*** *vehicle*).

De la même manière, nous enrichissons le graphe engendré par les exigences d'un projet avec les relations issues de l'ontologie ConceptNet 5. ConcepNet 5 est un graphe qui modélise des connaissances

nécessaires à la compréhension de textes (Fig. 6.17). Ainsi, alors que toutes les solutions de segmentation des exigences vues dans l'état de l'art s'appuient sur des métriques de similarité linguistique ; avec cette proposition, nous pouvons bénéficier des relations qui relèvent des connaissances générales. Pour accéder aux connaissances de ConceptNet 5, nous faisons des requêtes sur le Web et récupérons le graphe associé sous le format JSON.

FIGURE 6.17 – Extrait des relations conceptuelles pour le concept *airplane* issues de ConceptNet 5.

Data: Given a set of requirements $\mathbb{R}$
foreach *requirement* $R_i \in \mathbb{R} = \{R_1, R_2, ..., R_m\}$ **do**
  get set of key lemmas $\mathcal{L}_i = \{L_{i1}, L_{i2}, ..., L_{ij}\}$ of requirement R_i;
  foreach *requirement* $R_k \in \mathbb{R} = \{R_1, R_2, ..., R_m\}$ *such that* $i < k$ **do**
    get set of key lemmas $\mathcal{L}_k = \{L_{k1}, L_{k2}, ..., L_{kl}\}$ of requirement R_k;
    foreach *key lemma* $L_{km} \in \mathcal{L} = \{L_{k1}, L_{k2}, ..., L_{kn}\}$ *of requirement* R_k **do**
      get set of synonyms $\mathcal{S} = \{S_{km1}, S_{km2}, ..., S_{kmp}\}$ of key lemma L_{km};
      foreach *synonym* $S_{kmq} \in \mathcal{S}_{km} = \{S_{km1}, ..., S_{kmp}\}$ *of key lemma* L_{km} **do**
        if $S_{kmq} \in \mathcal{L}_i = \{L_{i1}, L_{i2}, ..., L_{ij}\}$ **then**
          create semantic relation (R_i,R_j,$synonymy$);
        end
      end
      get set of meronyms $\mathcal{M} = \{M_{km1}, M_{km2}, ..., M_{kmp}\}$ of key lemma L_{km};
      foreach *meronym* $M_{kmq} \in \mathcal{M}_{km} = \{M_{km1}, ..., M_{kmp}\}$ *of key lemma* L_{km} **do**
        if $M_{kmq} \in \mathcal{L}_i = \{L_{i1}, L_{i2}, ..., L_{ij}\}$ **then**
          create semantic relation (R_i,R_j,$meronymy$);
        end
      end
      // Repeat for holonyms, hypernyms, hyponyms, derivationally related terms. ...
    end
    get set of knowledge relations $\mathcal{KR} = \{KR_{km1}, ..., KR_{kmp}\}$ of key lemma L_{km};
    foreach *knowledge relation* $KR_{kmq} \in \mathcal{KR}_{km} = \{KR_{km1}, ..., KR_{kmp}\}$ **do**
      get term in source node t_s;
      get term in target node t_t;
      get conceptual relation type KR_type;
      if $T_S \in \mathcal{L}_i = \{L_{i1}, L_{i2}, ..., L_{ij}\}$ **then**
        create conceptual relationship (R_i,R_j,t_s,t_t,$conceptual$,KR_{type});
      else
        create conceptual relationship (R_i,R_j,t_t,t_s,$conceptual$,KR_{type});
      end
    end
  end
end

Algorithm 1: Algorithme pour enrichir le graphe de relations sémantiques et conceptuelles à partir de WordNet et ConceptNet.

Le graphe sur lequel le manager applique l'algorithme de détection de communautés est alors le sous-graphe engendré par les exigences qui sont connectées par des arêtes qui correspondent : aux relations transversales, aux relations sémantiques issues du thésaurus WordNet, et aux relations conceptuelles issues de l'ontologie ConcetNet 5. Ce sous-graphe engendré est, sauf exception, un multi-graphe car il possède des arêtes multiples [5]. On ne peut pas directement appliquer l'algorithme spectral de Newman sur un multi-graphe, il faut au préalable le transformer en un graphe simple pondéré. Un graphe simple pondéré est un graphe dans lequel les arêtes multiples sont remplacées par des arêtes simples dont le poids associé à chacune d'entre elles correspond au nombre d'arêtes. Par exemple, l'arête multiple entre les sommets A et C du multi-graphe de la figure 6.18 correspond à une arête simple dont le poids est 3. Cette transformation n'est pas directement appliquée au niveau du graphe des propriétés dans la base de données. Pour la réaliser, nous faisons une requête pour récupérer le sous-graphe engendré par les exigences avant de générer sa matrice d'adjacence dans un fichier écrit selon le Graph Modeling Language (GML). C'est ce fichier GML qui sert d'entrée à l'algorithme de détection de communautés.

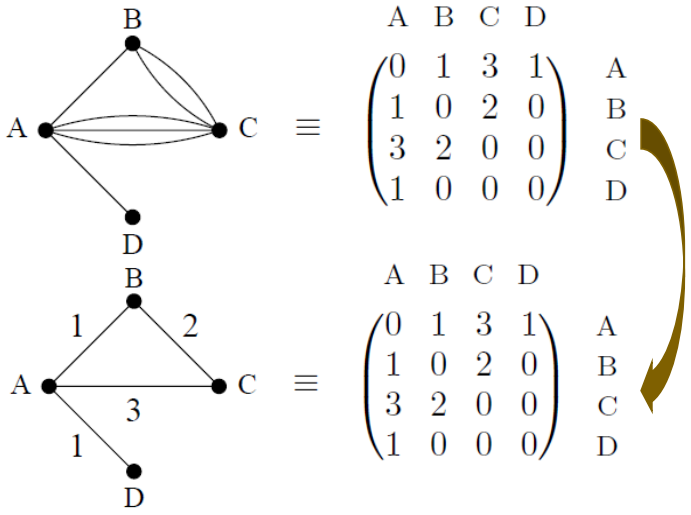


FIGURE 6.18 – Exemple de transformation d'un multi-graphe en un graphe simple pondéré.

Les communautés détectées qui résultent de la segmentation du graphe sont alors présentées à chaque analyste sous une forme graphique (Fig. 6.19). Le diamètre d'une communauté est fonction du nombre d'exigences qu'elle contient, tandis que la couleur indique si la communauté a été estimée – communautés vertes – ou non – communautés rouges – par l'expert en session. Plutôt que de chercher un moyen efficace de donner un titre à chaque communauté, quand l'expert clique sur l'une d'entre elles, un nuage des noms et verbes les plus fréquents dans les exigences sous-jacentes est généré.

Définir (FT32) et estimer (FT33) les critères d'aide à la décision

Dans l'état de l'art, les propositions ont pour objectif de prioriser des communautés d'exigences. Dans les travaux présents, nous ne cherchons pas à établir un ordre de priorités entre les exigences, nous souhaitons demander aux différentes fonctions de l'entreprise d'exploiter leurs connaissances pour aider un manageur de projet à prendre une décision informée. Nous supposons que cela est faisable au moyen d'un système de vote collaboratif et multi-disciplinaire qui permet à chacune des fonctions de l'entreprise d'estimer des critères d'aide à la décision associés aux communautés d'exigences.

Pour cela, nous proposons de définir deux catégories d'experts qui estiment les critères d'aide à la décision : les « fonctions métiers » et les « fonctions transverses ». Les fonctions métiers sont les disciplines techniques, les génies : mécanique, électrique, électronique, civil, thermique, informatique, etc. Les fonctions transverses, elles, interviennent sur tout ou partie des fonctions métiers et sont, par

5. Une arête entre deux sommes E_1 et E_2 est dite multiple s'il existe plusieurs arêtes entre E_1 et E_2.

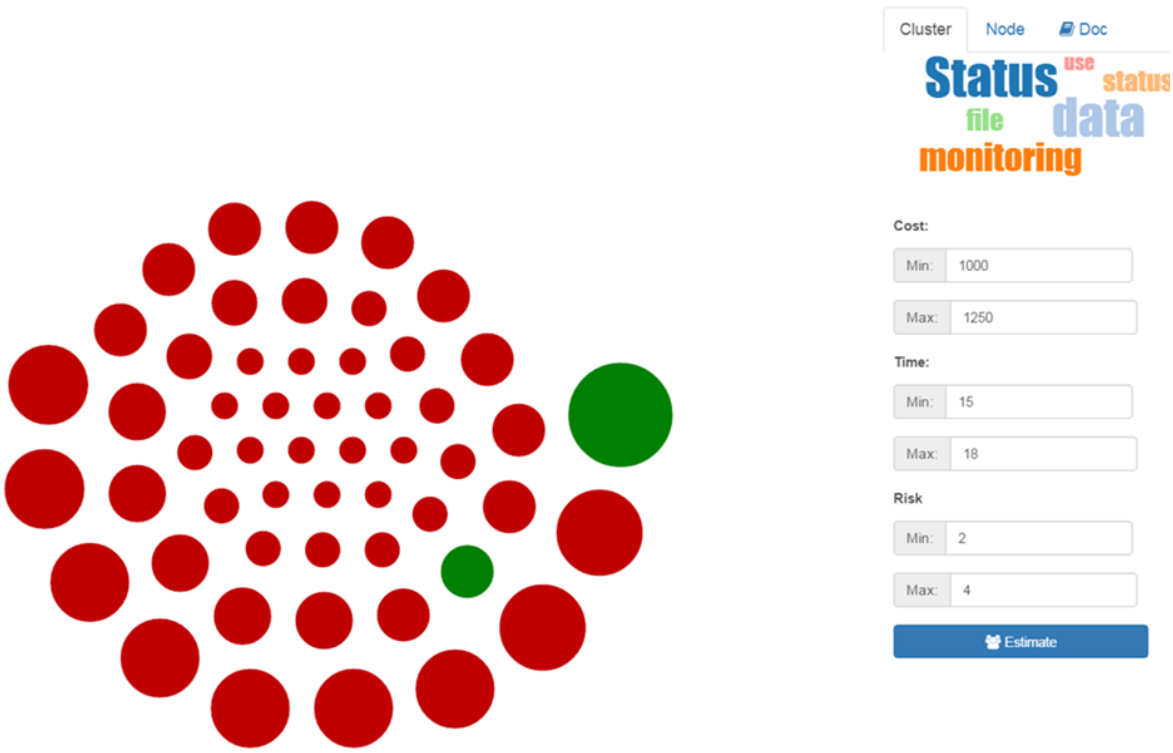


FIGURE 6.19 – Vue d'un expert qui doit estimer les communautés d'exigences.

exemple, le marketing ou la qualité. Ainsi, quand le manageur d'un projet créé les experts dans l'environnement numérique, il attribue à chacun d'entre eux une fonction métier ou transverse. Cette propriété peut non seulement permettre au manageur de pondérer les estimations, mais aussi d'observer les dispersions selon les deux perspectives.

Une fois que le manageur a, pour un projet donné, créé les experts et automatiquement segmenté les exigences, il définit les critères d'aide à la décision qui sont associés à chacune des communautés. Dans l'exemple de la figure 6.19, trois critères d'aide à la décision – coût, temps et risque – ont été défini pour le projet. Nous avons volontairement choisi de laisser le choix au manageur de définir les critères car, comme cela a été discuté précédemment, pas moins de 280 critères ont été recensés dans la littérature. Ces critères changent selon le contexte (l'entreprise, le projet, les parties prenantes, etc.), ils doivent donc être configurables par la manageur de chaque projet.

Nous avons aussi choisi que les critères d'aide à la décision doivent être quantitatifs et évalués selon un intervalle [min ; max], et ce, pour deux raisons. Comme nous l'avons vu dans l'état de l'art, les estimations qualitatives n'aident pas à décider, elles font naître le doute, le scepticisme, l'ambiguïté, sauf si, par exemple, des valeurs quantitatives servent à quantifier une échelle ordinale. Par exemple, si un expert estime que le coût d'une communauté d'exigences est « élevé » ; comment est-ce que cela peut aider le manageur du projet à décider ? Par contre, si un coût est « élevé » quand il appartient à l'intervalle [50 000 Euros ; 75 000 Euros], alors le manageur a une idée concrète des ressources financières qu'il va engager s'il décide de développer cette communauté d'exigences. C'est d'ailleurs ce même principe de fourchette que nous utilisons plus ou moins tous pour estimer le coût et le temps lors d'une réponse à un appel d'offre. Cela ne veut pas dire que des techniques analytiques ne peuvent pas être appliquées en amont de l'estimation - bien au contraire ! Par ailleurs, nous ne définissons pas de correspondance entre des critères qualitatifs ordinaux [bas ; élevé] et des intervalles quantitatifs [0-100 ; 100-200] car tout est relatif. Par exemple, un coût de 100 Euros pour la fabrication d'une paire de lunette ordinaire est élevé, mais il est bas pour une paire de lunette de luxe. Ainsi, ici, l'estimation des critères d'aide à la décision est libre moyennant le respect d'une règle de bon sens : pour un projet donné, chaque critère doit être estimé selon une unité de mesure quantitative unique. Par exemple, si le manageur définit un critère de

coût en Euros, et un critère de temps en heures, alors tous les experts font des estimations du coût en Euros et de temps en heures.

En ingénierie, un employé a, en moyenne, une durée de vie fonctionnelle inférieure à celle d'un produit de haute technologie. Les employés changent de projet, de fonction, d'entreprise ou partent à la retraite bien avant que le prochain avion ou satellite ne soit opérationnel. Le stockage des votes est un moyen de capitaliser une partie des connaissances tacites des experts.

Après que le manageur a généré les communautés d'exigences, défini les critères d'aide à la décision et les unités associées, les experts sont invités à estimer chacune des communautés d'exigences. Dans la section 5.3, nous avons postulé que le degré d'ambiguïté est relatif à la richesse du contexte auquel appartient l'énoncé d'une exigence. Un énoncé isolé est probablement plus ambigu qu'un énoncé contextualisé. Ici, comme l'illustre la figure 6.20, le contexte d'une exigence est défini comme la conjonction de trois perspectives :

- **le contexte du document.** Le contexte du document correspond à l'ensemble du texte contenu dans le document prescriptif dont l'exigence est extraite. Par exemple, l'introduction et le cadre d'application (scope) du document prescriptif sont deux sections informatives qui permettent de mieux interpréter les exigences. De plus, les documents prescriptifs ne sont pas rédigés aléatoirement, l'interprétation d'une exigence est donc plus fiable si l'expert a une vue sur les exigences environnantes qui la précède et la suive.
- **le contexte sémantique & conceptuel.** Le contexte sémantique d'une exigence est l'ensemble des exigences voisines via des relations linguistiques issues de WordNet, tandis que le contexte conceptuel d'une exigence est l'ensemble des exigences voisines via des relations conceptuelles issues de ConceptNet 5. Cela permet, entre autres, la mise en relation des exigences qui traitent d'un même concept linguistique ou ontologique, mais qui sont initialement éparpillées dans divers documents prescriptifs.
- **Le contexte des références transversales.** Le contexte des références transversales s'applique à une exigence qui fait référence à un document externe applicable. Ainsi, l'interprétation de cette exigence est améliorée si les exigences spécifiées dans le document référencé sont navigables à partir de l'exigence contenant la référence transversale.

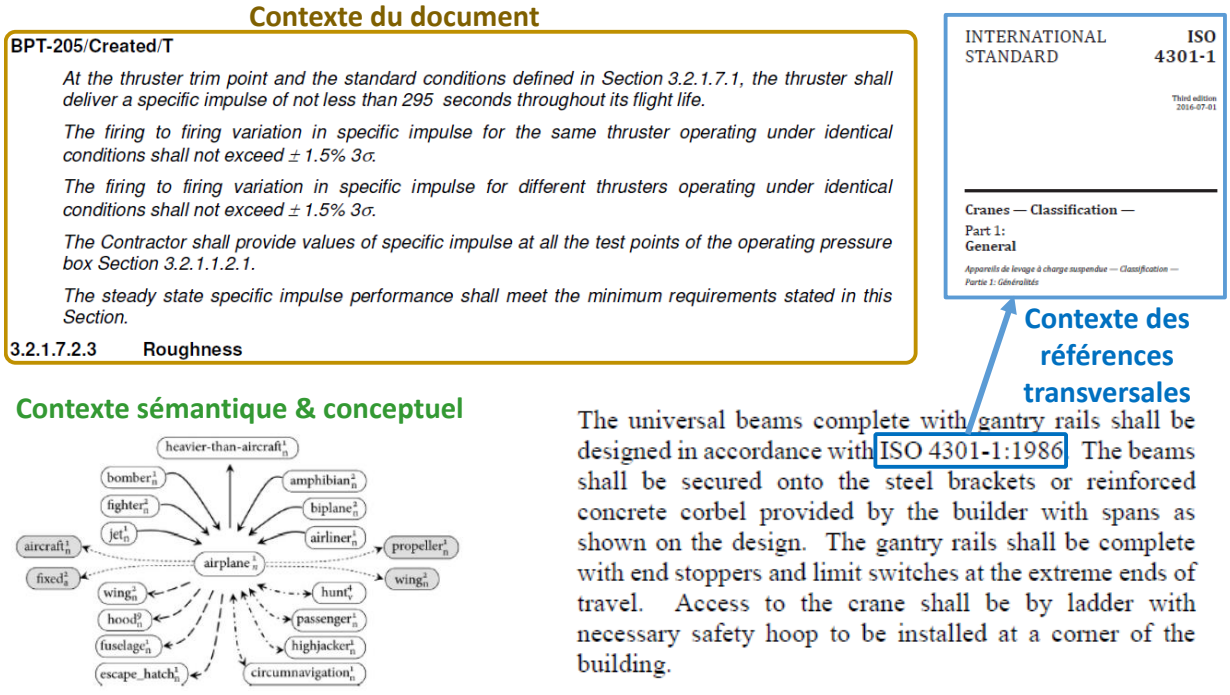


FIGURE 6.20 – Les trois contextes d'une exigence : (1) le contexte du document, (2) le contexte sémantique & conceptuel, et (3) le contexte des références transversales.

Quand un expert souhaite estimer une communauté d'exigences, il peut double cliquer sur le cercle correspondant (Fig. 6.19) afin d'obtenir une vue contextuelle des exigences regroupées dans la communauté. Cette vue contextuelle restitue le contexte sémantique et conceptuel, ainsi que le contexte des références transversales sous la forme unique d'un graphe (Fig. 6.21). Dans ce graphe, on retrouve les exigences reliées par des relations sémantiques – arêtes rouges –, des relations conceptuelles – arêtes vertes –, et des relations de références transversales – arêtes bleues. Ainsi, on constate que parmi les exigences issues du premier document prescriptif – sommets bleus –, l'exigence $R3$ fait référence à un deuxième document prescriptif dont les exigences – sommets verts – sont reliées à $R3$. De même pour l'exigence $R6$ qui fait référence à un second document prescriptif dont les exigences – sommets oranges – sont connectées à $R6$. Avant d'estimer la communauté, l'expert est incité à faire une lecture des exigences soit en cliquant sur chacun des nœuds, soit en utilisant une liste similaire à celle de la figure 5.6. En effet, la structure de graphe est utile pour analyser les données, en particulier pour identifier des communautés de sommets, mais nous constatons que sa représentation graphique n'apporte pas vraiment d'informations qui permettent d'appréhender les exigences. Quand le nombre d'exigences est restreint, une liste est plus appropriée.

Nous avons aussi indiqué que pour qu'une exigence soit moins ambiguë, un expert a besoin d'avoir son contexte au sein du document prescriptif dont elle est issue. Quand un document est soumis, nous enregistrons deux versions sur le serveur. Une version native – le .DOC par exemple – et une version PDF pour la visualisation. Nous avons vu au début que chaque sommet « Exigence » contient les méta-données du document prescriptif source. Ainsi, quand un expert clique sur l'un des sommets du graphe, l'onglet dans la barre d'outil à droite est mis à jour sur « Doc » et une version PDF du document source est affichée.

Le second onglet « Node » est une liste des propriétés de l'exigence sélectionnée. Parmi ces propriétés on retrouve les méta-données du document source – l'auteur, le titre, la version, etc. –, l'énoncé de l'exigence et sa classe métier.

