



THÈSE DE DOCTORAT

Justification Factory : de l'élicitation d'exigences de justification jusqu'à leur production en continu

Clément DUFFAU

CNRS, LABORATOIRE I3S, SOPHIA-ANTIPOLIS

PRÉSENTÉE EN VUE DE L'OBTENTION
DU GRADE DE DOCTEUR EN INFORMATIQUE
DE UNIVERSITÉ CÔTE D'AZUR

DIRIGÉE PAR: MIREILLE BLAY-FORNARINO

CO-ENCADRÉE PAR: THOMAS POLACSEK

TUTEURS INDUSTRIELS : BARTOSZ GRABIEC
ET JEAN-LOUIS DIVOUX

SOUTENUE LE : 16 NOVEMBRE 2018

DEVANT LE JURY, COMPOSÉ DE:

MME MIREILLE BLAY-FORNARINO,
PROFESSEUR DES UNIVERSITÉS, CÔTE D'AZUR
DIRECTRICE

M. JEAN-MICHEL BRUEL,
PROFESSEUR DES UNIVERSITÉS, TOULOUSE
RAPPORTEUR

M. PHILIPPE COLLET,
PROFESSEUR DES UNIVERSITÉS, CÔTE D'AZUR
EXAMINATEUR

M. JEAN-LOUIS DIVOUX,
DIRECTEUR DES PROGRAMMES, AXONIC, SOPHIA ANTIPOLIS
TUTEUR INDUSTRIEL

MME RÉGINE LALEAU,
PROFESSEUR DES UNIVERSITÉS, PARIS-EST CRÉTEIL
RAPPORTEUR

M. THOMAS POLACSEK,
RESEARCH SCIENTIST, ONERA, TOULOUSE
ENCADRANT



Justification Factory : de l'élicitation d'exigences de justification jusqu'à leur production en continu

Jury

Président du jury

- M. Philippe Collet, Professeur des Universités, Côte d'Azur

Rapporteurs

- M. Jean-Michel Bruel, Professeur des Universités, Toulouse
- Mme Régine Laleau, Professeur des Universités, Paris-Est Créteil

Invité

- M. Jean-Louis Divoux, Directeur des programmes, AXONIC, Sophia Antipolis

Titre : *Justification Factory*, de l'élicitation d'exigences de justification jusqu'à leur production en continu

Résumé

Dans de nombreux domaines où il existe des risques pour l'homme, comme la médecine, le nucléaire ou l'avionique, il est nécessaire de passer par une phase de certification visant à garantir le bon fonctionnement d'un système ou d'un produit. La certification se fait en fonction de documents normatifs qui expriment les exigences de justifications auxquelles le produit et le processus de développement doivent se conformer. Un audit de certification consiste alors à produire une documentation attestant la conformité avec ce cadre réglementaire. Pour faire face à ce besoin de justifications visant à assurer la conformité avec les normes en vigueur et la complétude des justifications apportées, il faut dès lors être capable de cibler les exigences de justification à revendiquer pour un projet et produire les justifications durant le développement du projet. Dans ce contexte, élucider les exigences de justifications à partir des normes et produire les justifications nécessaires et suffisantes sont des enjeux pour assurer le respect des normes et éviter la sur-justification.

Dans ces travaux nous cherchons à structurer les exigences de justification pour ensuite aider à la production des justifications associées tout en restant attentif à la confiance que l'on peut placer en elles. Pour relever ces défis, nous avons défini une sémantique formelle pour une modélisation existante des justifications : les Diagrammes de Justification. A partir de cette sémantique, nous avons pu définir un ensemble d'opérations permettant de contrôler le cycle de vie des justifications pour assurer la conformité des justifications au regard des exigences de justification. Par ce formalisme, nous avons également pu guider, voire automatiser dans certains cas, la production des justifications et la vérification de la conformité.

Ces contributions ont été appliquées dans le contexte des technologies médicales pour l'entreprise AXONIC, porteuse de ces travaux. Ceci a permis de i) élucider les exigences de justification des normes médicales et pratiques internes de l'entreprise, ii) produire automatiquement les justifications associées à la norme IEC 62304 pour le logiciel en médical, iii) automatiser la vérification et validation des justifications ainsi que la production de documents utilisables lors d'audit.