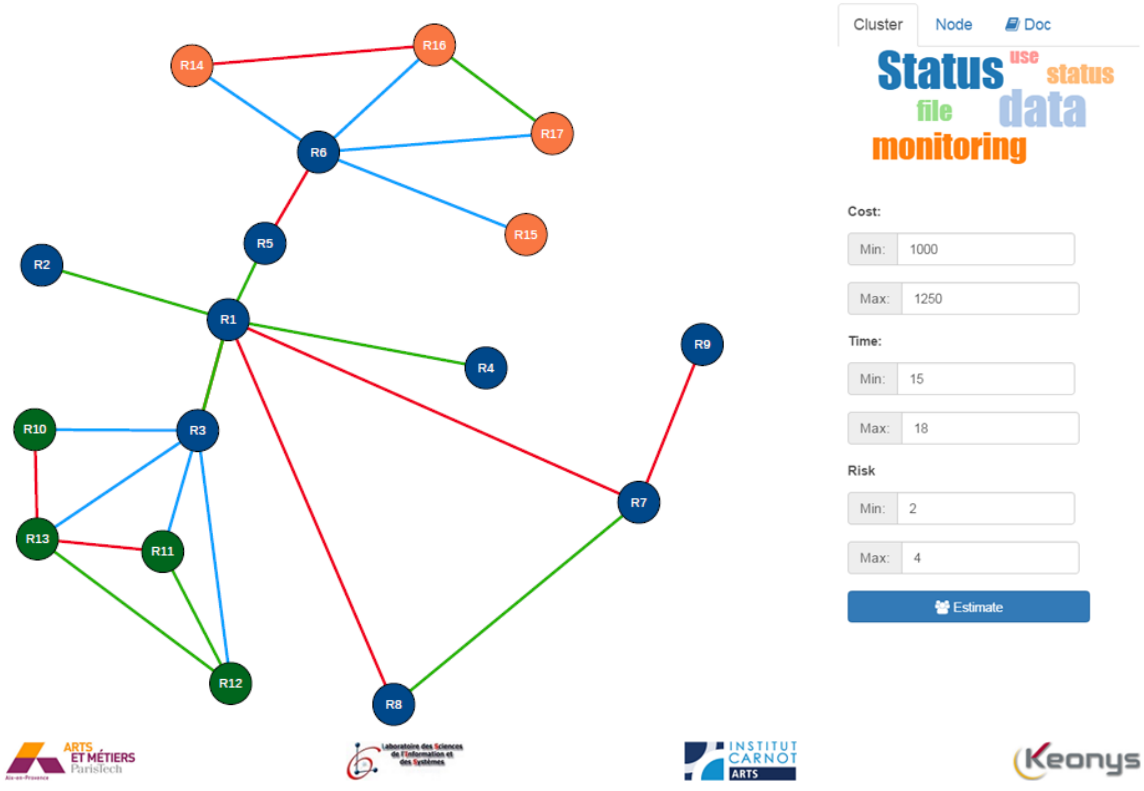


FIGURE 6.21 – Vue contextuelle des exigences qui appartiennent à une communauté.

Prospecter une synthèse des estimations (FT34)

En dernier lieu, l'environnement numérique doit permettre au manageur du projet de prospecter l'espace prescriptif afin qu'il soit à même de prendre des décisions informées. Pour cela, comme l'illustre la figure 6.23, nous proposons un tableau de bord constitué de résumés statistiques qui a deux fonctions : (1) synthétiser les estimations des experts, et (2) simuler des scénarios « *What-if ?* ».

Les techniques d'apprentissage statistique utilisées précédemment ont pour but de prévoir une variable quantitative ou qualitative. Dans cette section, nous appliquons des techniques de statistique descriptive [6] qui décrivent les estimations des experts via des résumés graphiques. Les critères d'aide à la décision étant configurés par le manageur du projet, nous ne savons pas à l'avance combien de variables sont à analyser. La seule information que nous avons c'est que les critères sont des variables quantitatives. Ainsi, initialement nous disposons des données suivantes (Fig. 6.22 pour une illustration) :

– un ensemble de communautés d'exigences :

$$\mathbb{C} = \{C_1, C_2, ..., C_m\} \tag{6.10}$$

– un ensemble d'experts occupants une fonction métier :

$$\mathbb{E}m = \{Em_1, Em_2, ..., Em_n\} \tag{6.11}$$

– un ensemble d'experts occupants une fonction transverse :

$$\mathbb{E}t = \{Et_1, Et_2, ..., Et_o\} \tag{6.12}$$

– un ensemble de critères quantitatifs continus :

$$\mathbb{K} = \{[K_1min; K_1max], [K_2min; K_2max]..., [K_pmin - K_pmax]\} \tag{6.13}$$

– un ensemble d'estimations réalisées par les experts à fonction métier :

$$\mathbb{V}m_{\{i=1:p;j=1:m;k=1:n\}} \tag{6.14}$$

– un ensemble d'estimations réalisées par les experts à fonction transverse :

$$\mathbb{V}t_{\{i=1:p;j=1:m;k=1:o\}} \tag{6.15}$$

Les variables à décrire sont les critères d'aide à la décision. Certaines entreprises se contenteront d'estimer un coût général calculé avec leurs propres hypothèses ; ou peut-être préféreront-elles le diviser en différents coûts, voire introduire de nouveaux critères tels que le temps ou le risque. C'est pour cette raison que les résumés statistiques graphiques que nous avons choisis sont uni-, bi- et multidimensionnels.

L'exemple volontairement simplifié de la figure 6.23 se positionne à la frontière de la statistique multidimensionnelle avec trois critères d'aide à la décision : le coût, le temps, et le risque (Fig. 6.23 A). Ces variables n'ont pas de sémantique particulière, elles peuvent être interprétées d'une quelconque façon. Il y a différents types de coûts et diverses façons de des calculer, chaque entreprise a ses propres hypothèses.

Le critère sélectionné pour la variable $X1$ est représenté par le diagramme de colonnes (Fig. 6.23 B). Chaque paire de colonnes correspond à une communauté d'exigences dont les critères ont été estimées par les deux catégories d'experts, métiers et transverses. Chaque colonne rouge représente la moyenne de la variable $X1$ pour les experts métiers, tandis que chaque colonne verte représente la moyenne de la variable $X1$ pour les experts transverses. Par exemple, si le manageur sélectionne le critère « coût » pour la variable $X1$, alors nous obtenons le coût moyen estimé par les experts métiers en vert, et le coût moyen estimé par les experts transverses en rouge. On peut ainsi voir s'il y a un consensus ou non entre

6. On parle aussi de statistique exploratoire ou d'analyse des données.

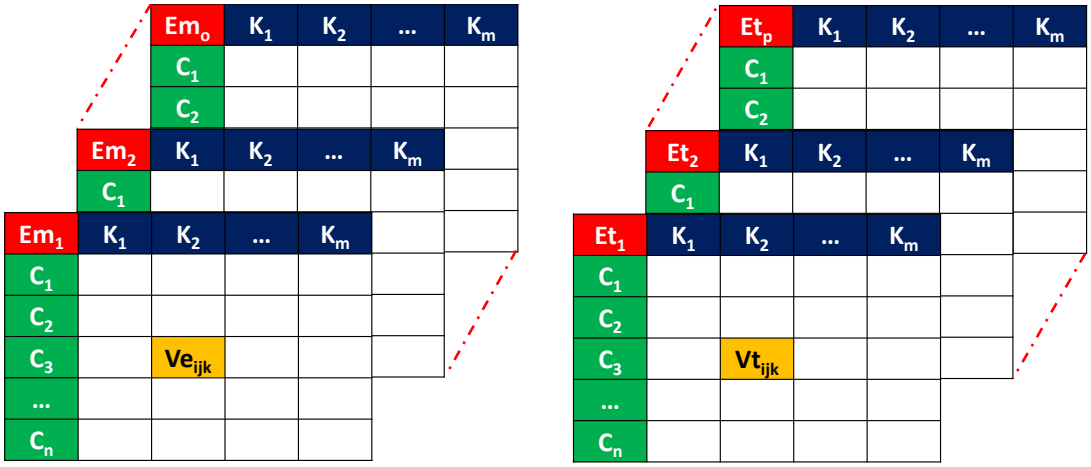


FIGURE 6.22 – Données disponibles après l'estimation des critères d'aide à la décision associés à chaque communauté d'exigences. On retrouve deux types d'estimations : les estimations des experts qui occupent une fonction métier (à gauche) et une fonction transverse (à droite). La valeur Vm_{ijk} (resp. Vt_{ijk}) correspond à l'estimation du critère K_i de la communauté C_j par l'expert à fonction métier Vm_k (resp. transverse Vt_k) .

les diverses fonctions de l'entreprise. La moyenne entre les fonctions métiers et transverses est représentée par un trait horizontal discontinu pour chaque communauté d'exigences. Ici nous n'avons pas d'échantillon d'estimations, mais dans un cas d'application concret nous pourrions tracer des mesures de dispersions. Les intervalles de confiance à 95 % par exemple.

Le diagramme de colonnes (Fig. 6.23 C), lui, représente la variable $X1$ cumulée. Ainsi, si le manageur choisit d'analyser le coût, alors la barre rouge (resp. verte) résume le coût moyen cumulé selon les experts métiers (resp. transverse). Cela donne une idée approximative du coût total de l'espace prescriptif. Idem, si le temps est associé à la variable $X1$, alors le manageur observe le temps total estimé.

Alors que les deux premiers diagrammes de barres synthétisent une seule variable, le nuage de points (Fig. 6.23 D) représente deux ou trois variables quantitatives et une modalité. Les deux premières variables, $X1$ et $X2$ sont liées aux abscisses et aux ordonnées. La variable $X3$, elle, est représentée par le diamètre de chaque point, tandis que chaque couleur est une communauté. Contrairement aux graphiques précédents, le manageur ne peut pas observer les deux perspectives : métier ou transverse. Chaque variable est la moyenne de la moyenne des estimations des experts métiers et de la moyenne des estimations des experts transverses. Autrement dit, par exemple, si le coût est sélectionné pour la variable $X1$, alors le coût représenté selon les abscisses est la moyenne du coût moyen métier $X1_{metier}$ et du coût moyen transverse $X1_{transverse}$. Ici, le manageur a respectivement choisi d'associer le coût, le temps et le risque moyen à $X1$, $X2$ et $X3$. Avec ce type de visualisation bi- et tri-dimensionnelle, on peut observer des tendances de corrélation ; ici, le coût semble être corrélé au temps. Plus le temps nécessaire au développement d'une communauté d'exigences est important, plus le coût le coût de la communauté est élevé.

Les trois résumés statistiques présentés ci-dessus sont basiques, ils ne nécessitent aucune connaissance pour être interprétés. C'est d'ailleurs l'une des raisons de notre choix. Un décideur accorde de la confiance aux choses simples. Cependant, bien que de nombreuses informations soient déjà là pour nourrir la décision du manageur, nous utilisons une quatrième représentation (E). Ce graphique est utile pour visualiser des données de dimension supérieure à trois que l'on ne peut pas observer dans le nuage de points (D). La projection des données quantitatives appartenant à un espace à N dimension dans un espace à deux dimensions se fait au moyen de l'analyse en composantes principales (*Principal Component Analysis*, en anglais) [Everitt and Hothorn, 2011].

L'analyse en composantes principales consiste à transformer N variables quantitatives en M nouvelles variables, les composantes principales, qui sont des combinaisons linéaires des N variables ini-

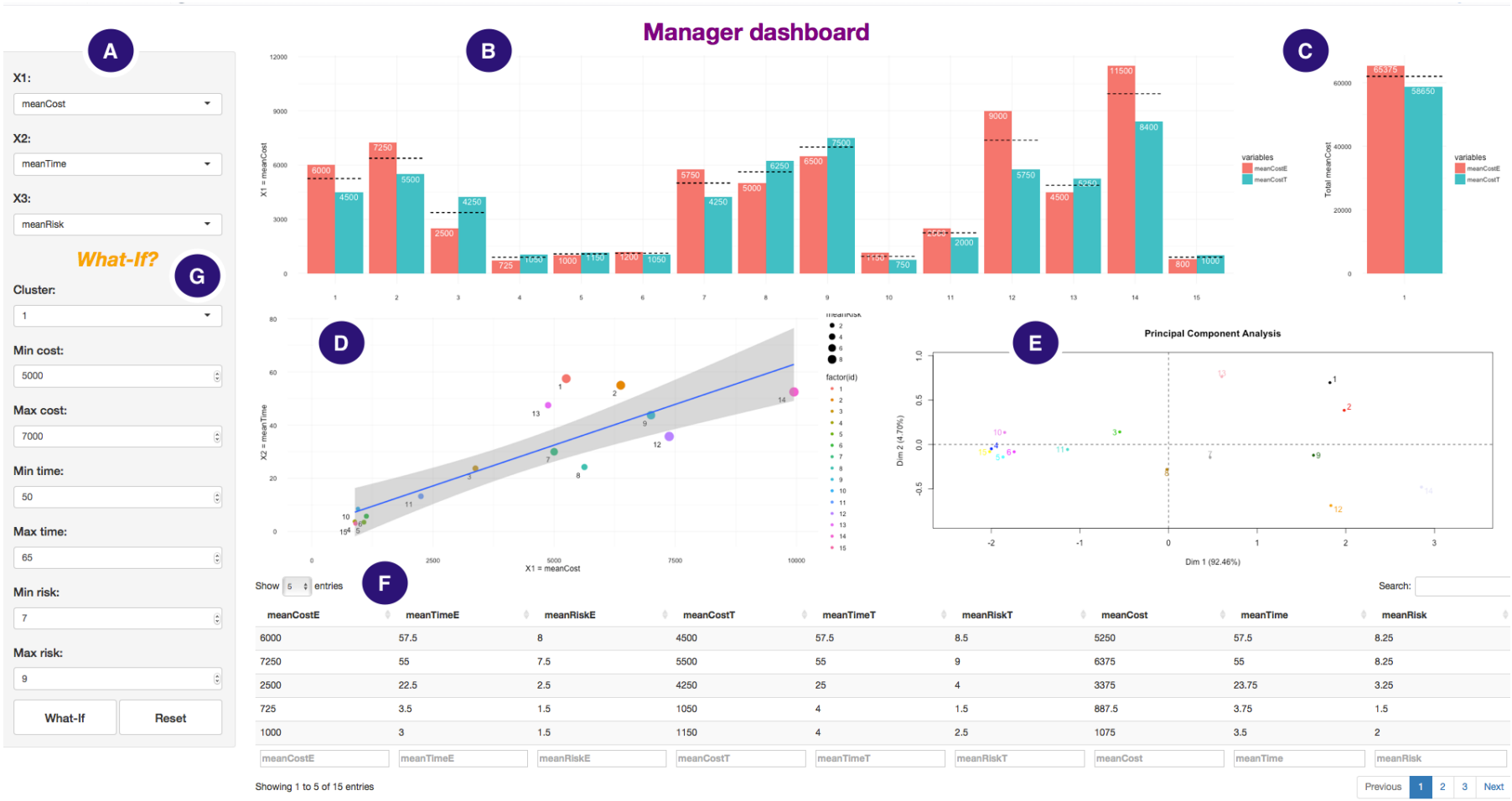


FIGURE 6.23 – Tableau de bord d'un manageur constitué de résumés statistiques des estimations des experts.

tiales. Techniquement, cela consiste à étudier la forme d'un nuage de points dans un hyper-espace afin de trouver une image simplifiée la plus fidèle possible, c'est-à-dire trouver le sous-espace qui résume au mieux les données. Une image simplifiée est de qualité si sont inertie – la variance généralisée à plusieurs dimensions – est minimisée. Par exemple, la figure 6.24 illustre la première (en bleu) et la seconde (en vert) composante principale – des droites de dimension 1 – pour lesquelles l'inertie est minimale, ce qui permet de résumer fidèlement les données initiales – un nuage de points de dimension 2.

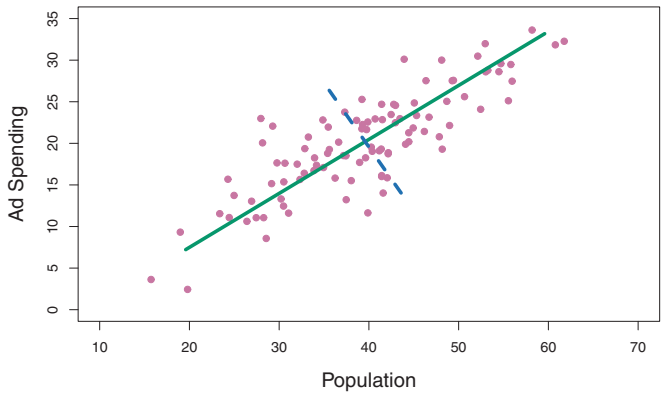


FIGURE 6.24 – Première et seconde composantes principales qui résument le plus fidèlement les données [James et al., 2013].

Le tableau (F) est la représentation numérique des graphiques. Les six premières colonnes correspondent à la moyenne de chaque critères selon les experts métiers et les experts à fonction transverse, tandis que les trois dernières sont les moyennes des moyennes pour chaque critère. Enfin, chaque ligne correspond à une communauté d'exigences.

Pour terminer, dans la barre d'outils (G), le manageur peut sélectionner une communauté et faire ses propres estimations avant d'observer les conséquences en simulant un scénario « What-if ? ».

Pour des contraintes de temps, le tableau de bord a été programmé avec le langage d'analyse statistique R. Plus précisément, nous avons utilisé la librairie Shiny qui permet de développer des applications Web avec un côté serveur en R et un côté client Web en HTML 5 et CSS 3. Les résumés statistiques ont été réalisés à partir d'un fichier de données au format CSV. Enfin, les bibliothèques Ggplo2 ont servi au développement des graphiques uni- et bi-dimensionnels, tandis que FactorMineR a été préféré pour réaliser l'analyse en composantes principales. Aujourd'hui, cette partie logicielle est donc isolée de notre application Web Java EE. Néanmoins, le fait d'avoir opté pour un langage de programmation qui a été spécialement conçu pour manier des données nous à permis d'obtenir, dans les plus brefs délais, un premier aperçu de la solution. Certes, comme chacune des briques logicielles exposées jusqu'ici, le tableau de bord est perfectible, mais il nous donne une vision quasiment complète du prototype de l'environnement numérique de synthèse des exigences. Des représentations graphiques semblables pourront être réalisées avec la librairie de visualisation D3.js que nous utilisons pour prototyper notre méthode.

6.4 Expérimentation

Expérimentation 1 - Classification des exigences.

L'expérimentation permettant d'évaluer la fonction de classification qui assigne une catégorie métier à chaque exigence se fait en deux temps. D'une part, nous présentons la phase de sélection des caractéristiques discriminantes. D'autre part, nous faisons un test de classification sur un nouveau corpus d'exigences.

Dans une expérience parfaite de classification, nous ne devrions utiliser aucune information du corpus de test pour estimer la fonction de classification. Ainsi, nous avons construit trois corpus. Le corpus d'apprentissage constitué des phrases qui ont été extraites des livres. Le corpus de développement pour

trouver les bonnes valeurs des paramètres comme le nombre de caractéristiques discriminantes à conserver. Enfin, quand tous les paramètres sont définis, nous utilisons un corpus de test pour exécuter une dernière simulation sur des exigences inconnues par la fonction de classification. Les performances doivent alors être représentatives de celles que nous devrions obtenir en phase opérationnelle. Par ailleurs, le fait d'utiliser trois corpus différents réduit les chances de sur-apprentissage et, par conséquent, d'obtenir des résultats sur-estimés. Le corpus de développement est constitué de 289 exemples – 70 ME, 81 EE, 72 CS, et 66 RAMS –, tandis que le corpus de test comprend 200 exemples constitué de 50 exigences par catégories.