Mots clés

Justification, Certification, Domaines critiques, Modélisation, Automatisation

Title : *Justification Factory*, from justification requirements elicitation to their continuous production

Abstract

In many areas where it exists human risks, such as medicine, nuclear or avionics, it is necessary to go through a certification stage to ensure the proper functioning of a system or product. Certification is based on normative documents that express the justification requirements to which the product and the development process must conform. A certification audit then consists of producing documentation certifying compliance with this regulatory framework. To cope with this need for justifications to ensure compliance with the standards in force and the completeness of the justifications provided, it must therefore be able to target the justification requirements to be claimed for a project and produce justifications during the development of the project. In this context, eliciting the justification requirements from the standards and producing the necessary and sufficient justifications are issues to ensure compliance with standards and avoid over-justification.

In these works we seek to structure the justification requirements and then help to produce the associated justifications while remaining attentive to the confidence that can be placed in them. To address these challenges, we have defined a formal semantics for an existing model of justifications: Justification Diagrams. From this semantics, we have been able to define a set of operations

to control the life cycle of the justifications to ensure that the justifications regarding the justification requirements. Through this semantics, we have also been able to guide, and even automate in some cases, the production of justifications and the verification of conformance.

These contributions were applied in the context of medical technologies for the company AXONIC, the bearer of this work. This made it possible to i) elicitate the justification requirements of the medical standards and company's internal practicals, ii) automatically produce the justifications associated with the IEC 62304 standard for medical software, iii) automate the verification and validation of the justifications as well as the production of documents that can be used during the audit.

Keywords

Justification, Certification, Critical domains, Modeling, Automation

Table des matières

Table des matières	v
Liste des figures	vii
Liste des tableaux	ix
1 Introduction	1
1.1 Contexte et Problématique	2
1.2 Contribution : de l'élicitation d'exigences de justifications jusqu'à la production continue de justifications	3
1.3 Plan du document	4
1.4 Exemple fil rouge	5
I État de l'art	7
2 Modélisation des justifications	9
2.1 Schémas argumentaires	10
2.2 Cas d'assurance structuré	13
2.3 Organisation des connaissances	15
3 Processus de justification	19
3.1 Processus de certification	20
3.2 De l'ingénierie des exigences à l'élicitation d'exigences de justifications	22
3.3 Production des justifications	25
4 Positionnement et objectifs	29
4.1 Espace couvert par l'état de l'art	30
4.2 Raffinement des objectifs de la contribution	31
II Contribution	35
5 Formalisation des Diagrammes de Justification (DJs)	37
5.1 Une sémantique formelle pour les Diagrammes de Justification (DJs)	38
5.2 La sémantique en actions	42
5.3 Discussion	45
6 Cycles de vie des Diagrammes de Justification (DJs)	47
6.1 Diagrammes-Patron de Justification (DPJs)	48
6.2 Opérations entre Diagramme-Patron de Justification (DPJ) et Diagramme de Justification (DJ)	53
6.3 Cycles de vies des opérations	57
6.4 Discussion	59

7 Production guidée des justifications	61
7.1 Automatisation de la construction des justifications	62
7.2 Automatisation de la vérification et validation des justifications	64
7.3 Justification Factory, vers une chaîne d'outils	66
7.4 Interactions avec d'autres formalismes	68
7.5 Discussion	73
III Validation	75
8 Application pour la certification ISO 13485 dans une entreprise de technologies médicales	77
8.1 AXONIC : Contexte des technologie médicales	78
8.2 Environnement technique	79
8.3 Elicitation des exigences de justifications pour IEC 62304	80
8.4 Production automatisée des justifications pour la IEC 62304	84
8.5 Des Diagrammes de Justification (DJs) vers l'intégration dans le Système de Management de la Qualité (SMQ) d'AXONIC	88
9 Conclusion et perspectives	93
9.1 Autres cas d'application	94
9.2 Leçons apprises et limites	99
9.3 Perspectives	101
9.4 Synthèse	105
9.5 Liste des publications	107
A Annexes	I
A.1 Méta-modèle détaillé pour les Diagrammes de Justification (DJs)	I
A.2 Interfaces des services exposés par <i>Justification Factory</i>	IV
A.3 Diagramme de séquence pour l'orchestration du <i>Justification bus</i>	V
A.4 Autre exemple de I* et Diagramme de Justification (DJ)	V
A.5 Diagramme-Patron de Justification (DPJ) pour IEC 62304 du prototypage jusqu'au développement	VII
A.6 Données brutes de l'étude préliminaire sur la compréhension des Diagrammes-Patron de Justification (DPJs)	IX
A.7 Diagramme-Patron de Justification (DPJ) complet pour le dossier de conception d'AXONIC	IX
A.8 Données brutes de l'étude sur la production automatique des justifications pour IEC 62304	XIV
A.9 Données brutes de l'étude sur l'intégration au SMQ d'AXONIC	XVI

Liste des figures

1.1 Critère Réglementaire n°4 mettant en évidence les indicateurs et les éléments de preuve associés	5
2.1 Exemple de schéma argumentaire de Toulmin sur l'exemple fil rouge	11
2.2 Notation graphique d'un pas de justification inspirée de GSN	11
2.3 Exemple de Diagramme de Justification (DJ) associé au CR4 de Datadock dans le cadre du recrutement d'un vacataire	12
2.4 Exemple de diagramme GSN associé au CR4 de Datadock dans le cadre du recrutement d'un vacataire	14
2.5 Extrait du méta-modèle de SACM (partie argumentation)	15
2.6 IBIS sur Datadock	16
3.1 Processus de certification	21
3.2 Extrait du méta-modèle d'i* 2.0	24
3.3 Diagramme i* sur l'exemple de Datadock	25
5.1 Exemple de relation \mathcal{R} , affichée comme une extension de la notation des DJ (e.g., un <i>Template de réponse IUT</i> est conforme à un <i>Avis des experts</i>)	39
5.2 Noyau du méta-modèle de DJ	44
6.1 Diagramme-Patron de Justification (DPJ) sur la gauche et un Diagramme-Patron de Justification (DPJ) raffiné à partir de ce premier sur la droite sur l'exemple Datadock	50
6.2 Exemple d'évolution d'un Diagramme-Patron de Justification (DPJ) et ses impacts	51
6.3 Diagramme-Patron de Justification (DPJ) sur la gauche et un Diagramme de Justification (DJ) associé sur la droite pour l'étude de l'existant en vue de préparer Datadock	52
6.4 Gestion des compétences dans le référentiel RE.S.E.A.U.	53
6.5 Exemple de DPJ et DJ avant alignement	56
6.6 Modélisation des opérations	57
6.7 Exemple de flot d'utilisation des opérations pour l'évolution conjointe des exigences de justifications et la production des justifications	58
6.8 Exemple de flot d'utilisation des opérations pour la structuration d'exigences de justifications	58
7.1 Exemple de Système de Justification (SJ) sur Datadock	64
7.2 Écosystème de <i>Justification Factory</i>	66
7.3 GUI pour <i>Justification Factory</i>	68
7.4 Extrait du méta-modèle de SACM et les liens vers le méta-modèle de DJ	69
7.5 Exemple sur Datadock de collaboration entre i* et DJ	71
7.6 Points d'interaction entre DJ méta-modèle (en bleu) et i* 2.0 méta-modèle (en jaune)	72
8.1 DPJ à la fin des activités de <i>kick-off</i> du projet	80
8.2 DPJ à l'étape de conception	82
8.3 DPJ à l'étape de développement	82

8.4	Extrait du processus de V&V d'AXONIC	84
8.5	Écosystème de la chaîne d'intégration continue	85
8.6	DPJs pour valider l'exécution des tests d'intégration logicielle	87
8.7	Extensions de <i>Justification Factory</i> pour les outils d'AXONIC	89
9.1	Diagramme-Patron de Justification (DPJ) à l'étape de conception préliminaire	95
9.2	Architecture entre <i>Experience Factory</i> et <i>Justification Factory</i>	98
9.3	Exemple de DJ sur ROCKFlows	99
9.4	Relation entre <i>BPMN</i> et DJ	102
A.1	Méta-modèle détaillé DJ	II
A.2	Méta-modèle des DJ au niveau du code	III
A.3	API des services	IV
A.4	API des services	V
A.5	Exemple de collaboration entre i* et DJ	VI
A.6	DPJ représentant les exigences de justification pour IEC 62304 (du prototypage jusqu'au développement)	VIII
A.7	DPJ du dossier de conception	IX
A.8	Sous-DPJ du dossier de conception portant sur la revue d'initialisation	X
A.9	Sous-DPJ du dossier de conception portant sur la revue de faisabilité	X
A.10	Sous-DPJ du dossier de conception portant sur la revue des données d'entrée	XI
A.11	Sous-DPJ du dossier de conception portant sur la revue des spécifications techniques	XII
A.12	Sous-DPJ du dossier de conception portant sur la revue de développement	XIII
A.13	DPJ pour valider les livrables logiciels d'AXONIC	XV