Avant de classifier automatiquement les exigences, il est bon de s'assurer que les exemples sont objectifs. En effet, un corpus est très dépendant de la personne qui le construit, il faut donc vérifier que la définition de chaque classe n'est pas dépendante de cette personne. Pour cela nous avons catégorisé manuellement les 289 exigences du corpus de développement avant de demander à un second chercheur externe à notre projet d'en faire de même. Les résultats sont présentés dans la table de contingence de la figure 6.25. On peut alors mesurer le coefficient Kappa qui correspond au taux de concordance inter-observateurs, c'est-à-dire entre les deux juges indépendants. Le test de kappa K (Eq. 6.18) se calcule à partir du taux de paires concordantes observées (P_{obs}) (Eq. 6.16) et du taux de concordance aléatoire (P_a) (Eq. 6.17). Le test de Kappa est supérieur à 0,81, le taux de concordance inter-observateurs est donc très bon.

$$P_{obs} = \sum_{i=1}^{r} P_{ii} = \frac{1}{N} \sum_{i=1}^{r} n_{ii} \approx 0.934 \tag{6.16}$$

$$P_a = \sum_{i=1}^{r} P_{i.} P_{.i} = \frac{1}{N^2} \sum_{i=1}^{r} n_{i.} n_{.i} \approx 0.251 \tag{6.17}$$

$$k = \frac{P_{obs} - P_a}{1 - P_a} = \frac{0.934 - 0.251}{1 - 0.251} \approx 0.911 \tag{6.18}$$

		EXPERT 1 [R]				
		ME	EE	CS	RAMS	TOTAL
EXPERT 2 [Y]	ME	70	5	0	0	75
	EE	0	74	1	4	79
	CS	0	0	67	3	70
	RAMS	0	2	4	59	65
TOTAL		70	81	72	66	289

FIGURE 6.25 – Table de contingence qui résume les classifications manuelles des exemples du corpus de développement.

Pour choisir l'algorithme d'apprentissage et le nombre de caractéristiques discriminantes, nous avons réalisé plusieurs classifications. Chaque classification correspond à la combinaison d'un algorithme d'apprentissage et d'un nombre spécifique de caractéristiques retenues pour entrainer l'algorithme. Comme cela a été expliqué précédemment, nous avons choisi un algorithme rigide, un modèle Bayésien (NB), et un algorithme flexible, une machine à vecteurs supports (SVM). Chacun de ces modèles a été entraîné avec les 100, 500, 1000, 1500, 2000, 2500, 3000, 5000 et 10 000 caractéristiques les plus discriminantes (Fig. 6.26). La courbe verte, respectivement orange, représente l'évolution de la F-mesure en fonction du nombre de caractéristiques discriminantes sélectionnées pour entraîner l'algorithme NB, respectivement SVM.

Les tableaux 6.3 et 6.4 résument les performances obtenues pour chaque expérience. L'algorithme NB entraîné avec les 1000 caractéristiques les plus discriminantes a permis d'obtenir un taux de classifications correctes de 71.6%. Cependant, avec un taux de classifications correctes de 74%, c'est l'algorithme SVM entraîné avec les 2000 caractéristiques les plus discriminantes qui a donné la meilleure

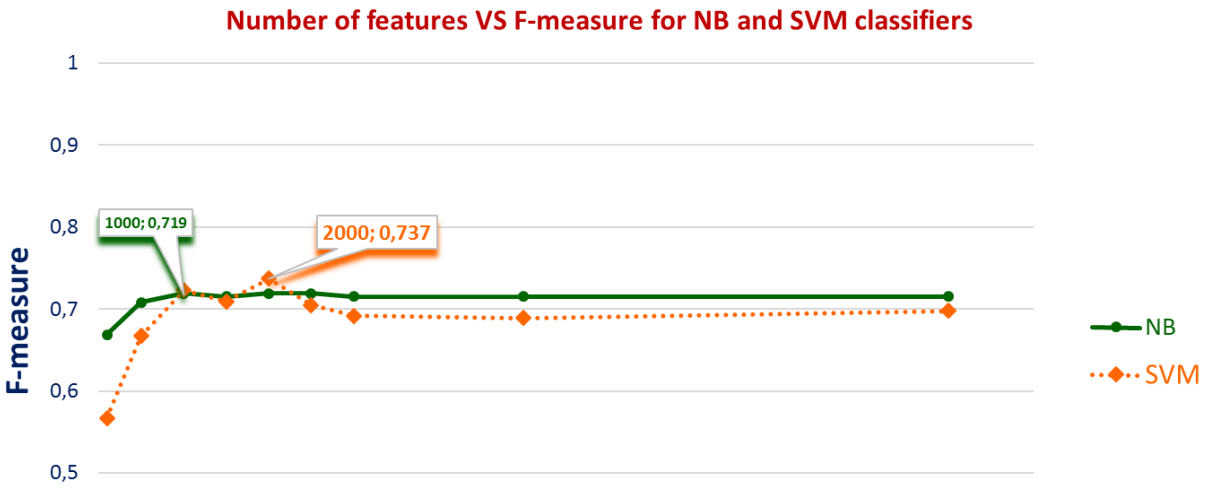


FIGURE 6.26 – Évolution de la F-mesure en fonction du nombre de caractéristiques discriminantes sélectionnées pour entraîner les algorithmes.

fonction de classification. Autrement dit, en moyenne, la fonction de classification fait 26 erreurs sur 100 exigences à classifier.

# Features	Accuracy	Precision	Recall	F-measure
100	0.664	0.719	0.664	0.669
500	0.705	0.726	0.706	0.708
1000	0.716	0.734	0.716	0.719
1500	0.712	0.729	0.713	0.715
2000	0.716	0.734	0.716	0.719
2500	0.716	0.734	0.716	0.719
3000	0.712	0.729	0.713	0.715
5000	0.712	0.729	0.713	0.715
10000	0.712	0.727	0.713	0.715

TABLE 6.3 – Performances du modèle estimé avec l'algorithme NB en fonction du nombre de caractéristiques sélectionnées.

Le tableau 6.5 correspond à la matrice de contingence pour la meilleure fonction de classification. On constate que la plupart des erreurs de prévisions occurent pour la catégorie RAMS. 19 (12+7) exigences appartenant à la catégorie RAMS ont été classifiées dans une autre catégorie, tandis que 26 (12+8+6) exigences ont été classifiées dans la catégorie RAMS alors qu'elles appartiennent à l'une des trois autres catégories. Ce constat est normal car la fiabilité, la disponibilité, la maintenance et la sécurité sont des métiers qui sont transverses. On parle de la fiabilité d'un système mécanique, électrique ou d'une machine à informations. Cependant, on constate aussi que 25 exigences de mécanique ont été mal classifiées, ceci illustre la difficulté de classifier des phrases sur la base de quelques termes discriminants. Sur la dizaine de mots qui constituent l'énoncé, deux ou trois sont des caractéristiques qui appartiennent à des catégories différentes.

Nous étions aussi curieux d'évaluer l'impact que peut avoir la normalisation des termes à leur racine (*stemming*, en anglais). Ainsi, nous avons modifié l'étape de pré-traitement et sélectionné les 2000 caractéristiques les plus discriminantes pour entraîner l'algorithme SVM. Le pourcentage de classifications correctes que nous avons obtenu était de 68.5%, soit légèrement inférieure à celui obtenu sans la racinisation (74 %). Cela confirme [Manning et al., 2008] le faible impact de la racinisation sur l'exercice de classification de textes. Nous n'avons donc pas considéré cette phase de pré-traitement.

La fonction de classification la plus performante sur le corpus de développement a ensuite été testée

# Features	Accuracy	Precision	Recall	F-measure
100	0.567	0.666	0.567	0.557
500	0.667	0.69	0.668	0.667
1000	0.723	0.74	0.723	0.725
1500	0.709	0.725	0.709	0.71
2000	0.737	0.753	0.737	0.738
2500	0.705	0.73	0.706	0.707
3000	0.692	0.72	0.692	0.695
5000	0.689	0.717	0.699	0.703
10000	0.698	0.711	0.699	0.701

TABLE 6.4 – Performances du modèle estimé avec l'algorithme SVM en fonction du nombre de caractéristiques sélectionnées.

a	b	c	d	⟵ classified as
45	10	3	12	**a = ME**
4	64	5	8	**b = EE**
1	8	57	6	**c = CS**
0	12	7	47	**d = RAMS**

TABLE 6.5 – Table de contingence des résultats obtenus sur le corpus de développement en ayant entraîné l'algorithme SVM avec les 2000 caractéristiques les plus discriminantes.

sur les exigences inconnues du corpus de test afin d'avoir une approximation des performances que l'on peut espérer en phase opérationnelle. La matrice de contingence est donnée dans le tableau 6.6. Les résultats obtenus sont sensiblement meilleurs que ceux obtenus en phase de développement puisque le taux de classifications correctes est de 76% (74% en développement). La précision (0.78), le rappel (0.76) et la F-Mesure (0.76) sont eux aussi légèrement meilleurs. Le fait qu'il n'y a pas de variation significative entre les résultats obtenus sur le corpus de développement et ceux obtenus sur le corpus de test nous conforte dans l'idée que la fonction de classification n'est pas sur-apprise.

a	b	c	d	⟵ classified as
30	10	5	5	**a = ME**
3	39	0	8	**b = EE**
0	2	47	1	**c = CS**
0	11	3	36	**d = RAMS**

TABLE 6.6 – Table de contingence des résultats obtenus sur le corpus de test en ayant entraîné l'algorithme SVM avec les 2000 caractéristiques les plus discriminantes.

Expérimentation 2. Partitionnement des exigences

Pour évaluer la qualité du partitionnement des exigences avec l'algorithme Linlog, nous avons utilisé trois documents prescriptifs. Le premier document prescriptif, un fichier PDF, prescrit 358 exigences. Le deuxième document, un fichier Word, contient 1187 exigences, tandis que le troisième document est une feuille de calculs Excel qui spécifie 73 exigences. 1618 exigences ont donc été partitionnées, soit un volume de données qui est largement supérieur à ceux traités dans l'état de l'art.

Le nombre de partitions identifiées est la première information qui résulte du partitionnement. Comme le synthétise le tableau 6.7, dans cette expérience, 109 partitions ont été trouvées. Si les 1618 exigences sont partitionnées en 109 partitions, alors on peut espérer qu'il y ait approximativement 15 exigences par communauté. Or, en regardant de plus près le chiffre qui est associé à chaque commu-

nauté, on constate que la plus grosse partition contient 169 exigences, tandis que la plus petite partition contient seulement 2 exigences. Les communautés regroupant une centaine d'exigences seront difficilement estimables par les analystes. On ne fait que dériver le problème du volume des données à une échelle moindre.

De plus, la plus grosse partition ne contient pas exactement 169 exigences. En effet, il y a une partition « *Other Topics* » qui, elle, regroupe toutes les exigences qui n'appartiennent à aucune autre partition. Dans cette expérience, cela représente 436 exigences, soit environ un peu plus d'un quart du corpus initial. Attention, on n'obtient pas les 1182 (1618 - 426) exigences restantes en faisant la somme des exigences contenues dans les autres partitions, et ce, car une exigence peut appartenir à plusieurs partitions.

Shall be Installed,169	System Shall,153	Cable Shall,112	Shall be Used,110
Specified Shall,107	Contractor Shall,101	Cables Shall,100	Data Shall,98
Conduit Shall,94	Installation Shall,90	Conductors Shall,77	Shall be in Accordance,54
Shall be made by Means,53	Shall include the Following,52	ECV Data Product,48	Cover Shall,47
Conductor Shall,41	Shall be Connected,40	Mm Shall,38	Shall be Mounted,38
Equipment Shall,36	Shall Comply,36	Shall be Fixed,36	Type and Shall,36
Luminaries Shall,35	Boxes Shall,34	Joints Shall,33	Cables are Installed,32
Shall have a Minimum,31	Draw-box Shall,30	Outlets Shall,30	Shall be Approved,30
Shall be provided in the Switchboard,30	Channels Shall,29	Poles Shall,29	System must Allow,28
Trench Shall,28	Cover Plates Shall,27	Draw-boxes Shall,27	Box Shall,26
Connection Shall,26	Ends Shall,26	L2 Data,26	Services Shall,26
RMS Sample,23	Unless otherwise Specified,21	Contractor will be Required,19	Earth Conductors,19
File System,19	System s Software,19	Earth Conductor,18	ECV Production,17
System must have the Ability,16	Earth Wire,15	Copper Conductors,14	Flexible Conduit,14
Metal Channels,14	Draw-boxes Installed,13	Specified in DPMv2,13	Approved by the Department,12
Power Cables,12	Screwed Conduit,12	Cable Trenches,11	Installed in Accordance,11
Power Skirting and Installed,11	Processing Algorithms,11	Secured by Means,11	Cable Trays,10
Conductors are Connected,10	Conduit Ends Shall,10	Metallic Conduit,10	Outlets are not Installed,10
Permission of the Department,10	System Specification,10	Cable Channels,9	Cable Joints,9
Conduit Boxes,9	Non-metallic Conduit,9	Supply and Installation,9	Clamps Shall be Used,8
Damage Cables,8	Installed on the Surface,8	Required by the Department,8	Telephone Cables,8
Cable Ducts,7	On-line Data Archive Shall,7	Overhead Conductors,7	Requirements Specify,7
Complete Installation,6	Electrical Installation,6	Future Processing,6	Metal Channel,6
Shall be Used to Fix,6	Accessories Used,5	Copper Earth Conductors,5	Earth Conductor Shall not be Installed,5
Plugs may be Used,5	Processing History,5	Processing Step,5	Completion of the Installation,4
Conduits and Accessories,4	Product Versions,4	Diameter Conduits,3	Following Conditions,3
Interface Requirements,3	Product Description,3	Fix Conduits,2	Method Used Complies with the Performance Requirements,2
Other Topics,436			

TABLE 6.7 – Partitions d'exigences obtenues avec l'algorithme Linlog.

Par ailleurs, si l'on observe bien le nom attribué à chaque partition, on retrouve bien l'idée de phrases pleines expliquée dans la section 6.3. Néanmoins, ce concept n'est pas vraiment adapté à notre contexte, car on voit bien que cela correspond à des séquences de termes qui se répètent. Or, dans notre contexte, les séquences de termes ne sont pas les meilleurs indicateurs pour identifier les partitions, car la plupart d'entre elles correspondent aux co-occurrences « Sujet + Verbe Modal » telles que « *system shall . . . .* », « *contractor shall . . . ,* » ou « *conduit shall . . . .* ». Cela est naturellement observable quand on regarde les exigences d'une partition en particulier (Fig. 6.27). Ce type de solution peut potentiellement être utile pour identifier les différents sujets grammaticaux sur lesquels les exigences s'appliquent, mais elle ne permet pas de trouver des groupes d'exigences qui ont une réelle affinité. La qualité des partitions obtenues est aussi due au fait que Linlog, comme beaucoup d'autres techniques de partitionnement de textes, s'appuie seulement sur une similarité linguistique au niveau lexical, voire syntaxique dans certains cas.

Cette expérience est suffisamment explicite pour nous permettre de conclure que le partitionnement avec Linlog ne convient pas à notre problématique. Plus généralement, toutes les techniques de partitionnement basées sur une fonction de similarité mesurée dans un modèle vectoriel sont potentiellement inefficaces compte tenu de la pauvreté du vocabulaire d'une phrase.

Expérimentation 3. Détection de communautés d'exigences

Après avoir évalué notre fonction de classification des exigences et l'algorithme Linlog de partitionnement sémantique des exigences ; ici, nous réalisons une troisième expérience pour évaluer la technique de découpage du graphe engendré par les exigences reliées par des relations sémantiques, conceptuelles

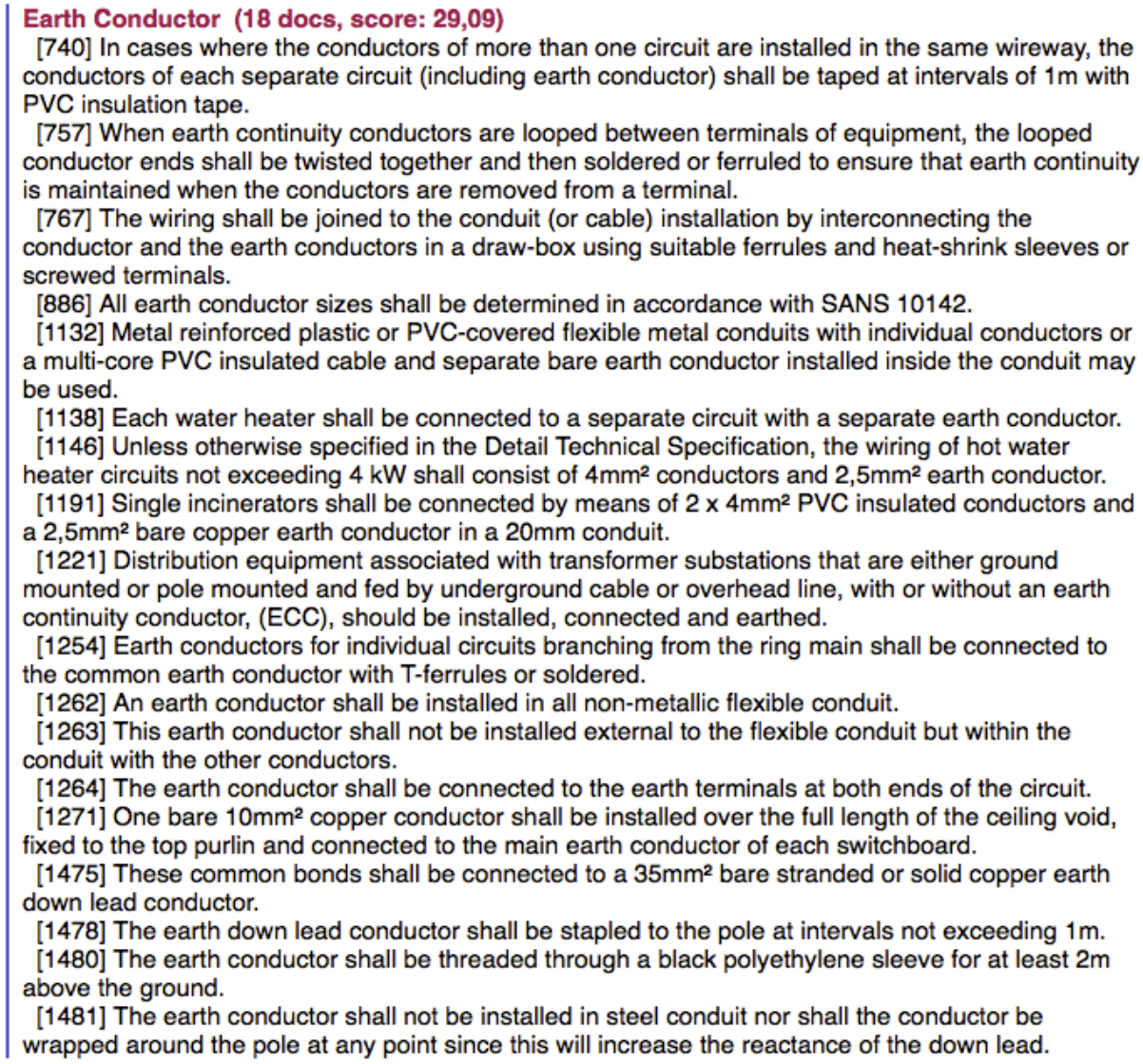


FIGURE 6.27 – Exemple d'une partition contenant 18 exigences obtenue avec Linlog.

et transversales.

Dans un premier temps, nous avons tenté d'utiliser des données représentatives des pratiques industrielles, c'est-à-dire des spécification de plusieurs centaines de pages prescrivant environ un millier d'exigences. Les opérations décrites dans l'algorithme 1, lesquelles consistent à chercher les interdépendances sémantiques et conceptuelles entre les exigences, se sont avérées trop gourmandes en temps de calculs. Pour être concret, sur une machine constituée d'un processeur i7 cadencé à 2.7 GHz avec 8GB de RAM, après 5 jours, le graphe n'était toujours pas construit. Ne disposant pas d'une seconde machine, nous avons donc stoppé le processus pour poursuivre nos travaux de recherche. Une deuxième expérience avec une centaine d'exigences a été tentée, mais après trois jours le graphe n'était toujours pas construit. Nous avons donc là aussi arrêté le traitement. Par ailleurs, au delà de plusieurs jours, la librairie Java WordNet Interface (JWI) [Finlayson, 2014] qui permet d'interroger le thésaurus WordNet s'arrête de fonctionner pour des raisons que nous n'avons pas eu le temps d'étudier.

Ceci n'est pas une limite théorique fondamentale mais pratique. En effet, de nombreux éléments peuvent aider à diminuer le temps de traitement. On peut par exemple utiliser une machine plus puissante, ou distribuer les opérations avec des moyens algorithmiques – Fork/Join, Map/Reduce, etc. –, technologiques – Hadoop, Spark, Flink, etc. – et matériels – processurs multi-coeurs, cluster de machines, etc. – adéquats. Nous avons envisagé de restructurer notre prototype selon une architecture de

calcul basée sur la technologie Spark, mais cela soulevait d'autres problèmes tels que l'accostage de notre base de données NoSQL Neo4j avec Spark qui utilise la technologie HDFS. Par ailleurs, cela nécessite une ré-ingénierie du code informatique. Une autre piste pour réduire le temps de traitement est de trouver des heuristiques pour diminuer le nombre d'opérations sans trop dégrader les performances. On peut s'interroger sur certains points tels que : est-il nécessaire d'étudier tous les types de relations linguistiques (méronymes, holonymes, hyponymes, etc.) ? est-il nécessaire de vérifier s'il existe des relations d'interdépendance pour chacun des termes clés ou peut-on se limiter aux noms ? etc. Ces questions nécessitent des tests sur plusieurs mois étant donné qu'une simulation occupe plusieurs jours. Nous avons par exemple essayé de construire le graphe en se limitant aux interdépendance entre les noms plutôt que tous les termes clés, mais cela ne nous a pas permis d'obtenir de résultats dans les délais espérés.

Cette limite ne remet pas en cause notre proposition d'un point de vue théorique. C'est une limite pratique qui, pour être outrepassée, nécessite une charge de travail non négligeable pour revoir l'architecture de notre prototype. Pour démontrer que ce sont des limites pratiques et non théoriques, nous avons construit une expérience avec des données qui, certes, sont extrêmement limitées en nombre, mais elles sont suffisamment pertinentes pour montrer l'intérêt du découpage de graphe comme moyen de segmentation des exigences.

La première étape de notre expérience a été de collecter 14 exigences (Tab. 6.8) dans un document prescriptif. Parmi les 14 exigences réunies, 7 spécifient des éléments relatifs à l'aéronautique (R1-R7), tandis que les 7 autres prescrivent des propriétés relatives à des objets électriques (R8-R14).

R1	The Spacecraft Composite shall be designed according to the tailored or integral ECSS and CCSDS standards.
R2	When the aircraft is on ground and its ground speed is lower than Vr, the clearance between the lowest point of the aircraft and the ground shall be equal or greater than Hprop_tip = 18 cm (Far Part 23 CS.25.925).
R3	When the plane is taking-off and after the nose up rotation, the fuselage rear upsweep point shall be at a ground height greater than a desired value to prevent the rear fuselage from striking the ground at take=off.
R4	When landing, the reaction factor shall not exceed 1.5.
R5	When touching-down with a vectical speed Vz that is equal or less than 3 m/s, the LGS shall not bounce 10% above its statically deflected position.
R6	When on ground, the aircraft shall turn its lights on.
R7	The wing of the aircraft shall be made of aluminium.
R8	The battery cell shall be accelerated 1000 times at 100m/s^2 for durations of 16mS.
R9	The voltage regulation accuracy shall be ground testable via a unit connector.
R10	Full protection against short circuit or overload at all outputs shall be provided by limiting the maximum current in any of those lines.
R11	The main bus voltage regulation requirements shall be in accordance with the requirements of PSS-02-10.
R12	The subsystem shall be designed such that in all operating modes where the power available from the solar cell generator exceeds the main bus and battery charge demand the surplus electrical energy is left in the solar arrays.
R13	The subsystem shall be designed such that in all operating modes where the power available from the solar cell generator is still insufficient to satisfy the load demand battery discharge regulators will provide the required electrical power from batteries automatically.
R14	The subsystem shall condition, control, store and distribute electrical power on the ROSETTA spacecraft.

TABLE 6.8 – Les 14 exigences du document prescriptif A.

Nous faisons l'hypothèse que les exigences $R1$ et $R2$ référent à un second document prescriptif externe applicable. Dans ce second document prescriptif nous avons mis deux exigences $R15$ et $R16$ (Tab. 6.9). Ainsi, comme cela est fait semi-automatiquement dans la section 4.3, nous créons quatre relations transversales ($R2$; $R15$), ($R2$; $R16$), ($R3$; $R15$) et ($R3$; $R16$). Jusqu'ici, le graphe d'exigences est alors engendré par $R2$, $R3$, $R15$ et $R16$.

R15	When touching down and for a vertical speed Vz that is equal or less than Vzmax, the landing gear shall deflect to absorb and to dissipate the aircraft kinetic energy without an excessive reaction load to the aircraft or without bottoming out.
R16	When touching down at any acceptable longitudinal ground speed Vx, the landing gear shall deflect to absorb and to dissipate the aircraft kinetic energy.

TABLE 6.9 – Les 2 exigences du document prescriptif B.

Intuitivement, sans consulter les relations dans WordNet ou ConceptNet, étant donné que les exigences traitent de deux thématiques distinctes, nous espérons obtenir deux communautés d'exigences. La première communauté devrait regrouper les exigences $\{R1, R2, R3, R4, R5, R6, R7, R15, R16\}$, car elles devraient être fortement liées par des relations sémantiques et conceptuelles vis-à-vis du concept d'avion (*wing, aircraft, landing, plane, etc.*) en plus des quatre relations transversales. La deuxième communauté, elle, devrait regrouper les autres exigences $\{R8, R9, R10, R11, R12, R13, R14\}$ qui traitent de composants électriques (*battery, discharge, electrical power, short circuit, etc.*).

Le recherche automatique d'interdépendances sémantiques et conceptuelles entre les lemmes clés dont la catégorie grammaticale est un nom a permis d'identifier 64 relations dont : 30 relations sémantiques et 34 relations conceptuelles en plus des 4 relations transversales que nous avions spécifiées manuellement. Le découpage du graphe engendré par les 16 exigences a résulté en deux communautés distinctes. La première communauté regroupe les exigences $\{R1, R2, R3, R4, R5, R6, R7, R9, R15, R16\}$, tandis que la seconde communauté regroupe les exigences $\{R8, R10, R11, R12, R13, R14\}$. Ainsi, seule l'exigence $R9$ n'appartient pas à la communauté espérée. Cette dernière parle de *voltage*, de *regulation* et de *connector*, mais a été affectée à la communauté aéronautique, et ce, parce qu'elle est liée à $R2$, $R3$ et $R6$ via le terme *ground*, lequel fait référence à la terre dans les branchements électriques, mais aussi au sol dans les atterrissages.

Pour les mêmes données, l'algorithme de partitionnement LinLog identifie les 12 communautés suivantes :

- <Aircraft shall> = $R2, R6, R7, R15, R16$
- <Speed> = $R2, R5, R15, 16$
- <Battery> = $R8, R12, 13$
- <Electrical power> = $R12, R13, R14$
- <Ground speed> = $R2, R16$
- <Power available from the solar cell generator> = $R12, R13$
- <Reaction> = $R4, R15$
- <Spacecraft> = $R1, R14$
- <Speed Vz that is equal> = $R5, R15$
- <Touching> = $R15, R16$
- <Voltage regulation> = $R9, R11$
- <Other topics> = $R3, R10$

Comme nous l'avions déjà constaté précédemment, on voit bien que le partitionnement sémantique ne convient pas à des énoncés courts. Il est probable qu'une telle solution soit utile pour segmenter des pages Web ou des documents à partir de séquences de termes ; cependant, ici, plus ou moins chaque sujet dans une exigences est une communauté. Cela revient presque à identifier le sujet grammatical de chaque exigence et à y associer une communauté.

Cette étude des trois approches – classification en métiers, partitionnement sémantique, détection de communautés dans un graphe – permettant de regrouper les exigences en communautés, la solution la

plus pertinente est celle basée sur le découpage du graphe engendré par les exigences.

6.5 Synthèse

Bilan de notre méthode d'exploration des exigences

Dans ce chapitre, nous cherchons une méthode qui puisse permettre à un manageur d'explorer un espace prescriptif régi, par plusieurs centaines ou milliers d'exigences afin de prendre des décisions informées.

Pour assurer le passage à l'échelle il est indispensable de segmenter les exigences pour travailler à un niveau d'abstraction supérieur. Le découpage de graphe, lequel permet de détecter des communautés dont les exigences partagent des affinités de tous types (sémantiques, conceptuelles, références transversales, etc.), est la meilleure solution que nous avons trouvé pour segmenter les exigences. Pour construire le graphe d'exigences à découper, nous cherchons les interdépendances sémantiques issues du thésaurus WordNet et les interdépendances conceptuelles issues de Concept Net 5 qui lient des paires d'exigences via leurs lemmes clés. De multiples représentations graphiques des communautés d'exigences sont mises à disposition des experts qui, pour aider le manager à décider, estiment une valeur minimale et une valeur maximale des critères quantitatifs d'aide à la décision (coût, temps, risques, etc.). Enfin, une synthèse des estimations est présentée au manager grâce à un tableau de bord constitué de résumés issus de la statistique descriptive. Le diagramme de flux d'informations (Fig. 6.28) résume la méthode que nous proposons pour explorer les exigences.

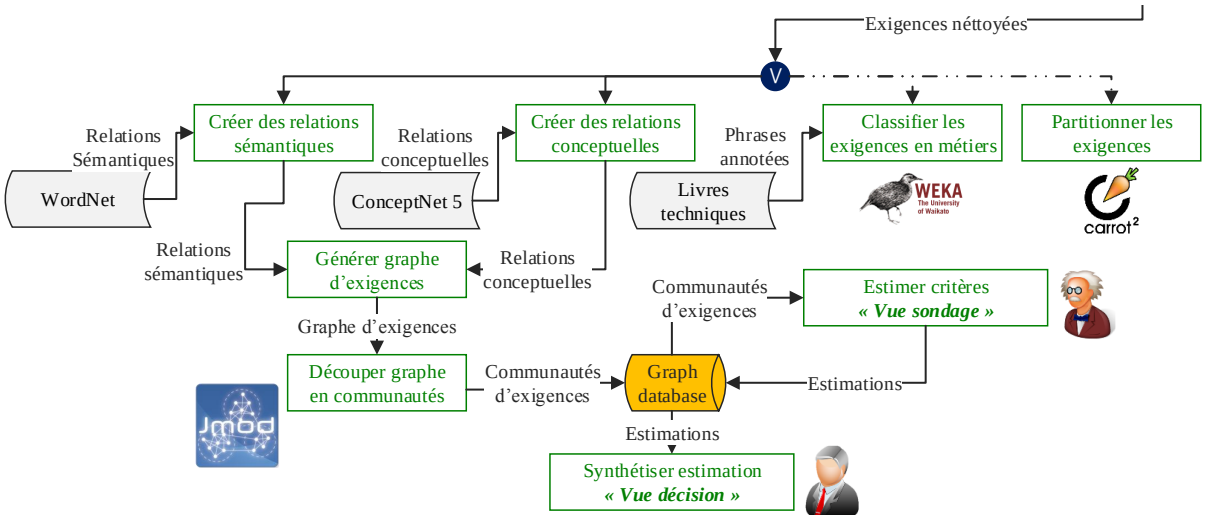


FIGURE 6.28 – Diagramme de flux d'informations qui résume la méthode d'exploration des exigences.

Avant de conclure que le découpage de graphe est la solution la plus efficace pour segmenter les exigences, nous avons étudié trois approches de segmentation : la classification des exigences en métiers, le partitionnement sémantique des exigences, et le découpage de graphe. Nous souhaitons que toutes les fonctions de l'entreprise puissent estimer toutes les exigences, nous avons donc abandonné notre idée initiale, laquelle consistait à demander à chaque métier d'estimer les exigences qui appartiennent à son expertise : les mécaniciens estiment les exigences de la mécanique, les électriciens estiment les exigences de l'électrique, etc. Nous avons néanmoins montré que cela est réalisable en estimant une fonction de classification capable de classifier les exigences au sein de quatre disciplines – mécanique, électronique, CS et RAMS – avec un taux de classifications correctes de 76 %, une précision de 0.78 et un rappel de 0.76. Par contre, nos tests de partitionnement sémantique ont montrés que l'algorithme LinLog est trop dépendant des séquences de termes, lesquelles sont fréquentes dans la syntaxe des exigences. Par exemple, il créé une communauté pour chaque séquence « Sujet + shall ». Par ailleurs, une communauté

par défaut « *Other topics* » regroupe trop d'exigences dans une communauté unique qui fait office de « poubelle ». Enfin les exigences appartiennent à trop de communautés à la foi, et cela peut détériorer la qualité des estimations réalisées par les experts. Deux expérimentations ont illustré les limites de l'algorithme LinLog dans notre application, tandis que le découpage de graphe a permis d'obtenir des résultats espérés sur une étude de cas simplifié.

Qu'est-ce qui est nouveau ou différent ?

Le premier élément nouveau c'est la manière de créer le corpus d'apprentissage nécessaire à la classification automatique des exigences au sein de catégories qui correspondent à des métiers. Dans la littérature, les auteurs qui ont essayé de classifier des exigences se plaignent du manque de données annotées ou de la difficulté pour les collecter. Ici, nous proposons de générer automatiquement les exemples à partir d'un corpus de livres, puis de sélectionner automatiquement les termes caractéristiques discriminants qui sont nécessaires pour entrainer un algorithme d'apprentissage supervisé. Enfin, la fonction de classification est différente des travaux voisins qui se concentrent sur des exigences rédigées en allemand.

La mise en œuvre de l'algorithme LinLog n'est pas totalement nouvelle puisque Sannier [2013] ont déjà étudié son efficacité pour partitionner les titres d'un document prescriptif. Cependant, l'algorithme n'avait jamais été mise en œuvre sur des exigences. Nous avons pu montrer et expliquer pourquoi il ne convient pas à notre problématique.

Le troisième élément nouveau c'est l'utilisation d'un algorithme issu de la théorie des graphes pour segmenter les exigences . L'état de l'art des solutions existantes a montré que toutes les propositions s'appuient sur des métriques de similarité sémantique. Ici, le découpage en communautés basé sur l'optimisation de la modularité permet de découper le graphe engendré par les exigences de manière naturelle. Ainsi, une communauté non seulement regroupe les exigences qui sont fortement liées par des interdépendances sémantiques, mais aussi des relations conceptuelles et des relations transversales. De manière générale, cela ouvre des portes à tous les types de relations imaginables entre des exigences.

Enfin, contrairement à toutes les propositions qui cherchent à fouiller un gros volume d'exigences pour les prioriser ou les tracer ; ici, nous demandons aux experts d'estimer un intervalle [min, max] pour chacun des critères d'aide à la décision associés aux communautés. Les estimations, synthétisées via un tableau de bord de résumés statistiques, doivent faciliter et objectiver la prise de décision du manageur.

Quelles sont les limites et les perspectives ?

Notre méthode de classification des exigences peut être améliorée de plusieurs manières. D'une part, il faut impérativement l'évaluer de façon plus exhaustive pour s'assurer qu'elle n'est pas sur-apprise. Le corpus d'apprentissage étant initialement constitué de phrases, nous ne pouvons pas faire une validation croisée en k-plis, un corpus de test de plus grande dimension est donc nécessaire. Par ailleurs, d'autres algorithmes d'apprentissage supervisés (arbres de décision, réseau de neurones, régression logistique, etc.) peuvent être testés. Nous avons aussi constaté que le nombre limité de mots dans une exigence rend la classification quasi-aléatoire dans certains cas. Par exemple, quand deux mots clés « *computer* » et « *failure* » sont des indicateurs de deux catégories différentes « CS » et « RAMS », il y a approximativement 50 % de chance que l'exigence appartienne à l'une ou l'autre. Pour outrepasser cette difficulté, on peut rajouter des caractéristiques qui correspondent au contexte de l'exigence à catégoriser. On pourrait par exemple ajouter les lemmes clés des exigences qui précédent et suivent, et ceux du titre du chapitre auquel l'exigence appartient. En effet, dans un document, les exigences sont généralement organisées par sujets, il semble donc pertinent d'exploiter le contexte grâce à des modèles de représentation de textes adaptés tels que GloVE [Pennington et al., 2014], Paragraph Vector [Le and Mikolov, 2014] ou Word2Vec [Mikolov et al., 2013a,b]. On pourrait aussi choisir de développer une fonction de classification multi-annotations (*multi-labels*, en anglais) qui annote une exigence avec zéro, une ou plusieurs catégories à la fois, cependant cela peut biaiser les estimations si les exigences sont dupliquées dans plusieurs communautés. Enfin, une autre alternative de solution est de calculer un taux de confiance

pour chaque prévision. Ainsi, avec une approche hybride automatique-manuelle, toutes les exigences dont le taux de confiance est faible seraient mises à l'écart pour une revue manuelle. Après la revue manuelle, elles seraient greffées au corpus d'apprentissage pour améliorer l'approximation de la fonction de classification.

Le partitionnement des exigences avec Linlog a plusieurs limites. Premièrement, on constate que de nombreuses exigences sont présentes dans plusieurs partitions à la fois ce qui peut induire une surestimation des communautés. De plus, la partition « *other topics* » qui regroupe toutes les exigences non catégorisées a tendance à être beaucoup trop importante pour être estimée par les experts. De manière plus générale, l'algorithme n'est pas adapté a des énoncés courts.

Le graphe engendré par les exigences, lui, doit être le plus riche possible. Ainsi, il serait utile d'exploiter d'autres sources de connaissances. Par exemple, la base de données DBpedia. Néanmoins, l'identification des arêtes du graphe est actuellement trop gourmande en temps de calcul, il est donc indispensable d'explorer différentes pistes pour remédier à cette limite. La première c'est de chercher des heuristiques en étudiant différentes alternatives : doit-on vérifier toutes les paires d'un lemme clé dans ConceptNet 5 ou peut-on se limiter au 10 ou 5 premières ? Doit-on vérifier les relations pour tous les lemmes clés ou seulement les noms ? Doit on vérifier de manière symétrique ou peut-on le faire qu'une seule fois, et ce, bien que WordNet et ConceptNet 5 ne soient pas symétriques (A est un synonyme de B, mais B n'est pas forcément un synonyme de A) ? Doit-on utiliser WordNet et ConceptNet ou peut-on se contenter de ConceptNet ? Pour répondre à ces questions il est indispensable de réaliser diverses expériences. Enfin, la seconde technique pour réduire le temps de recherche des relations entre exigences est de distribuer les calculs à l'aide d'une plate-forme telle que Apache Hadoop ou Apache Spak. La conjonction de ces deux propositions devrait permettre de réduire significativement le temps de calcul et, par conséquent, de réaliser des tests avec des données industrielles.

En plus d'être utile au partitionnement du graphe engendré par les exigences, les relations linguistiques et conceptuelles peuvent servir à faire de l'expansion de requêtes et à créer des facettes. En effet, la base de données Neo4J embarque le moteur de recherche Apache Lucene. Ce dernier permet d'indexer des propriétés de nœuds ou de relations. Dans notre prototype, nous avons par exemple indexé la propriété correspondant à l'énoncé d'un nœud « Exigence ». Nous pouvons alors retrouver des exigences en faisant des requêtes *full text*. Il serait intéressant de créer une visualisation qui synthétise les résultats d'une requête étendue ou non sur demande via l'utilisation de filtres généralement appelées facettes. Par exemple, si il y a une relation conceptuelle entre *airplane* et *pilot* alors une recherche libre du terme *airplane* suggèrerait une facette *pilot*.

Troisième partie

VALIDATION & CONCLUSION

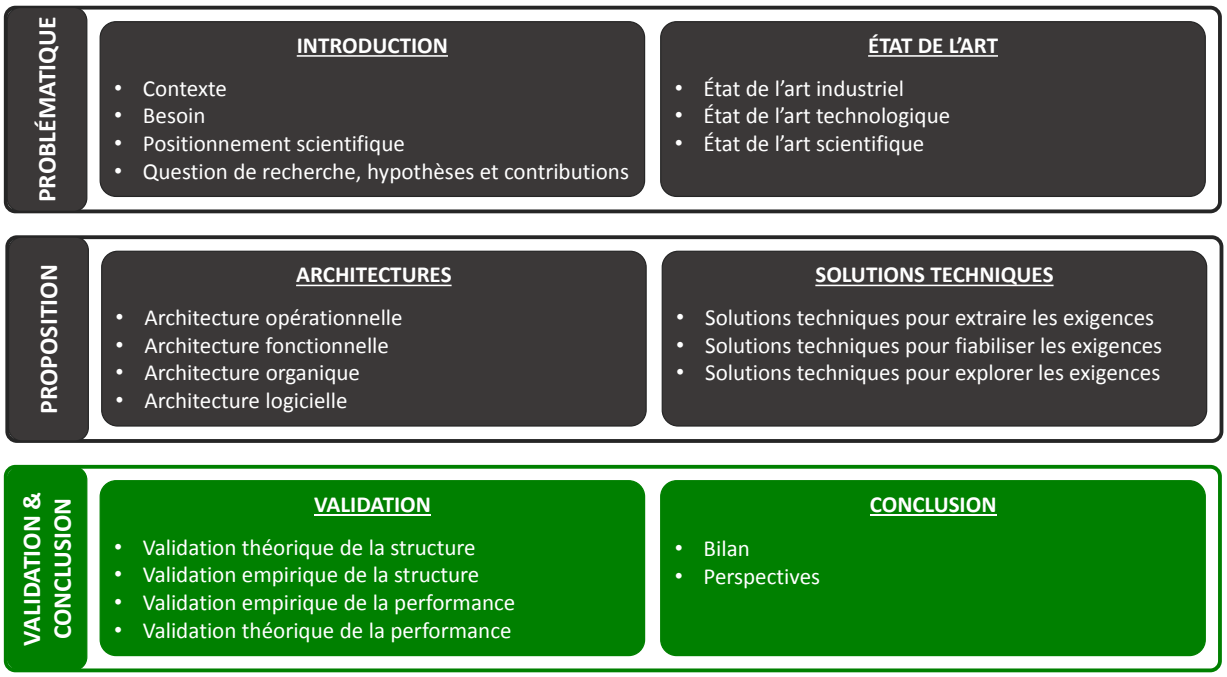


FIGURE 6.29 – 3ème partie dédiée à la validation et à la conclusion des travaux de recherche.