Liste des tableaux

4.1 Matrice de comparaison de l'état de l'art	31
9.1 Synthèse des contributions	106
A.1 Questionnaire pour l'étude sur la compréhension des Diagrammes-Patron de Justification (DPJs)	IX
A.2 Données brutes de l'étude de la production automatique des justifications	XIV
A.3 Données brutes de l'étude du SMQ	XVI

Chapitre 1

Introduction

Sommaire

1.1 Contexte et Problématique	2
1.2 Contribution : de l'élicitation d'exigences de justifications jusqu'à la production continue de justifications	3
1.2.1 \mathcal{O}_1 : Capturer des exigences de justifications	3
1.2.2 \mathcal{O}_2 : Guider la production des justifications	3
1.2.3 \mathcal{O}_3 : Vérifier et analyser la conformité entre exigences et production des justifications	4
1.3 Plan du document	4
1.4 Exemple fil rouge	5

1.1 Contexte et Problématique

Dans de nombreux domaines où il existe des risques pour l'homme, tels que la médecine, le nucléaire, l'avionique ou les transports publics, une phase de certification visant à garantir le bon fonctionnement d'un système ou d'un produit est requise. Cette certification est délivrée par une partie tierce, appelée autorité de certification. Il est important de comprendre que les activités de certification ne se concentrent pas seulement sur le produit final, mais concernent tous les aspects du processus, de la conception à la production. Dans la pratique, la certification s'obtient par la démonstration de la conformité du produit ou du processus à des référentiels normatifs déterminant les exigences auxquelles ils doivent répondre. Par exemple, la norme applicable aux instruments médicaux (ISO 13485) décrit le processus de haut niveau qui doit être suivi à chaque étape de développement. Dans le cadre de cette norme, l'audit de certification consiste, pour l'industriel, à produire, présenter et gérer un système documentaire selon lequel les processus suivis seront conformes au référentiel normatif. Ces documents comprennent des explications logicielles, des résultats d'essais ou des protocoles d'essais cliniques suivis pendant les expérimentations. Dans les systèmes logiciels avioniques, l'autorité de certification suit le processus de développement de bout en bout et elle doit être convaincue que le processus et le produit sont conformes à la directive DO 178 et à l'ARP4754. Ainsi, une grande partie des activités de certification est liée à la production de documents justificatifs correspondant à une argumentation pour démontrer à l'autorité la conformité aux référentiels.

Historiquement, les activités de certification sont fortement liées aux contextes critiques comme l'aéronautique, la santé, le ferroviaire et l'automobile. Cependant, le besoin de produire un ensemble de justifications pour convaincre de la validité d'une activité est de plus en plus étendu à d'autres domaines comme le management du risque, les décisions stratégiques ou l'IA. Dans ce contexte étendu, l'entité concernée par ces aspects de justification, que nous nommerons *second client*, peut-être un manager du projet, une autorité de certification ou le client du produit lui-même. Contrairement au client final qui sera intéressé par le produit lui-même sans regard sur la façon dont il a été développé, le *second client* est principalement intéressé par l'accomplissement d'exigences portant sur la qualité.

Nous pouvons faire le parallèle entre les activités du *second client* et les activités dans le domaine de la simulation. Dans ce domaine, la Verification, Validation and Accreditation (VV&A) [BALCI \[1998\]](#) est définie comme l'activité impliquant une autorité de certification qui valide qu'un modèle ou une simulation peut être utilisés pour un usage spécifique. Pour atteindre ce but, il est nécessaire d'avoir une documentation exhaustive, un ensemble de justifications expliquant non seulement les résultats, mais aussi les données d'entrée, les hypothèses, les techniques employées, etc. Les activités d'accréditation consistent alors à collecter et évaluer cette documentation.

Comme montré par Knauss [KNAUSS et collab. \[2017\]](#), dans le contexte d'un projet où la sûreté et sécurité sont un enjeu, ces exigences de qualité sont difficiles à spécifier. Pour faire face à ce besoin de justifications dans le but d'être en conformité avec les normes en vigueur et d'être sûr de la complétude des justifications apportées, nous observons une pratique courante qui consiste à enregistrer, tracer, rapporter et motiver tout ce qui est fait [VERNAY et KETELS \[2007\]](#). On parle alors de *surqualité*. Ceci mène à une inflation de la documentation qui génère du bruit. En effet, certains documents sont inutiles dans le cadre des justifications, car ils n'apportent pas de valeur (e.g. logs), sont redondants (e.g. même information dans des formats différents à travers de multiples documents). Ce tsunami documentaire semble apporter illusoirement une meilleure confiance, mais dans la réalité coûte cher à produire et les documents inutiles rajoutent du bruit dans les justifications.