Dans cette dernière partie III du manuscrit, nous validons la méthode outillée que nous proposons et concluons sur ces travaux de recherche.

La validation de la méthode outillée que nous proposons est présentée dans le chapitre 7. Cette validation repose sur le « carré de validation » proposé par Seepersad et al. [2006] et dont le but est de valider une méthode d'ingénierie en quatre temps : validation de la structure *vs.* validation de la performance, puis validation théorique *vs.* validation empirique.

Le chapitre 8 de cette partie III du manuscrit est une conclusion constituée d'un bilan général qui synthétise les travaux, ainsi que d'une énumération de perspectives à court, moyen et long termes.

Chapitre 7

VALIDATION

Toute connaissance scientifique ou technologique doit faire l'objet d'une validation à l'aide d'un protocole approprié au risque de tomber dans la rhétorique. Maintenant que nous avons exposé notre méthode, nous souhaitons fournir quelques arguments qui valident que celle-ci répond bien au besoin de synthèse des exigences.

Selon Popper [1934], pour qu'une théorie soit scientifique elle doit être réfutable, c'est-à-dire qu'il est possible de la contredire au sens logique. Dans le cas contraire on parle de pseudo-science. Les théories qui appartiennent aux sciences fondamentales ont des protocoles de validation quasi-universels. En mathématiques, on utilise la démonstration, tandis qu'en physique on réalise des expériences. Par exemple, aujourd'hui, on entend beaucoup parler de la validation de la théorie de la relativité générale dont l'une des conséquences est l'existence d'ondes gravitationnelles. Il aura fallu un siècle et des moyens considérables (Ligo, Virgo, ...) pour les détecter. Selon le vocabulaire de Popper [1934], on dit que les détections des ondes gravitationnelles corroborent la théorie de la relativité générale. La corroboration marque le fait qu'au contraire des pseudo-connaissances qui sont des actes de foi, une connaissance scientifique est approximativement vraie et peut par principe être mise en défaut, on dit alors qu'elle est falsifiable.

Bien que l'ingénierie puisse s'appuyer sur des connaissances scientifiques, elle n'appartient pas aux sciences mais au domaine de l'action technique. Le critère de validation n'est alors pas d'être le plus vrai possible, mais d'être le plus efficace possible. Le plus efficace ne coïncide pas nécessairement avec le plus vrai. Ce qui importe donc pour une méthode d'ingénierie c'est d'être plus efficace que ses concurrentes.

En ingénierie, la validation de la proposition est très, voire trop souvent négligée. En effet, de nombreuses thèses ou publications proposent, par exemple, le énième modèle de données d'un produit, ou la énième méthode de conception, sans véritablement montrer qu'il ou qu'elle est plus efficace que les méthodes existantes. Les personnes à l'origine de ces contributions se défendent alors au moyen de divers arguments pour justifier l'impossibilité de valider leurs propositions. Parmi ces arguments on retrouve : pour valider il me faut un ou plusieurs contextes d'applications (entreprise, produit, projet, personnel, etc.) ; pour valider il me faut un prototype opérationnel ; pour valider il me faut des projets collaboratifs et pluridisciplinaires qui durent plusieurs mois ou années, etc.

Certes, les méthodes d'ingénierie ont une dimension subjective pour les raisons invoquées précédemment, mais aussi parce qu'elles sont appliquées par des personnes dont l'habilité et les connaissances ont un effet sur la qualité des résultats. Par ailleurs, les questions de recherche toujours plus ambitieuses et un état de l'art qui ne cesse d'accroître l'immensité de son propre non-savoir rentrent en conflit avec l'impatience de nos modes de recherche. Cependant, pour éviter de tomber dans le domaine du boniment, on peut objectiver les choses en construisant des essais conceptuels et expérimentaux permettant de comparer des approches méthodologiques. On ne peut pas dire que ces essais soient totalement dépourvus de biais subjectifs, mais c'est mieux que de s'adonner aux incantations magiques et invérifiables.

Le problème de la validation en ingénierie a été récemment mis en avant par quelques auteurs [See-

persad et al., 2006, Vermaas, 2014]. Pour valider notre proposition, nous utilisons le carré de validation (*validation square*, en anglais) proposé par Seepersad et al. [2006]. Selon les auteurs, valider une méthode c'est établir un processus de confiance dans son utilité pour un objectif donné. Ici, notre objectif est de synthétiser plusieurs centaines ou milliers d'exigences.

L'utilité d'une méthode se divise en deux critères. Le premier critère est la validité (*effectiveness*, en anglais) : est-ce que les résultats sont corrects ? Le deuxième critère est l'efficience (*efficiency*, en anglais) : est-ce que les résultats sont obtenus avec des performances acceptables (moins de temps et moins de dépenses) ? Pour valider une méthode, la matrice de validation préconise alors de diviser l'exercice en quatre : (1) la validation théorique de la structure, (2) la validation théorique de la performance, (3) la validation empirique de la structure, (4) la validation empirique de la performance.

7.1 Validation théorique de la structure

L'objectif de la validation théorique structurelle est de valider la consistance logique de la méthode proposée selon deux niveaux : celui de la méthode elle même, mais aussi celui des connaissances pré-existantes réutilisées.

La première étape c'est de définir explicitement le domaine d'applicabilité de notre méthode. Ci-dessous nous précisons à quel type de mission elle est destinée et à quelles exigences de haut niveau elle satisfait :

- Mission - La méthode doit permettre de synthétiser des centaines ou milliers d'exigences.
  - Extraire les exigences.
    - Extraire les exigences de documents prescriptifs non structurés.
    - Extraire les exigences de documents prescriptifs semi-structurés.
    - Identifier les exigences contenant des références transversales.
  - Fiabiliser les exigences
    - Identifier les défauts intrinsèques.
    - Identifier les défauts relationnels.
    - Notifier et supprimer les défauts intrinsèques.
    - Notifier et supprimer les défauts relationnels.
  - Explorer les exigences
    - Segmenter les exigences.
    - Définir et estimer des critères d'aide à la décision.
    - Prospecter une synthèse des estimations.

La deuxième étape consiste à faire un état de l'art des solutions pré-existants vis-à-vis des exigences. Cet état de l'art a été réalisé dans les sections 4.2, 5.2 et 6.2. Les limites des solutions existantes que notre méthode se veut de repousser y sont également détaillées et ont motivé la poursuite de nos recherches.

Enfin, pour valider la consistance logique de notre méthode nous utilisons un diagramme de flux d'informations [1] (Fig. 7.1) qui démontre que les entrées d'une étape sont en adéquation avec les sorties de l'étape précédente. En plus de ce diagramme de flux, notre prototype logiciel intégré matérialise la consistance logique de notre méthode. Dans ce diagramme on peut lire que la méthode résulte d'un raisonnement qui lui assure une certaine consistance logique. On peut donc affirmer que la structure théorique de notre méthode est validée pour le domaine d'applicabilité défini.

1. Les deux fonctions « Classifier les exigences en métiers » et » « Partitionner les exigences » qui correspondent aux deux pistes d'exploration des exigences que nous avons délaissées pour préférer le découpage de graphe sont, pour le moment, mises à l'écart dans notre méthode, mais cela illustre le parcours sinueux et obscure de la recherche d'une solution.

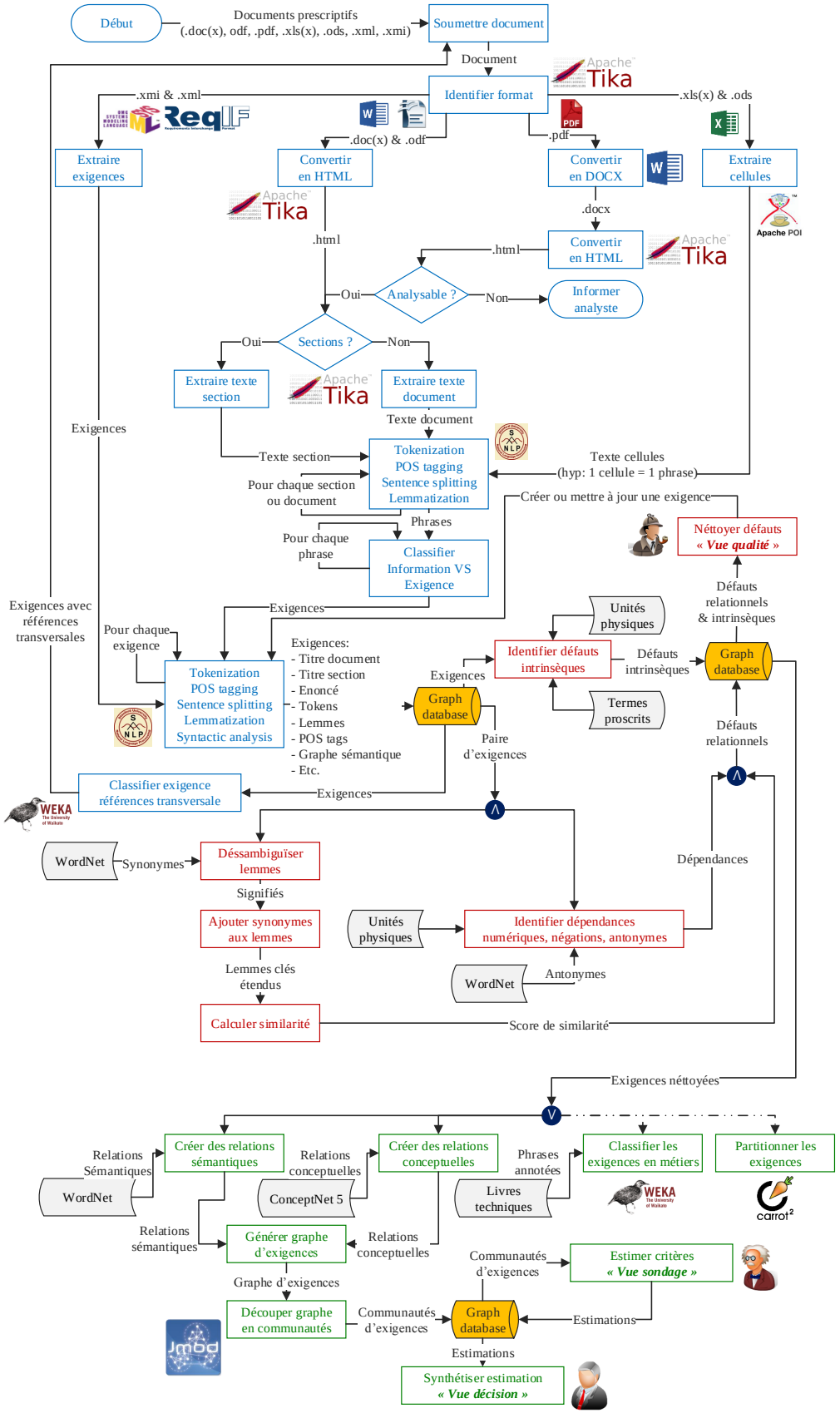


FIGURE 7.1 – Diagramme de flux d'informations qui résume la méthode outillée de synthèse des exigences.

7.2 Validation empirique de la structure

La validation empirique de la structure consiste à établir une confiance vis-à-vis des exemples que nous avons utilisés pour vérifier les performances de notre méthode.

Dans un premier temps nous comparons les expérimentations que nous avons menées aux exigences définies précédemment (Fig. 7.1). Toutes les opérations analytiques, c'est-à-dire de traitement (extraction, identification défauts, segmentation), ont été expérimentées et nous rappelons leurs limites dans chacune des sections 4.4, 5.4 et 6.4. Cela ne veut pas dire que notre méthode fonctionne dans tous les cas, mais sur certains problèmes avec des données qui sont représentatives des pratiques industrielles, sauf pour le découpage de graphe que nous avons testé sur un cas simplifié pour des raisons techniques. Par ailleurs, toutes les données utilisées dans nos tests sont accessibles. Nos résultats et nos données forment alors une base initiale pour les autres chercheurs de la communauté qui souhaiteraient comparer leurs résultats avec les nôtres.

Parmi les fonctions techniques qui ne sont pas validées par des expérimentations on retrouve celles qui nécessitent un contexte opérationnel : des analystes pour supprimer les défauts intrinsèques et relationnels, des experts pour estimer les critères d'aide à la décision, ainsi qu'un manageur pour prendre des décisions informées. Pour réaliser de tels tests nous avons besoin d'un véritable scénario opérationnel.

	Ex 4.4	Ex 5.4	Ex 6.4
Extraire les exigences			
Extraire les exigences de documents non structurés	✓		
Extraire les exigences de documents semi-structurés	✓		
Identifier les exigences contenant des références transversales.	✓		
Fiabiliser les exigences			
Identifier les défauts intrinsèques		✓	
Identifier les défauts relationnels		✓	
Notifier et supprimer les défauts intrinsèques		✗	
Notifier et supprimer les défauts relationnels		✗	
Explorer les exigences			
Segmenter les exigences			✓
Définir et estimer des critères d'aide à la décision			✗
Prospecter une synthèse des estimations			✗

TABLE 7.1 – Table de contingence pour le premier document prescriptif.

Pour être objectif, nous devons expliciter les hypothèses simplificatrices sous-jacentes aux exemples utilisés pour tester notre méthode. En effet, la confiance qu'ont peut avoir pour les résultats obtenus est étroitement liée à la qualité des exemples utilisés. Plus les exemples se rapprochent des cas que l'on rencontre dans un contexte opérationnel, meilleure est la confiance.

Ici, toutes les données sont des documents prescriptifs, en particulier des spécifications issues de l'industrie, qui prescrivent entre une centaine et un millier d'exigences. Cette volumétrie de données est représentative de notre problématique qui vise à synthétiser des centaines ou milliers d'exigences. De plus, nous travaillons dans le cas le plus défavorable pour lequel les documents prescriptifs sont non structurés et les exigences n'ont pas été rédigées selon des recommandations ou des contraintes spécifiques limitant la diversité du langage naturel.

L'une des hypothèse simplificatrices c'est que nous avons utilisé moins de 10 documents différents dans nos expérimentations et que la forme du contenu varie beaucoup d'un document à l'autre. Par exemple, nous avons fait l'hypothèse que ces documents doivent être rédigés avec l'éditeur de texte Word, OpenOffice ou Excel, et ce, même pour le format natif des fichiers PDF, ce qui assure l'analyse de quelques éléments de structure dans le contenu textuel (énumérations, titres, tableaux, en-têtes et pieds de pages, etc.). La seconde hypothèse simplificatrice c'est que nous travaillons uniquement avec

des documents rédigés en anglais.

Sous-réserve de quelques tests supplémentaires pour évaluer la sensibilité de notre méthode, nous pouvons conclure que sa structure est empiriquement vérifiée grâce aux résultats obtenus à partir d'expériences menées sur des données représentatives des pratiques industrielles.

7.3 Validation empirique de la performance

La validation empirique de la performance consiste à s'assurer que la méthode permet d'économiser du temps ou de l'argent sur les expérimentations réalisées précédemment dans la validation empirique de la structure.

Ceci est difficilement quantifiable, il faudrait faire une synthèse des exigences sans l'outil et avec l'outil. Cela reviendrait à réaliser tout le processus manuellement : lire les documents prescriptifs pour identifier les exigences ; identifier et supprimer les défauts de qualité ; diviser les exigences en communautés ; estimer les communautés ; et analyser les estimations en vue d'une prise de décision.

Nous n'avons pas réalisé un tel scénario pour valider le gain de performance avec notre méthode outillée. Cependant, dans nos expériences, nous avons manuellement vérifié les résultats retournés par le prototype. Par exemple, pour extraire les 1618 exigences des deux documents non structurés il nous a fallu approximativement deux jours de lecture assidue. Le prototype, lui, met moins de deux minutes.

Il faut presque autant de temps pour identifier les exigences qui font référence à un document externe, car cela nécessite de lire les exigences une par une. Là aussi, le prototype met moins d'une minute. On ne peut pas négliger le temps nécessaire à la construction des corpus pour apprendre et tester la fonction de classification automatique. Néanmoins, bien que cela nous ait aussi pris plusieurs jours, l'investissement est ponctuel, tandis que la revue des documents prescriptifs est systématique à chaque appel d'offre.

Ce constat de gain de temps est aussi valable pour la classification des exigences au sein de catégories métiers. Pour classifier 289 exigences il faut approximativement entre deux et trois heures de travail manuel, tandis que le prototype a besoin moins d'une minute pour un taux de réussite qui avoisine les 74 %.

La seule tâche qui est coûteuse en temps c'est la construction du graphe d'exigences à partir du thésaurus WordNet et de l'ontologie ConceptNet, mais nous sommes convaincus que ce problème peut être résolu avec des moyens d'implémentations adaptés tels que le calcul distribué et l'adoption d'heuristiques.

La détection de contradiction et de redondance nécessiterait de faire $\frac{n*(n-1)}{2}$ comparaisons pour n exigences. Quand n est grand c'est tout simplement infaisable. Pour 1073 exigences, soit 575 128 comparaisons, le prototype a besoin d'environ dix minutes pour identifier les conflits potentiels inter-exigences.

Ces revues manuelles des documents ont conforté notre idée que c'est un travail terriblement fastidieux et peu attractif. Une véritable contrainte pour les entreprises qui justifie le besoin d'une solution technologique avec des représentations plus synthétiques et esthétiques. Toutes les opérations sont porteuses d'un gain de temps qui est aussi évident que considérable, nous concluons donc que la performance de notre méthode est empiriquement validée.

7.4 Validation théorique de la performance

Alors que la validation empirique de la performance se limite aux exemples utilisés dans les expérimentations, la validation théorique de la performance, elle, consiste à s'assurer que la méthode va permettre d'économiser du temps ou de l'argent dans toutes les situations analogue à la notre. Autrement dit, dans n'importe quel problème régit par les exigences définies précédemment, notre méthode doit permettre de gagner en performance.

L'extraction, la fiabilisation et l'exploration des exigences ne se limitent pas à notre mission de synthèse. Par exemple, pour prioriser les exigences il est indispensable de les extraire et de fiabiliser leur interprétation, voire de les segmenter s'il y en a trop. Segmenter les exigences sert aussi à mieux les tracer. Par exemple, nous savons que la spécification d'un sous-marin nucléaire avoisine le million d'exigences, donc quand les auteurs recommandent de tracer des exigences singulières tout au long du cycle de vie, c'est une utopie ! Il est indispensable de les gérer en configuration à un niveau d'abstraction plus élevé ; des communautés par exemple. La détection des exigences qui font référence à des documents externes potentiellement prescriptifs et applicables facilite aussi la traçabilité, en particulier l'analyse d'impact des modifications.

En plus d'être des solutions techniques qui aident à faire une synthèse des exigences, les éléments constitutifs de notre méthode sont avant tout des briques théoriques et technologiques qui sont applicables, avec ou sans modifications, à diverses problématiques en ingénierie des exigences. L'outillage des tâches manuelles telles que l'extraction, le tri, la priorisation, etc. est dans tous les cas un gain de temps considérable quand on observe l'état actuel des pratiques industrielles. On peut donc conclure que la performance de notre méthode est théoriquement validée.

Chapitre 8

CONCLUSION

Pour conclure ce manuscrit de thèse, nous faisons un bilan qui compare ce que nous cherchions il y a trois ans avec ce que nous savons aujourd'hui. Enfin, pour terminer, nous suggérons un ensemble de perspectives à explorer à court, moyen et long termes.

8.1 Bilan

Quelle était la problématique ?

Pour de nombreuses raisons telles que la complexification des produits et des processus, la personnalisation de masse et l'entreprise étendue, le nombre d'exigences nécessaires pour spécifier un produit tend à croître de manière exponentiel à tel point qu'on peut les considérer comme des « Big Data ». Ainsi, dans une relation client-fournisseur, le fournisseur – en particulier le sous-traitant qui se situe en bas de la chaîne logistique – est incapable de prendre des décisions stratégiques informées en phase amont du cycle de vie. Par exemple, une décision de « Go *vs.* No Go » lors de la réponse à un appel d'offre.

Quel était l'objectif ?

Pour résoudre ce problème, nous avons cherché une méthode qui puisse répondre à la question : comment supporter les entreprises, en particulier les fournisseurs jouant le rôle de sous-traitants, à faire la synthèse d'un corpus de documents prescriptifs rédigés en anglais, lesquels spécifient des centaines ou milliers d'exigences que le produit doit satisfaire ?

Quelle était l'hypothèse ?

Pour atteindre notre objectif, nous considérons que le problème est un exercice de fouille de données textuelles qu'on peut résoudre au moyen des techniques issues des sciences des données (traitement du langage naturel, classification et partitionnement de textes, recherche et visualisation d'informations, etc.). Ainsi, nous supposons qu'un environnement numérique constitué de briques théoriques et technologiques issues des sciences des données permet de supporter les entreprises, en particulier les sous-traitants, à faire la synthèse de centaines ou milliers d'exigences.

Quelle est la méthode proposée ?

Pour faire la synthèse d'un gros volume de données nous proposons de fournir trois services : (1) extraire les exigences de documents prescriptifs ; (2) fiabiliser l'interprétation des exigences ; et (3) explorer les exigences.

L'extraction des exigences concerne les documents semi-structurés. Par exemple, ceux qui sont conformes à la norme d'interopérabilité ReqIF ou les diagrammes d'exigences SysML. Celles-ci peuvent

être extraites avec un parseur XML. Pour les documents non structurés tels que les fichiers Word, OpenOffice, Excel ou PDF nous proposons un processus de traitement du langage naturel qui identifie les phrases avant de les classifier sur la base de termes prescriptifs. Enfin, nous avons appris une fonction de classification, laquelle est estimée à partir d'un algorithme d'apprentissage machine, qui permet d'identifier les exigences faisant référence à un document externe qui, lui aussi, est potentiellement prescriptif et applicable.

La fiabilisation des exigences, elle, consiste à minimiser le champ des interprétations des exigences afin que les estimations réalisées par les experts et les décisions prises par le manageur soient respectivement les plus objectives et rationnelles possibles. Nous utilisons un dictionnaire de termes proscrits pour détecter les mots vagues et les connecteurs logiques qui rendent les exigences ambiguës. Les exigences incomplètes qui n'ont pas de domaine de tolérance mesurable sont aussi détectées par des règles que nous appliquons au graphe sémantique de chaque exigence. En plus de ces trois défauts intrinsèques, nous utilisons une fonction de similarité pour identifier les exigences redondantes et contradictoires. Cette fonction de similarité n'est pas une intersection brute entre les mots clés de chaque exigence, mais considère les synonymes de chaque mot clé. Enfin, des représentations graphiques et interactives permettent à l'analyste d'observer les défauts et de les corriger ou de les exclure.

Enfin, la troisième partie de la méthode permet à un manageur de découper le graphe engendré par les exigences en communautés. Chaque communauté est alors caractérisée par un ensemble de critères d'aide à la décision dont l'estimation est assurée par les différentes fonctions de l'entreprise. Une synthèse des estimations est restituée au manageur grâce à des résumés statistiques graphiques qui lui permettent de prospecter l'espace prescriptif en vue de prendre des décisions stratégiques avisées.

Quels sont les résultats obtenus ?

Les résultats obtenus sont pour la plupart réalisés avec des données industrielles représentatives de la volumétrie que l'on rencontre en pratique, soit plusieurs centaines ou milliers d'exigences non structurées.

Pour tester notre méthode d'extraction des exigences nous avons utilisé deux documents prescriptifs non structurés contenant 194 et 1066 exigences. Dans le premier cas le rappel est de 1 tandis que la précision est de 0.73. Pour le second document, le rappel est de 0.90 et la précision est de 0.99. En moyenne, le rappel est donc de 0.95, tandis que la précision est de 0.86. Concernant l'identification des exigences qui font référence à un document externe, la fonction de classification apprise avec l'algorithme Support Vector Machine a obtenu un taux de classifications correctes de 91.2 % pour une validation croisée en k-plis avec k = 10 et 500 exemples.

La détection des défauts intrinsèques n'a pas été évaluée car elle se résume à la détection de mots clés à partir d'un dictionnaire. Si le terme est dans le dictionnaire, alors il est détecté. S'il n'y est pas, alors il suffit de l'ajouter. La détection des contradictions et des redondances, elle, a été testée sur un corpus constitué de trois documents prescrivant un total de 1618 exigences. Parmi ces 1618 exigences, le prototype a identifié 96 défauts relationnels dont 90 redondances et 6 contradictions. Une revue manuelle des 96 paires d'exigences retournées nous a permis de confirmer qu'elles sont toutes potentiellement conflictuelles.

Concernant le choix de la méthode la plus appropriée pour segmenter un gros volume des exigences, nous avons obtenu un taux de classifications correctes de 76 % avec une précision de 0.78 et un rappel de 0.76. L'application de l'algorithme LinLog à un corpus de 1618 exigences a permis d'obtenir 109 communautés d'exigences. Avec une moyenne approximative de 15 exigences par communauté, les résultats semblent encourageants, mais ce n'est pas le cas car une communauté regroupe 436 exigences qui n'ont pas pu être catégorisées. Par ailleurs, nous avons opposé cette technique de partitionnement sémantique à la méthode de découpage de graphe sur un cas simplifié contenant une dizaine d'exigences. La technique basée sur l'analyse du graphe d'exigences a permis de détecter les deux communautés espérées, tandis que LinLog a identifié douze communautés.

Quelles sont les limites ?

Étant donné l'étendue du domaine d'études, chacune des contributions techniques est sujette à quelques limites plus ou moins difficiles à repousser.

Pour l'extraction des exigences, les limites principales sont liées au format des documents. Si le format ne permet pas d'identifier les éléments de structure (listes, titres, tableaux, etc.), alors on accumule du bruit dans les exigences. Il est par ailleurs nécessaire de définir une mesure qui quantifie le degré de confiance associé à une exigence afin de mettre en avant les énoncés classifiés comme étant des exigences mais qui ne le sont probablement pas. Enfin, quand le document contient des exigences multi-représentées (tableau, croquis, graphique, etc.), notre méthode se limite à leur dimension textuelle.

Pour la fiabilisation des exigences, la première limite c'est d'être capable d'identifier les autres types (*factive, structural, lexical, world knowledge*) de contradictions définis par de Marneffe et al. [2008]. Par ailleurs, la qualité des exigences dans les documents prescriptifs industriels est déplorable. Ainsi, quand on réalise des diagnostics de qualité automatiques on obtient une quantité faramineuse de défauts, et ce, au point qu'on puisse se demander si une telle fonctionnalité est utile car il faut un temps considérable pour les corriger. Il faut réfléchir à une autre alternative de solution pour minimiser l'ambiguïté des exigences. Nous avons déjà initié une alternative de solution en proposant d'exploiter le contexte (sémantique, conceptuel, transversal, document, etc.) d'une exigence pour réduire son champ d'interprétations.

La principale limite concernant notre méthode d'exploration des exigences c'est la construction du graphe d'exigences que l'on propose de segmenter pour travailler avec des communautés. En effet, les nombreuses comparaisons par paires requièrent un temps de calcul impraticable pour des données réalistes.

8.2 Perspectives

Pour outrepasser les limites discutées précédemment, nous suggérons diverses perspectives à court, moyen et long terme. Les perspectives à court terme ne nécessitent pas une réflexion particulière mais un effort de développement nécessaire pour consolider ces travaux. Les perspectives à moyen terme, elles, nécessitent de tester différentes alternatives afin de choisir celle dont les performances seront les plus satisfaisantes. Enfin, les perspectives à long terme sont des problèmes pour lesquels nous n'avons pas de recommandations ou seulement des suggestions qui sont purement spéculatives. Ce sont des problématiques pour de nouveaux projets de recherche.

Les perspectives à court terme

La première perspective que nous recommandons dans l'immédiat c'est d'enrichir les corpus de tests pour toujours mieux évaluer les risques de sur-apprentissage des fonctions de classification automatique. Cela passe par la collection de documents prescriptifs industriels de sources variées. Une fois enrichis, les corpus devront servir à de nouveaux tests pour, d'une part, confirmer les résultats préliminaires obtenus et, d'autre part, étudier d'autres configurations d'apprentissage en faisant varier le nombre de caractéristiques discriminantes sélectionnées, la manière dont elles sont sélectionnées (χ^2, information mutuelle, algorithme glouton, etc.), et le type d'algorithme d'apprentissage (régression logistique, réseau de neurones, arbres de décisions, etc.).

La deuxième perspective indispensable c'est de réviser l'architecture du prototype pour l'adapter à une plate-forme de calcul distribué telle que la technologie Apache Spark. Cela nécessite de trouver une solution pour accoster la base de données orientée graphe Neo4J et l'algorithme de calcul distribué Map/Reduce. Une autre alternative serait de remplacer Neo4J par des technologies telles que Apache Giraph dédiées au traitement de gros graphes qui sont utilisées dans l'analyse du Web par Google ou de réseaux sociaux par Facebook. Adopter ces technologies spécifiques au traitement de gros volumes de données garantira le passage à l'échelle vers des dizaines de milliers d'exigences.

Enfin, la troisième perspective à court terme est de tester les nouveaux modèles de représentions de textes tels que GloVe [Pennington et al., 2014], Paragraph Vector [Le and Mikolov, 2014] et Word2Vec [Mikolov et al., 2013a,b] pour améliorer la fonction de similarité et, par conséquent, la détection des redondances et contradictions entre les exigences.

Les perspectives à moyen terme

Une fois que les perspectives à court terme ont été implémentées et validées, il est indispensable de trouver un scénario opérationnel dans l'industrie pour poursuivre la phase de validation de la méthode outillée. Trouver un ou plusieurs partenaires industriels c'est aussi avoir un accès quasi-illimité à leurs patrimoines informationnels. Pour les techniques statistiques mises en œuvre dans notre proposition, plus le volume de données est important, mieux c'est. Les résultats se raffinent en fonction des spécificités du vocabulaire manipulé par chaque entreprise.

Par ailleurs, nous avons souligné le fait que notre solution d'extraction des exigences ne considère que la dimension textuelle. Si une exigence est complétée par une image, un tableau de valeurs, un croquis, un graphique, etc., alors l'exigence est incomplète. Cela peut biaiser les estimations qui sont réalisées *a-posteriori*. Les images sont facilement identifiables, mais il est parfois difficile d'identifier à quelle exigence elle réfère. Une piste est d'utiliser les techniques et les outils de traitement et d'annotation d'images pour collecter tous les artefacts graphiques que l'on peut lier à l'exigence textuelle qui la précède dans le texte. Par exemple, si l'analyste prescrit une évolution de température, il aura tendance à ajouter une courbe juste après avoir spécifié l'énoncé prescriptif.

Une autre perspective à moyen terme est d'exploiter les capacités de la librairie Stanford CoreNLP à traiter le langage naturel français. En effet, certains secteurs d'activité comme le nucléaire travaille encore en français, nous pouvons donc essayer de répliquer notre méthode et de l'adapter aux spécificités de la langue. Parmi les adaptations il faut trouver d'autres sources de connaissances linguistiques et conceptuelles.

Les perspectives à long terme

Enfin, parmi les perspectives à long terme, il y a l'identification des contradictions qui nécessitent une compréhension plus fine du texte. Cette problématique relève plus des recherches fondamentales en compréhension de la langue, mais on peut espérer que la solution viendra avec les futures avancées techniques de l'apprentissage profond et des moteurs d'inférences développés pour le Web sémantique.

Un second projet de recherche qui pourrait être un beau sujet de thèse est la réutilisation des connaissances. Notre méthode permet d'extraire une multitude d'informations telles que les exigences, leurs catégories métiers, leurs affinités sémantiques et conceptuelles, leurs interdépendances transversales, etc. Elle permet aussi de collecter les connaissances tacites des experts qui représentent les fonctions de l'entreprise grâce aux estimations des critères d'aide à la décision. Pour chaque projet, des décisions seront prises par les manageurs en fonction des résumés des estimations. La perspective de recherche est de trouver un moyen théorique et technologique de valoriser toutes ces connaissances et informations à chaque fois qu'un nouveau document prescriptif est soumis à l'environnement numérique, mais aussi à chaque fois qu'un nouveau projet est initié. Par exemple, si les exigences d'un nouveau document, ou d'un nouveau projet, sont pour tout ou partie plus ou moins similaires à celles d'un projet antérieur, il est alors envisageable de réutiliser ces informations et ces connaissances pour obtenir de meilleurs résultats. Par exemple, des estimations plus rapides et plus fiables intégrant un retour sur expériences. La comparaison de graphes peut également permettre d'identifier et de recommander des exigences qui ont été potentiellement oubliées lors de la spécification du nouveau produit.

Enfin, cette structure de modélisation sous forme de graphe ouvre d'autres perspectives, en particulier pour résoudre les problématiques d'analyse d'impacts des modifications. Par exemple, en connaissant les coûts et délais de développement d'une communauté d'exigences, on pourrait faire des simulations pour observer la propagation d'une modification sur les autres exigences. Cela permettrait de mieux appréhender les risques liés aux changements.

Publications

Journaux internationaux

[1] Pinquié, R., Véron, P., Segonds, F., Croué, N. (2016) 'Requirement Mining for Model-Based Product Design', *International Journal of Product Lifecycle Management*, X(X), pp. XX-XX. (in press)

[2] Pinquié, R., Rivest, L., Segonds, F., Véron, P. (2015) 'An illustrated glossary of ambiguous terms used in discrete manufacturing', *International Journal of Product Lifecycle Management*, 8(2), pp. 142-171.

Conférences internationales avec comité de sélection et actes

[1] Pinquié, R., Micouin, P., Véron, P., Segonds, F. (2016) 'Property Model Methodology : a case study with Modelica', *Proceedings of the 11th Tools and Methods of Competitive Engineering (TMCE) international conference*. Aix-en-Provence, France, 9–13 May 2015.

[2] Pinquié, R., Véron, P., Segonds, F., Croué, N. (2015) 'Natural language processing of requirements for model-based product design with Enovia/Catia V6', *Proceedings of 12th Product Lifecycle Management (PLM) international conference*. Doha, Qatar, 19–21 October 2015, pp. 205-215.

[3] Pinquié, R., Véron, P., Segonds, F., Croué, N. (2015) 'A collaborative requirement mining framework to support OEMs', *Proceedings of the 12th Cooperative Design, Visualisation and Engineering (CDVE) international conference.* Mallorca, Spain, 20-23 September 2015, pp 105-114.

Bibliographie

S. Abiteboul. *Sciences des données : de la logique du premier ordre à la Toile*. Collège de France, Fayard, 2012.

S. Abiteboul, I. Manolescu, P. Rigaux, M.-C. Rousset, and P. Senellart. *Web Data Management*. Cambridge University Press, New York, NY, USA, 2011.

J.-R. Abrial. *The B-book : Assigning Programs to Meanings*. Cambridge University Press, New York, NY, USA, 1996.

P. Achimugu, A. Selamat, R. Ibrahim, and M. Mahrin. A systematic literature review of software requirements prioritization research. *Information and software technology*, 56(6) :568–585, 2014a.

P. Achimugu, A. Selamat, and R. Ibrahim. *A Clustering Based Technique for Large Scale Prioritization during Requirements Elicitation*, pages 623–632. Springer International Publishing, Cham, 2014b.

P. Achimugu, A. Selamat, and R. Ibrahim. *A Web-Based Multi-Criteria Decision Making Tool for Software Requirements Prioritization*, pages 444–453. Springer International Publishing, Cham, 2014c.

AFIS. *Guide des bonnes pratiques en ingénierie des exigences*. Cédapués, 2012a.

AFIS. *Découvrir et comprendre l'ingénierie système*. Cédapuès, 2012b.

AFIS. *Guide : bonnes pratiques en expression du besoin*. Cédapuès, 2014.

V. Ahl. *An experimental comparison of five prioritization methods - Investigating ease of use, accuracy and scalability*. Thesis, 2005.

B. Alenljung and A. Persson. *Decision-Making Activities in Requirements Engineering Decision Processes : A Case Study*, pages 707–718. Springer US, Boston, MA, 2006.

E. Andrianarison and J.-D. Piques. Sysml for embedded automotive systems : a practical approach. In *Conference on Embedded Real Time Software and Systems. IEEE*, 2010.

P. R. Anish, B. Balasubramaniam, J. Cleland-Huang, R. Wieringa, M. Daneva, and S. Ghaisas. Identifying architecturally significant functional requirements. In *Proceedings of the Fifth International Workshop on Twin Peaks of Requirements and Architecture*, TwinPeaks '15, pages 3–8, 2015.

F. J. Anscombe. Graphs in statistical analysis. *The American Statistician*, 27(1) :17–21, 1973.

ANSI/EIA 632. Processes for engifneering a system. Standard, American National Standards Institute/Electronic Industries Alliance standard, 2003.

C. Arora, M. Sabetzadeh, L. Briand, F. Zimmer, and R. Gnaga. Rubric : A flexible tool for automated checking of conformance to requirement boilerplates. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013, pages 599–602, 2013b.

M. Asimow. *Introduction to design*. Englewood Cliffs : Prentice-Hall, 1962.

P. Avesani, S. Ferrari, and A. Susi. *Case-Based Ranking for Decision Support Systems*, pages 35–49. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

J. Azar, R. K. Smith, and D. Cordes. Value-oriented requirements prioritization in a small development organization. *IEEE Software*, 24(1) :32–37, January 2007.

Z. Azmeh, I. Mirbel, and L. E. Jiani. A tool to improve requirements review in collaborative software development platforms. In *2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS)*, pages 1–6, May 2014.

Z. Azmeh, I. Mirbel, and P. Crescenzo. *Highlighting Stakeholder Communities to Support Requirements Decision-Making*, pages 190–205. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

B. Bachimont. *Ingénierie des connaissances et des contenus : le numérique entre ontologies et documents*. Science informatique et SHS. Hermès, Paris, 2007.

S. Badreau and J.-L. Boulanger. *Ingénierie des exigences, méthodes et bonnes pratiques pour construire et maintenir un référentiel*. Dunod, 2014.

U. Beck. *La société du risque*. Flammarion, 2008.

A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. Henzinger, and K. G. Larsen. Contracts for System Design. Research Report RR-8147, INRIA, November 2012.

A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. Henzinger, and K. Larsen. Contracts for Systems Design : Methodology and Application cases. Research Report RR-8760, Inria Rennes Bretagne Atlantique ; INRIA, July 2015a.

A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. Henzinger, and K. Larsen. Contracts for Systems Design : Theory. Research Report RR-8759, Inria Rennes Bretagne Atlantique ; INRIA, July 2015b.

P. Berander and A. Andrews. *Requirements Prioritization*, pages 69–94. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

A. Bernard, E. Coatanea, F. Christophe, and F. Laroche. Design : A key stage of product lifecycle. *Procedia CIRP*, 21 :3–9, 2014. 24th {CIRP} Design Conference.

Y. Bernard. Requirements management within a full model-based engineering approach. *Systems Engineering*, 15(2) :119–139, 2012.

D. M. Berry, P. D. C. Science, M. M. Krieger, and P. D. Mathematics. From contract drafting to software specification : Linguistic sources of ambiguity - a handbook version 1.0, 2003.

G. Berry. *Pourquoi et comment le monde devient numérique*. Collège de France, 2008.

C. Bishop. *Pattern recognition and machine learning*. Springer, 2007.

B. Boehm and H. Kitapci. *The WinWin Approach : Using a Requirements Negotiation Tool for Rationale Capture and Use*, pages 173–190. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

S. Boschert and R. Rosen. *Digital Twin—The Simulation Aspect*, pages 59–74. Springer International Publishing, 2016.

R. Brath and D. Jonker. *Graph analysis and visualization : discovering business opportunity in linked data*. John Wiley & Sons, 2015.