De plus, le développement d'un projet est souvent difficile à prédire. Des changements fondamentaux (e.g. modifications matérielles, retour de résultats expérimentaux sur le terrain) peuvent arriver durant la phase de développement. Dès lors, ces évolutions ont un impact sur les activités de justifications qui doivent être adaptées. Une pratique courante pour minimiser cet impact consiste ainsi à produire les justifications requises seulement au moment où elles sont requises. En effet, certaines justifications sont difficiles, voire impossibles à produire avant certaines étapes

du projet (e.g., protocole de tests avant d'avoir décrit les spécifications, procédure de déploiement avant d'avoir choisi les technologies). Il est donc important d'assurer l'incrémentalité dans les activités de certification.

Ce lexique pose un vocabulaire utile pour cette thèse et vulgarisé de la démarche qualité présenté par le Mouvement Français pour la Qualité **POUR LA QUALITÉ [2005]**.

1.2 Contribution : de l'élicitation d'exigences de justifications jusqu'à la production continue de justifications

Dans ces travaux nous cherchons à structurer les exigences de justifications pour ensuite aider à la production des justifications tout en restant attentifs à la confiance que l'on peut placer en elles. De ceci découlent les trois objectifs ci-dessous auxquels nos travaux visent à répondre. Ces trois objectifs seront raffinés à travers les chapitres suivants (Partie I).

1.2.1 \mathcal{O}_1 : Capturer des exigences de justifications

Pour répondre aux exigences de justifications, il est attendu par l'organisme d'accréditation qu'une entreprise identifie et revendique les normes qui la concernent pour ses activités. Un audit de l'entreprise par l'organisme de certification consiste ainsi à vérifier et valider le bon respect de ces normes voire à lui demander de se conformer et respecter d'autres normes. Néanmoins, ces normes et guides ne décrivent que peu la mise en œuvre ou des outils particuliers pour atteindre la conformité requise. Il est alors de la responsabilité de l'entreprise d'identifier et mettre en œuvre les moyens de maîtrise permettant de répondre aux normes revendiquées. Ces pratiques internes sont parfois revues et validées par l'organisme d'accréditation avant le début d'un projet (e.g., avionique où l'organisme d'accréditation se met d'accord sous seing privé, nommé *Certification Review Item*, avec le manufacturier sur le processus de mise au point d'un appareil). Cet ensemble de normes, guides et pratiques internes englobe les exigences de justifications des plus générales aux plus spécifiques pour une entreprise. D'autre part, certaines normes sont très liées entre elles (e.g., en médical la gestion des risques avec ISO 14971 donne le cadre général qui est complété pour chaque norme orientée métier. Les exigences de justification concernant les risques se retrouvent ainsi diffuses à travers plusieurs normes). Le processus de raffinement des exigences de justifications en des exigences adaptées à son contexte est donc une tâche complexe et coûteuse pour une entreprise. Par ailleurs, ces normes sont amenées à évoluer pour tenir compte de nouvelles exigences réglementaires. Ceci mène les entreprises à identifier les nouveautés dans les normes et réviser les exigences de justifications pour être conforme. Ces tâches nécessitent au sein des entreprises une veille réglementaire continue et une analyse précise des impacts sur les exigences de justifications des projets existants. Pour finir, une entreprise acquiert un savoir-faire tout au long des différents projets en matière de développement, mais également en matière d'affaires réglementaires. Être capable de capitaliser sur ses expériences passées pour gagner en productivité et confiance est un enjeu fort pour la réussite des projets. Pour autant, les exigences de justifications restent souvent textuelles, conceptuelles et difficilement transposables. Nous allons donc nous intéresser à comment **Capturer des exigences de justifications** (\mathcal{O}_1).

1.2.2 \mathcal{O}_2 : Guider la production des justifications

Une fois que cette structuration complexe des exigences de justifications entrelaçant normes, guides et pratiques internes a été mise en place par une entreprise pour répondre aux exigences de justifications qui lui sont propres, il s'agit pour elle d'être capable de produire de telles justifications. Ici aussi la tâche est complexe puisqu'il s'agit pour chaque corps de métier concerné de comprendre les exigences de justifications qui lui sont spécifiques et d'identifier quand et comment

produire les justifications demandées. Des disparités entre les exigences définies et ce qui peut être produit peuvent parfois mener à des révisions des exigences ou à la mise en place de nouveaux moyens de maîtrise. De plus, produire ces justifications repose souvent sur un ensemble hétérogène d'outils faiblement liés où par strates successives les documents sont ajoutés au système de management de la qualité (SMQ). Le management d'un tel système devient ainsi difficile à maîtriser. La production de ces justifications est donc une tâche complexe et coûteuse pour l'entreprise. Nous allons donc nous intéresser à comment **Guider la production des justifications** (\mathcal{O}_2).

1.2.3 \mathcal{O}_3 : Vérifier et analyser la conformité entre exigences et production des justifications

Une fois l'ensemble des justifications produit, il s'agit d'être capable d'identifier si cet ensemble de justifications répond bien aux exigences définies initialement. Cette revue est d'autant plus complexe que les moyens pour répondre à l'ensemble des exigences sont hétérogènes et diffus. En effet, il s'agit dans ce cas pour le reviewer de devoir maîtriser un ensemble de logiciels conséquent. Attester que ce qui a été produit est suffisant est une tâche complexe et coûteuse pour l'entreprise. Ce qui conduit souvent à produire plus de justifications que nécessaire, car cela apporte naïvement plus de confiance pour répondre aux exigences de justifications. Cette pratique introduit un coût supplémentaire pour l'entreprise qui produit ainsi des documents non nécessaires. Comme précisé en \mathcal{O}_1 , des changements normatifs ont lieu. L'évolution des exigences induites par ces changements doit être scrupuleuse analysée et vérifiée pour de nouveau atteindre la conformité. Nous allons donc nous intéresser à **Vérifier et analyser la conformité entre exigences et production des justifications** (\mathcal{O}_3) afin d'aider à produire les justifications nécessaires et suffisantes.

1.3 Plan du document

Cette thèse est répartie en dix chapitres comme suit :

PARTIE I : Etat de l'art Cette partie analyse l'état de l'art et positionne nos objectifs au regard de celui-ci

CHAPITRE 2 Ce chapitre se focalise sur différentes modélisations de justifications. Nous parcourons ainsi les aspects argumentaires, cas d'assurance et organisations des connaissances.

CHAPITRE 3 Ce chapitre sera consacré à une étude approfondie des processus et activités liés à la justification. Nous parcourons ainsi les processus des domaines critiques, mais également les aspects connexes comme l'ingénierie des exigences et la traçabilité.

CHAPITRE 4 Ce chapitre compare l'état de l'art présenté précédemment et positionne notre contribution au regard de celui-ci. Ce chapitre se conclut ainsi par le raffinement de nos objectifs.

PARTIE II : Contribution Cette partie présente les contributions scientifiques de nos travaux.

CHAPITRE 5 Ce chapitre décrit la première contribution de cette thèse, sur l'objectif \mathcal{O}_1 , qui vise à définir et utiliser un formalisme pour l'expression d'exigences de justifications.

CHAPITRE 6 Ce chapitre décrit la deuxième contribution, sur l'objectif \mathcal{O}_3 , de cette thèse qui vise à définir et utiliser des opérations entre les exigences de justifications et les justifications elles-mêmes pour encadrer l'évolution de celles-ci au cours de leur cycle de vie.

CHAPITRE 7 Ce chapitre décrit la troisième contribution de cette thèse, sur l'objectif \mathcal{O}_2 , qui vise à encadrer et partiellement automatiser la production des justifications en se basant sur le formalisme précédemment défini. Ce chapitre permet de poser une

première architecture pour faciliter l'interaction avec d'autres formalismes pour la justification, mais également établir le lien avec l'ingénierie des exigences.

PARTIE III : Validation Cette partie démontre comment notre contribution répond bien à nos objectifs.

CHAPITRE 8 Ce chapitre valide les contributions par rapport aux objectifs de cette thèse à travers un cas d'application industrielle issu de l'entreprise AXONIC, porteuse de cette thèse.

CHAPITRE 9 Ce dernier chapitre met en avant d'autres études de cas que nous avons menés, dresse une conclusion et présente les perspectives de recherche et d'industrialisation de cette thèse.

1.4 Exemple fil rouge

Pour illustrer les thèmes abordés dans l'état de l'art et chaque étape clé de notre contribution dans ces travaux, nous utilisons un exemple concret