T. D. Breaux and A. I. Antón. Mining rule semantics to understand legislative compliance. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, WPES '05, pages 51–54, 2005.

P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos : An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3) :203–236, 2004.

N. Carlson and P. Laplante. The nasa automated requirements measurement tool : a reconstruction. *Innovations in Systems and Software Engineering*, 10(2) :77–91, 2013.

A. Casamayor, D. Godoy, and M. Campo. Identification of non-functional requirements in textual specifications : A semi-supervised learning approach. *Information and Software Technology*, 52(4) : 436–445, 2010.

A. Casamayor, D. Godoy, and M. Campo. Functional grouping of natural language requirements for assistance in architectural software design. *Knowledge-Based Systems*, 30 :78–86, 2012.

C. Castro-Herrera, C. Duan, J. Cleland-Huang, and B. Mobasher. Using data mining and recommender systems to facilitate large-scale, open, and inclusive requirements elicitation processes. In *2008 16th IEEE International Requirements Engineering Conference*, pages 165–168, September 2008.

P. Chatzipetrou, L. Angelis, P. Rovegard, and C. Wohlin. Prioritization of issues and requirements by cumulative voting : A compositional data analysis framework. In *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 361–370, September 2010.

D. Chen and C. Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October 2014. Association for Computational Linguistics.

J. Cheung, J. Scanlan, J. Wong, J. Forrester, and H. Eres. Application of valued-driven design to commercial aero-engine systems, 13-15 September 2010 2010.

L. Cholvy and C. Garion. Exigences, réglementations et contraintes. In *Journées Nationales sur les Modèles de Raisonnement (JNMR)*, Paris, France, 27-28 November 2003.

N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2 :113–124, 1956.

F. Christophe, M. Wang, E. Coatanea, Y. Zeng, and A. Bernard. Grammatical and semantic disambiguation of requirements at elicitation and representation stage. In *Proceedings of the ASME internation design engineering technical conferences and computers and information in engineering conference*, pages 17–31, 2011.

L. Chung and J. C. S. do Prado Leite. *On Non-Functional Requirements in Software Engineering*, pages 363–379. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

J. Cleland-Huang, R. Settimi, C. Duan, and X. Zou. Utilizing supporting evidence to improve dynamic requirements traceability. In *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, pages 135–144, August 2005.

J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc. The detection and classification of non-functional requirements with application to early aspects. In *Requirements Engineering, 14th IEEE International Conference*, pages 39–48, September 2006.

CMMI Product Team. CMMI for Development, Version 1.3. Technical Report CMU/SEI-2010-TR-033, Software Engineering Institute, 2010.

E. Coatanéa, F. Mokammel, and F. Christophe. Requirements model for engineering, procurement and interoperability : a graph and power laws vision of requirements engineering. Technical report, Aalto university, 2013.

L. Cohen. *Quality Function Deployment : How to make QFD work for you*. Prentice Hall, 1995.

P. Collopy. Adverse impact of extensive attribute requirements on the design of complex systems. In *7th AIAA Aviation Technology, Integration and Operations (ATIO) Conference*, Belfast, Northern Ireland, 2007.

N. Cross. *Engineering Design Methods : Strategies for Product Design (4th ed.)*. John Wiley & Sons, 2008.

A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Sci. Comput. Program.*, 20(1-2) :3–50, April 1993.

P. David. *Understanding digital technology's evolution and the path of measured productivity growth : present and future in the mirror of the past*. MIT Press, 2000.

A. Davis, S. Overmyer, K. Jordan, J. Caruso, F. Dandashi, A. Dinh, G. Kincaid, G. Ledeboer, P. Reynolds, P. Sitaram, A. Ta, and M. Theofanos. Identifying and measuring quality in a software requirements specification. In *Software Metrics Symposium, 1993. Proceedings., First International*, pages 141–152, May 1993.

A. M. Davis. The art of requirements triage. *Computer*, 36(3) :42–49, March 2003.

P. de Chazelles, M. Comes, and A. Kerrien. 1.3.2 Customer focused engineering in airbus a380 programme. *INCOSE International Symposium*, 14(1) :94–103, 2004.

J. de Kervasdoué. *Ils ont perdu la raison*. Robert Laffont, 2014.

E. de Maat, R. Winkels, and T. van Engers. Automated detection of reference structures in law. In *Proceedings of the 2006 Conference on Legal Knowledge and Information Systems : JURIX 2006 : The Nineteenth Annual Conference*, pages 41–50, Amsterdam, The Netherlands, The Netherlands, 2006. IOS Press.

M.-C. de Marneffe, A. N. Rafferty, and C. D. Manning. Finding contradictions in text. In *In ACL 2008*, 2008.

O. L. de Weck, D. Roos, and C. L. Magee. *Engineering Systems : Meeting Human Needs in a Complex Technological World*. MIT Press, 2011.

D. Dermeval, J. Vilela, I. I. Bittencourt, J. Castro, S. Isotani, P. Brito, and A. Silva. Applications of ontologies in requirements engineering : a systematic review of the literature. *Requirements Engineering*, pages 1–33, 2015.

R. Descartes. *Discours de la méthode*. 1637.

O. Djebbi, C. Salinesi, and C. Rolland. Product Line Requirements Configuration in the Context of Multiple Models. *Insight / American Society of Ophthalmic Registered Nurses, Inc*, pages 19 – 20, 2008.

C. Duan. Clustering and its application in requirements engineering. Technical report, DePaul University, College of Computing and Digital Media, 2008.

C. Duan and J. Cleland-Huang. Clustering support for automated tracing. In *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering*, ASE '07, pages 244–253, New York, NY, USA, 2007a.

C. Duan and J. Cleland-Huang. A clustering technique for early detection of dominant and recessive cross-cutting concerns. In *Proceedings of the Early Aspects at ICSE : Workshops in Aspect-Oriented Requirements Engineering and Architecture Design*, EARLYASPECTS '07, pages 1–7, Washington, DC, USA, 2007b. IEEE Computer Society.

C. Duan, P. Laurent, J. Cleland-Huang, and C. Kwiatkowski. Towards automated requirements prioritization and triage. *Requirements Engineering*, 14(2) :73–89, 2009.

J. J. Durillo, Y. Zhang, E. Alba, and A. J. Nebro. A study of the multi-objective next release problem. In *Search Based Software Engineering, 2009 1st International Symposium on*, pages 49–58, May 2009.

J. P. Elm. A study of systems engineering effectiveness - initial results. In *Systems Conference, 2008 2nd Annual IEEE*, pages 1–7, April 2008.

B. Everitt and T. Hothorn. *Introduction to applied multivariate analysis with R.* Springer, Berlin, 2011.

F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami. The linguistic approach to the natural language requirements quality : Benefit of the use of an automatic tool. In *Proceedings of the 26th Annual NASA Goddard Software Engineering Workshop*, SEW '01, Washington, DC, USA, 2001. IEEE Computer Society.

A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, and A. Bouras. A survey of clustering algorithms for big data : Taxonomy and empirical analysis. *IEEE Transactions on Emerging Topics in Computing*, 2(3) :267–279, September 2014.

R. Feldman and J. Sanger. *Text Mining Handbook : Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, New York, NY, USA, 2006.

A. Felfernig, G. Ninaus, H. Grabner, F. Reinfrank, L. Weninger, D. Pagano, and W. Maalej. *An Overview of Recommender Systems in Requirements Engineering*, pages 315–332. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

A. Felfernig, M. Schubert, M. Mandl, F. Ricci, and W. Maalej. Recommendation and decision technologies for requirements engineering. In *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*, RSSE '10, pages 11–15, New York, NY, USA, 2010. ACM.

A. Ferrari, S. Gnesi, and G. Tolomei. *Using Clustering to Improve the Structure of Natural Language Requirements Documents*, pages 34–49. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

L. Ferry. *L'innovation desctructrice*. Flammarion, 2015.

M. A. Finlayson. Java libraries for accessing the princeton wordnet : Comparison and evaluation. In *In Proceedings of the 7th Global Wordnet Conference*, 2014.

D. Firesmith. Quality requirements checklist. *Journal of object technology*, 4(9) :31–38, 2005.

P. Fitsilis, V. Gerogiannis, L. Anthopoulos, and I. K. Savvas. Supporting the requirements prioritization process using social network analysis techniques. In *Enabling Technologies : Infrastructures for Collaborative Enterprises (WETICE), 2010 19th IEEE International Workshop on*, pages 110–115, June 2010.

G. Forman. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3 :1289–1305, March 2003.

G. Forman. *Feature selection for text classification*, pages 257–276. CRC Press Taylor & Francis Group, 2007.

A. Fraga, J. Llorens, L. Alonso, and J. M. Fuentes. *Ontology-Assisted Systems Engineering Process with Focus in the Requirements Engineering Process*, pages 149–161. Springer International Publishing, Cham, 2015.

Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4 :933–969, 2003.

S. Friedenthal, A. Moore, and R. Steiner. *A practical guide to SysML : the Systems Modeling Language*. Morgan Kaufmann, 2nd edition, 2011.

M. Garetti and S. Terzi. Organisational change and knowledge management in plm implementation. *International Journal of Product Lifecycle Management*, 1(1) :43–51, 2005.

C. Garion. *Apports de la logique mathématique en ingénierie des exigences.* Thesis, 2002.

G. Génova, J. M. Fuentes, J. Llorens, O. Hurtado, and V. Moreno. A framework to measure and improve the quality of textual requirements. *Requirements Engineering*, 18(1) :25–41, 2013.

V. Gervasi and D. Zowghi. Reasoning about inconsistencies in natural language requirements. *ACM Trans. Softw. Eng. Methodol.*, 14(3) :277–330, July 2005.

R. Gold. *The plenitude : creativity, innovation, and making stuff*. The MIT Press, 2007.

H. Goldstein. Who killed the virtual case file ? *IEEE Spectrum*, 2005.

W. H. Gomaa and A. A. Fahmy. A survey of text similarity approaches. *International Journal of Computer Applications*, 68(13) :13–18, April 2013. Full text available.

T. S. Group. Chaos manifesto. think big, act small. Report, The Standish Group International, Inc., 2013.

A. D. Hall. *A Methodology for Systems Engineering*. Van Nostrand, Princeton, USA, 1962.

N. Hariri, C. Castro-Herrera, J. Cleland-Huang, and B. Mobasher. *Recommendation Systems in Requirements Discovery*, pages 455–476. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning : data mining, inference, and prediction*. Springer-Verlag New York, 2009.

G. A. Hazelrigg. *Fundamentals of decision making for engineering design and systems engineering*. 2012.

P. Heim, S. Lohmann, K. Lauenroth, and J. Ziegler. Graph-based visualization of requirements relationships. In *Requirements Engineering Visualization, 2008. REV '08.*, pages 51–55, September 2008.

E. Hochmüller. Requirements Classification as a First Step to Grasp Quality Requirements. In *International Workshop on Requirements Engineering : Foundation for Software Quality*, Austria, 1997.

I. Hooks. Writing good requirements. *INCOSE International Symposium*, 4(1) :1247–1253, 1994.

F. Houdek. *Improving Requirements Engineering Processes*, pages 1–2. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

F. Houdek. Managing large scale specification projects. In *19th international working conference on requirements engineering : foundation for software quality, REFSQ 2013*, Essen, Germany, 08-11 April 2013 2013.

V. Hubka and E. W. Eder. *Design science : introduction to the needs, scope and organization of engineering science knowledge*. Springer, 1995.

E. Hull, K. Jackson, and J. Dick. *Requirements engineering*. Springer London, 3rd edition, 2011.

I. Hussain, L. Kosseim, and O. Ormandjieva. *Using Linguistic Knowledge to Classify Non-functional Requirements in SRS documents*, pages 287–298. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

IEEE 1362. IEEE guide for information technology - system definition - concept of operations (conops) document. *IEEE Std 1362-1998*, pages 1–24, December 1998.

M. Ilyas and J. Küng. A comparative analysis of similarity measurement techniques through simreq framework. In *Proceedings of the 7th International Conference on Frontiers of Information Technology*, FIT '09, pages 471–476, 2009.

INCOSE. Systems engineering vision 2020. Technical report, International Council On Systems Engineering, 2007.

INCOSE. Guide for writing requirements. Technical report, International Council on Systems Engineering (INCOSE) Requirements Working Group, 2015a.

INCOSE. *Systems engineering handbook : a guide for system life cycle processes and activities*. Fourth edition, 2015b.

W. Isaacson. *The innovators : how a group of hackers, geniuses, and geeks created the digital revolution*. 2014.

ISO. Information processing systems - open systems interconnection - LOTOS : a formal description technique based on the temporal ordering of observational behaviour = systemes de traitement de l'information - interconnexion de systemes ouverts - LOTOS, 1989.

ISO/IEC/IEEE 15288. ISO/IEC/IEEE international standard - systems and software engineering – system life cycle processes. *ISO/IEC/IEEE 15288 First edition 2015-05-15*, pages 1–118, May 2015.

ISO/IEC/IEEE 29148. Systems and software engineering – life cycle processes –requirements engineering. pages 1–94, December 2011.

S. Jackson. Memo to industry : the crisis in systems engineering. *INSIGHT*, 13(1) :43–43, 2010.

S. Jackson. The essence of complexity : is it subjective or objective ?, January 2015.

G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning with applications in R*. Springer-Verlag New york, 2013.

B. Jeannet and F. Gaucher. Debugging Embedded Systems Requirements with STIMULUS : an Automotive Case-Study. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, TOULOUSE, France, January 2016.

T. Joachims. Text categorization with suport vector machines : Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 137–142, London, UK, UK, 1998.

C. B. Jones. *Systematic Software Development Using VDM (2Nd Ed.)*. Prentice-Hall, Inc., 1990.

D. Jurafsky and J. H. Martin. *Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

J. Kang and P. Saint-Dizier. Discourse structure analysis for requirements mining. *International journal of knowledge content development & technology*, 3(2) :43–65, 2013.

J. Kang and P. Saint-Dizier. *Requirement Compound Mining and Analysis*, pages 186–200. Springer International Publishing, 2014.

J. Kang and P. Saint-Dizier. Une experience de déploiement industriel de lelie : une relecture intelligente des exigences. In *INFORSID*, 2015a.

J. Kang and P. Saint-Dizier. Une expérience d'un déploiement industriel de LELIE : une relecture intelligente des exigences. In *INFORSID*, pages 267–282, 2015b.

J. Karlsson. Software requirements prioritizing. In *Proceedings of the 2Nd International Conference on Requirements Engineering (ICRE '96)*, pages 110–116, Washington, DC, USA, 1996. IEEE Computer Society.

J. Karlsson, C. Wohlin, and B. Regnell. An evaluation of methods for prioritizing software requirements, 1998.

L. Karlsson, P. Berander, B. Regnell, and C. Wohlin. Requirements prioritisation : an experiment on exhaustive pair-wise comparisons versus planning game partitioning, 2004.

L. Karlsson, T. Thelin, B. Regnell, P. Berander, and C. Wohlin. Pair-wise comparisons versus planning game partitioning—experiments on requirements prioritisation techniques. *Empirical Software Engineering*, 12(1) :3–33, 2007.

J. Kasser. Tool to ingest and elucidate requirements (tiger), 2006b.

J. Kasser. Getting the right requirements right. *INCOSE International Symposium*, 22(1) :1005–1020, 2012.

M. Kaufmann and D. Wagner. *Drawing Graphs : Methods and Models*. Springer-Verlag, London, UK, UK, 2001.

M. A. Khan and S. Mahmood. A graph based requirements clustering approach for component selection. *Adv. Eng. Softw.*, 54 :1–16, December 2012.

A. Khoo, Y. Marom, and D. Albrecht. Experiments with sentence classification. In L. Cavedon and I. Zukerman, editors, *Proceedings of the 2006 Australasian language technology workshop*, pages 18–25, 2006.

N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry. Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. *Requir. Eng.*, 13(3) :207–239, August 2008.

C. R. Klaus Pohl. *Requirements Engineering Fundamentals : A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB compliant*. Rocky Nook, 2 edition, 2015.

E. Klein. *Les nouvelles questions posées à la science*, pages 774–785. CAIRN, 2006.

E. Klein. *Allons-nous liquider la science ? Galilée et les indiens*. Flammarion, 2013.

E. Knauss. *Supporting requirements engineers in recognising security issues*, pages 4–18. Lecture notes in computer science. Springer Berlin Heidelberg, 2011.

E. Knauss and D. Ott. *(Semi-) automatic Categorization of Natural Language Requirements*, pages 39–54. Springer International Publishing, 2014.

Y. Ko, S. Park, J. Seo, and S. Choi. Using classification techniques for informal requirements in the requirements analysis-supporting system. *Information and software technology*, 49(11-12) :1128–1140, November 2007.

L. Kof. Natural language processing for requirements engineering : applicability to large requirements documents, 2004.

L. Kof. *Natural language processing : mature enough for requirements documents analysis ?*, volume 3513 of *Lecture notes in computer science*, pages 91–102. Springer Berlin Heidelberg, 2005.

S. körner and T. Brumm. Natural language specification improvement with ontologies. *Semantic computing*, 3(4) :445–470, 2009.

S. J. Körner and T. Brumm. Natural language specification improvement with ontologies. *International Journal of Semantic Computing (IJSC)*, 03(04) :445–470, 2010.

T. Kuhn. A survey and classification of controlled natural languages. *Comput. Linguist.*, 40(1) :121–170, March 2014.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields : Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

C. Lamar. *Linguistic analysis of natural language engineering requirements*. Thesis, 2009.

G. Lami. Quars : A tool for analyzing requirement. Technical Report CMU/SEI-2005-TR-014, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2005.

V. Langenfeld, A. Post, and A. Podelski. Requirements defects over a project lifetime : An empirical analysis of defect data from a 5-year automotive project at bosch. In *REFSQ*, 2016.

A. Lash. *Computational representation of linguistic semantics for requirement analysis in engineering design*. Msc thesis, Clemson University, 2013.

T. Laudan, A. Mauritz, M. Tollenaere, and M. Gardoni. Business meets systems engineering : Facing and handling collaboration challenges in requirements analysis. *INCOSE International Symposium*, 19(1) :637–654, 2009.

P. Laurent, J. Cleland-Huang, and C. Duan. Towards automated requirements triage. In *Requirements Engineering Conference, 2007. RE '07. 15th IEEE International*, pages 131–140, 2007.

Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.

L. Lehtola and M. Kauppinen. *Empirical Evaluation of Two Requirements Prioritization Methods in Product Development Projects*, pages 161–170. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

L. Lehtola, M. Kauppinen, and S. Kujala. *Requirements Prioritization Challenges in Practice*, pages 497–508. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

N. Leveson, M. Heimdahl, H. Hildreth, and J. Reese. Requirements Specification for Process-Control Systems. *IEEE Transactions on Software Engineering*, 20(9) :684–706, September 1994.

S. Liaskos, S. A. McIlraith, S. Sohrabi, and J. Mylopoulos. Representing and reasoning about preferences in requirements engineering. *Requir. Eng.*, 16(3) :227–249, September 2011.

S. L. Lim and A. Finkelstein. Stakerare : Using social networks and collaborative filtering for large-scale requirements elicitation. *IEEE Transactions on Software Engineering*, 38(3) :707–735, May 2012.

S. L. Lim, D. Quercia, and A. Finkelstein. Stakenet : using social networks to analyse the stakeholders of large-scale software projects. In *2010 ACM/IEEE 32nd International Conference on Software Engineering*, volume 1, pages 295–304, May 2010.

J. Lockerbie, N. A. M. Maiden, J. Engmann, D. Randall, S. Jones, and D. Bush. Exploring the impact of software requirements on system-wide goals : a method using satisfaction arguments and i* goal modelling. *Requirements Engineering*, 17(3) :227–254, 2012.

P. Loucopoulos and V. Karakostas. *System Requirements Engineering*. McGraw-Hill, Inc., New York, NY, USA, 1995.

A. Mahmoud. An information theoretic approach for extracting and tracing non-functional requirements. In *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, pages 36–45, August 2015.

C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999.

C. D. Manning, R. Prabhakar, and H. Schütze. *An introduction to information retrieval*. Cambridge University Press, 2008.

C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

C. Mattmann and J. Zitting. *Tika in Action*. Manning Publications Co., Greenwich, CT, USA, 2011.

J. McZara, S. Sarkani, T. Holzer, and T. Eveleigh. Software requirements prioritization and selection using linguistic tools and constraint solvers—a controlled experiment. *Empirical Software Engineering*, 20(6) :1721–1761, 2014.

L. Mich. Nl-oops : from natural language to object oriented requirements using the natural language processing system lolita. *Natural Language Engineering*, 2 :161–187, June 1996.

L. Mich, M. Franch, and P. L. Novi Inverardi. Market research for requirements analysis using linguistic tools. *Requirements Engineering*, 9(2) :151–151, 2004.

P. Micouin. Toward a property based requirements theory : system requirements structured as semilattice. *Systems engineering*, 11(3) :235–245, 2008.

P. Micouin. Property-model methodology : A model-based systems engineering approach using VHDL-ams. *Systems Engineering*, 17(3) :249–263, 2014.

P. Micouin, L. Fabre, and P. Pandolfi. Property Model Methodology : A First Assessment in the Avionics Domain. In *ERTS 2016 8th European Congress Embedded Real Time Software and Systems*, Toulouse, France, January 2016.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. 2013a.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013b.

M. Minsky. Matter, mind and models. In *IFIP Congress*, pages 45–50. Spartan Books, 1965.

I. Mirbel and P. Crescenzo. *From End-User's Requirements to Web Services Retrieval : A Semantic and Intention-Driven Approach*, pages 30–44. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

I. Mirbel and S. Villata. *Enhancing Goal-Based Requirements Consistency : An Argumentation-Based Approach*, pages 110–127. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

I. Mirbel and S. Villata. An Argumentation-based Support System for Requirements Reconciliation. In *Computational Models of Argument - Proceedings of COMMA 2014*, Frontiers in Artificial Intelligence and Applications, pages 467–468, 2014.

L. Mitch and R. Garigliano. Nl-oops : a requirements analysis and tool based on natural language processing. In *Proceedings of third international conference on data mining*, pages 321–330, 2002.

T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

B. Mobasher and J. Cleland-Huang. Recommender systems in requirements engineering. *AI magazine*, 32 :81–89, 2011.

F. Mokammel, E. Coatanea, M. Bakhouya, F. Christophe, and S. Nonsiri. Impact analysis of graph-based requirements models using PageRank algorithm. In *Systems Conference (SysCon), 2013 IEEE International*, pages 731–736, April 2013.

F. Mokammel, E. Coatanéa, F. Christophe, S. Nonsiri, and A. Elman. Analysis and graph representation of requirements models using computational linguistics methods, 2015.

T. Moser, D. Winkler, M. Heindl, and S. Biffl. *Requirements Management with Semantic Technology : An Empirical Study on Automated Requirements Categorization and Conflict Analysis*, pages 3–17. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

J. Mylopoulos, L. Chung, and B. Nixon. Representing and using nonfunctional requirements : a process-oriented approach. *IEEE Transactions on Software Engineering*, 18(6) :483–497, June 1992.

NASA. *Systems engineering handbook*. 2007.

J. Natt och Dag, B. Regnell, P. Carlshamre, M. Andersson, and J. Karlsson. A feasibility study of automated natural language requirements analysis in market-driven development. *Requirements Engineering*, 7(1) :20–33, 2002.

M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23) :8577–8582, 2006.

M. Newman. *Networks : An Introduction*. Oxford University Press, Inc., New York, NY, USA, 2010.

N. Niu, S. Reddivari, and Z. Chen. Keeping requirements on track via visual analytics. In *2013 21st IEEE International Requirements Engineering Conference (RE)*, pages 205–214, July 2013.

Object Modeling Group. *OMG System Modeling Language (OMG SysML)*. 2015.

Object Modeling Group. *OMG Requirements Interchange Format V1.2 (OMG ReqIF)*. 2016.

M. Osborne and C. K. MacNish. Processing natural language software requirement specifications. In *Requirements Engineering, 1996., Proceedings of the Second International Conference on*, pages 229–236, April 1996.

S. Osiński, J. Stefanowski, and D. Weiss. *Lingo : Search Results Clustering Algorithm Based on Singular Value Decomposition*, pages 359–368. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

D. Ott. Defects in natural language requirement specifications at mercedes-benz : An investigation using a combination of legacy data and expert opinion. In *2012 20th IEEE International Requirements Engineering Conference (RE)*, pages 291–296, September 2012.

D. Ott. *Automatic Requirement Categorization of Large Natural Language Specifications at Mercedes-Benz for Review Improvements*, pages 50–64. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

N. Oxman. Age of entanglement. *Journal of Design and Science*, 2016.

G. Pahl, W. Beitz, and K. Wallace. *Engineering design : a systematic approach*. Springer, London, New York, 1996.

F. Palma, A. Susi, and P. Tonella. Using an smt solver for interactive requirements prioritization. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, pages 48–58, 2011.

J. Pennington, R. Socher, and C. D. Manning. Glove : Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

A. Perini, A. Susi, F. Ricca, and C. Bazzanella. An empirical study to compare the accuracy of ahp and cbranking techniques for requirements prioritization. In *Comparative Evaluation in Requirements Engineering, 2007. CERE '07. Fifth International Workshop on*, pages 23–35, October 2007.

A. Perini, A. Susi, and P. Avesani. A machine learning approach to software requirements prioritization. *IEEE Transactions on Software Engineering*, 39(4) :445–461, April 2013.

A. Perini, F. Ricca, and A. Susi. Tool-supported requirements prioritization : Comparing the ahp and cbrank methods. *Inf. Softw. Technol.*, 51(6) :1021–1032, June 2009.

R. Pinquié, P. Véron, F. Segonds, and N. Croué. A collaborative requirement mining framework to support oems. In *Cooperative Design, Visualization, and Engineering - 12th International Conference, CDVE 2015, Mallorca, Spain, September 20-23, 2015. Proceedings*, pages 105–114, 2015a.

R. Pinquié, P. Micouin, P. Véron, and F. Segonds. Property model methodology : a case study with modelica. In I. Horváth, J.-P. Pernot, and Z. Rusák, editors, *TMCE 2016 - 12th international symposium on Tools and Methods of Competitive Engineering*, Tools and Methods of Competitive Engineering : proceedings of the 12th international symposium on Tools and Methods of Competitive Engineering, TMCE 2016, Aix-en-Provence, France, May 9 - 13, 2016, pages 79–91, 2016.

S. Ploux, A. Boussidan, and H. Ji. The Semantic Atlas : an Interactive Model of Lexical Representation. In *Proceedings of the seventh conference of International Language Resources ans Evaluation*, pages 1–5, Valletta, Malta, May 2010.

K. R. Popper. *The Logic of Scientific Discovery*. Hutchinson, London, 1934.

E. Poupart, J.-M. Wallut, and P. Micouin. *Property Model Methodology : A First Application to an Operational Project in the Space Domain*, pages 157–169. Springer International Publishing, 2016.

S. Pugh. *Total design : integrated methods for successful product engineering*. Addison-Wesley, 1991.

L. Rabelo and T. Clark. Modeling space operations systems using sysml as to enable anomaly detection. *SAE International Journal of Aerospace*, 8(2015-01-2388) :189–194, 2015.

A. Rago, C. Marcos, and A. Diaz-Pace. Text analytics for discovering concerns in requirements documents. In *13th Argentine Symposium on Software Engineering, ASSE, 2012*, pages 185–198, La Plata, Argentina, 27-31 August 2012 2012.

M. Ramzan, A. Jaffar, and A. A. Shahid. Value based intelligent requirement prioritization (virp) : expert driven fuzzy logic based prioritization technique. *International Journal of Innovative Computing, Information and Control*, 7(3) :1017–1038, 2011.

A. Rashwan, O. Ormandjieva, and R. Witte. Ontology-based classification of non-functional requirements in software specifications : A new corpus and SVM-based classifier. In *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*, pages 381–386, 2013.

S. Reddivari, Z. Chen, and N. Niu. Recvisu : A tool for clustering-based visual exploration of requirements. In *2012 20th IEEE International Requirements Engineering Conference (RE)*, pages 327–328, September 2012.

S. Reddivari, S. Rad, T. Bhowmik, N. Cain, and N. Niu. Visual requirements analytics : A framework and case study. *Requir. Eng.*, 19(3) :257–279, September 2014.

B. Regnell, R. B. Svensson, and K. Wnuk. *Can We Beat the Complexity of Very Large-Scale Requirements Engineering ?*, pages 123–128. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

N. Riegel and J. Doerr. *A Systematic Literature Review of Requirements Prioritization Criteria*, pages 300–317. Springer International Publishing, 2015.

F. Roda. *Integrating high-level requirements in optimization problems : theory and applications*. Theses, Ecole Polytechnique X, March 2013.

G. Ruhe, A. Eberlein, and D. Pfahl. Quantitative winwin : A new method for decision support in requirements negotiation. In *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*, SEKE '02, pages 159–166, 2002.

G. Ruhe, A. Eberlein, and D. Pfahl. Trade-off analysis for requirements selection. *International Journal of Software Engineering and Knowledge Engineering*, 13(4) :345–366, 2003.

M. J. Ryan. An improved taxonomy for major needs and requirements artifacts. *INCOSE International Symposium*, 23(1) :244–258, 2013.

R. Saaty. The analytic hierarchy process—what it is and how it is used. *Mathematical Modelling*, 9(3) : 161–176, 1987.

R. Saavedra, L. Ballejos, and M. Ale. Software requirements quality evaluation : state of the art and research challenges. In *14th Argentine Symposium on Software Engineering (ASSE)*, pages 240–256, 2013.

J. Sagrado, I. M. Águila, and F. J. Orellana. Multi-objective ant colony optimization for requirements selection. *Empirical Softw. Engg.*, 20(3) :577–610, June 2015.

N. Sannier. *INCREMENT : une approche hybride pour modéliser et analyser dans le large les exigences réglementaires de süreté*. PhD thesis, Université de Rennes 1, 2013.

N. Sannier and B. Baudry. Defining and retrieving themes in nuclear regulations. In *Fifth International Workshop on Requirements Engineering and Law (RELAW 2012)*, Chicago, United States, 2012.

N. Sannier and B. Baudry. INCREMENT : A Mixed MDE-IR Approach for Regulatory Requirements Modeling and Analysis. In C. Salinesi and I. van de Weerd, editors, *REFSQ'2014 - the 20th International Working Conference on Requirements Engineering : Foundation for Software Quality*, Essen, Germany, April 2014. Springer.

N. Sannier, M. Adedjouma, M. Sabetzadeh, and L. Briand. An automated framework for detection and resolution of cross references in legal texts. *Requirements Engineering*, pages 1–23, 2015.

N. Sannier, M. Adedjouma, M. Sabetzadeh, and L. C. Briand. Automated classification of legal cross references based on semantic intent. In *Requirements Engineering : Foundation for Software Quality - 22nd International Working Conference, REFSQ 2016, Gothenburg, Sweden, March 14-17, 2016, Proceedings*, pages 119–134, 2016.

P. Sawyer, P. Rayson, and R. Garside. Revere : Support for requirements synthesis from documents. *Information Systems Frontiers*, 4(3) :343–353, 2002.

J. A. Schumpeter. Capitalism, socialism and democracy, 1942.

C. C. Seepersad, K. Pedersen, J. Emblemsvag, R. Bailey, J. K. Allen, and F. Mistree. The validation square : how does one verify and validate a design method ? In *Decision making in engineering design*, pages 303–314. ASME Press, 2006.

P. C. Selby and R. W. Selby. 4.4.2 measurement-driven systems engineering using six sigma techniques to improve software defect detection. *INCOSE International Symposium*, 17(1) :640–651, 2007.

C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3) : 379–423, 1948.

A. Smith. *An inquiry into the nature and causes of the wealth of nations*. 1776.

Solhenius. Concurrent engineering. *Annals of the CIRP*, 37(2), 1992.

A. Souag, C. Salinesi, and I. Comyn-Wattiau. *Ontologies for Security Requirements : A Literature Survey and Classification*, pages 61–69. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

A. Souag, R. Mazo, C. Salinesi, and I. Comyn-Wattiau. Reusable knowledge in security requirements engineering : a systematic mapping study. *Requirements Engineering*, 21(2) :251–283, 2016.

J. M. Spivey. An introduction to z and formal specifications. *Software Engineering Journal*, 4(1) :40–50, January 1989.

G. Strang. *Linear algebra and its applications*. Thomson, Brooks/Cole, Belmont, CA, 2006.

N. P. Suh. *The principles of design*. Oxford University Press, 1990.

N. P. Suh. *Axiomatic design : advances and applications*. Oxford University Press, 2001.

N. P. Suh. *Complexity : theory and applications*. Oxford University Press, 2005.

N. P. Suh. *Challenges in Designing and Implementing Large Systems (Overcoming Cost Overruns and Missed Project Schedules)*, pages 273–309. Springer International Publishing, 2016.

R. B. Svensson, T. Gorschek, B. Regnell, R. Torkar, A. Shahrokni, R. Feldt, and A. Aurum. Prioritization of quality requirements : State of practice in eleven companies. In *2011 IEEE 19th International Requirements Engineering Conference*, pages 69–78, August 2011.

G. M. Swathi, c. Prasad, and A. Jagan. Writing software requirements specification quality requirements : an approach to manage requirements volatility. *International journal of computer technology and applications*, 2 :631–638, 2011.

R. Szczepaniak and I. Defarge. Lexior : Return from Galileo experience, 2006 2006.

S. Terzi, A. Bouras, D. Debashi, M. Garetti, and D. Kiritsis. Product lifecycle management – from its history to its new role. *International Journal of Product Lifecycle Management*, 4(4) :360–389, 2010.

The Standish Group. Chaos. Technical report, The Standish Group International, Inc., 2005.

P. Thitisathienkul and N. Prompoon. Quality assessment method for software requirements specifications based on document characteristics and its structure. In *Trustworthy Systems and Their Applications (TSA), 2015 Second International Conference on*, pages 51–60, July 2015.

J. J. Thomas and K. A. Cook. *Illuminating the Path : The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Ctr, 2005.

P. Tonella, A. Susi, and F. Palma. Using interactive ga for requirements prioritization. In *Search Based Software Engineering (SSBSE), 2010 Second International Symposium on*, pages 57–66, September 2010.

P. Tonella, A. Susi, and F. Palma. Interactive requirements prioritization using a genetic algorithm. *Inf. Softw. Technol.*, 55(1) :173–187, January 2013.

K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, 2003.

O. T. Tran, B. X. Ngo, M. L. Nguyen, and A. Shimazu. Automated reference resolution in legal texts. *Artificial Intelligence and Law*, 22(1) :29–60, 2014.

D. Ullman. *The Mechanical Design Process*. McGraw-Hill Education, 2015.

A. van Lamsweerde. Goal-oriented requirements engineering : a guided tour. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 249–262, 2001.

A. van Lamsweerde. Elaborating security requirements by construction of intentional anti-models. In *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, pages 148–157, May 2004.

VDI - Association of German Engineers. VDI 2221 - Systematic approach to the development and design of technical systems and products, May 1993.

P. E. Vermaas. *Design Theories, Models and Their Testing : On the Scientific Status of Design Research*, pages 47–66. Springer London, London, 2014.

M. Wang and Y. Zeng. Asking the right questions to elicit product requirements. *International journal of computer integrated manufacturing*, 22(4) :283–293, 2009.

Y. Wautelet, S. Heng, M. Kolp, and I. Mirbel. *Unifying and Extending User Story Models*, pages 211–225. Springer International Publishing, 2014.

K. Wiegers and J. Beatty. *Software requirements*. Microsoft Press, 3rd edition, 2013.

W. M. Wilson, L. H. Rosenberg, and L. E. Hyatt. Automated analysis of requirement specifications. In *Proceedings of the 19th International Conference on Software Engineering*, ICSE '97, pages 161–171, New York, NY, USA, 1997. ACM.

I. H. Witten, E. Frank, and M. A. Hall. *Data Mining : Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

I. Witten. *Text mining*, pages 1–22. Chapman & Hall/CRC Press, Boca Raton, Florida, 2005.

A. W. Wymore. *Model-Based Systems Engineering*. CRC Press, Inc., 1993.

H. Yang, A. De Roeck, V. Gervasi, A. Willis, and B. Nuseibeh. Analysing anaphoric ambiguity in natural language requirements. *Requirements engineering*, 16(3) :163–189, 2011.

E. S.-K. Yu. *Modelling Strategic Relationships for Process Reengineering*. PhD thesis, Toronto, Ont., Canada, Canada, 1996. UMI Order No. GAXNN-02887 (Canadian dissertation).

N. Zeni, N. Kiyavitskaya, L. Mich, J. R. Cordy, and J. Mylopoulos. Gaiust : supporting the extraction of rights and obligations for regulatory compliance. *Requirements Engineering*, 20(1) :1–22, 2015.

H. Zhang, Y. Ouzrout, A. Bouras, A. Mazza, and M. M. Savino. *PLM Components Selection Based on a Maturity Assessment and AHP Methodology*, pages 439–448. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013b.

X. Zhang. *Prise en compte de la valeur ajoutée client dans la spécification des exigences*. Thesis, 2012.

X. Zhang, G. Auriol, H. Eres, and C. Baron. A prescriptive approach to qualify and quantify customer value for value-based requirements engineering. *Int. J. Comput. Integr. Manuf.*, 26(4) :327–345, 2013a.

Y. Zhang, M. Harman, and S. A. Mansouri. The multi-objective next release problem. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, pages 1129–1137, New York, NY, USA, 2007. ACM.

PROPOSITION D'UN ENVIRONNEMENT NUMÉRIQUE DÉDIÉ À LA FOUILLE ET À LA SYNTHÈSE COLLABORATIVE D'EXIGENCES EN INGÉNIERIE DE PRODUITS.

RÉSUMÉ : Il est communément admis que 70 % des coûts du cycle de vie d'un produit sont engagés dès la phase de spécification. Or, aujourd'hui, dans chacune des relations contractuelles client-fournisseur, le fournisseur doit faire face à un amas d'exigences à partir duquel il est difficile de prendre des décisions stratégiques avisées. Pour aider les sous-traitants, nous proposons une méthode outillée de synthèse des exigences, laquelle est supportée par un environnement numérique basé sur les sciences des données. Des modèles de classification extraient les exigences des documents. Les exigences sont ensuite analysées au moyen des techniques de traitement du langage naturel afin d'identifier les défauts de qualité qui mettent en péril le reste du cycle de vie. Pour faciliter leur exploitation, les exigences, dépourvues de leurs principaux défauts, sont non seulement classifiées automatiquement au sein de catégories métiers grâce aux techniques d'apprentissage machine, mais aussi segmentées en communautés au moyen des récentes avancées en théorie des graphes. Chacune des communautés d'exigences est caractérisée par un ensemble configurable de critères d'aide à la décision, dont l'estimation collaborative est assurée par des experts représentant les diverses fonctions de l'entreprise. Enfin, une synthèse graphique des estimations est restituée au décideur via un tableau de bord de résumés statistiques descriptifs facilitant la prise de décisions informées. La validation théorique et empirique de notre proposition corrobore l'hypothèse que les sciences des données est un moyen de synthétiser plusieurs centaines ou milliers d'exigences.

Mots clés : exigence, spécification, fouille de données, graphe, aide à la décision, ingénierie.

A COLLABORATIVE REQUIREMENT MINING FRAMEWORK.

ABSTRACT: It is broadly accepted that 70 % of the total life cycle cost is committed during the specification phase. However, nowadays, we observe a staggering increase in the number of requirements. We consider the tremendous volume of requirements as big data with which sub-contractors struggle to make strategic decisions early on. Thus, we propose to methodologically integrate data science techniques into a collaborative requirement mining framework, which enables decision-makers to gain insight and discover opportunities in a massive set of requirements. Initially, classification models extract requirements from prescriptive documents. Requirements are subsequently analysed with natural language processing techniques so as to identify quality defects. After having removed the quality defects, the analyst can navigate through clusters of requirements that ease the exploration of big data. Each cluster gathers the requirements that belong to a functional area (mechanics, electronics, IT, etc.). Each domain expert can therefore easily filter out the requirements subset that is relevant for him. A complementary approach consists in detecting communities of requirements by analysing the topology of a graph. Each community owns a customisable set of decision-making criteria which are estimated by all functional areas. A dashboard of statistical visuals distils the estimation results from which a decision maker can make informed decisions. We conclude that the theoretical and empirical validation of our proposition corroborates the assumption that data science is an effective way to gain insight from hundreds or thousands of requirements.

Keywords : requirement, specification, data mining, graph, decision-making, engineering.

ARTS ET MÉTIERS ParisTech

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY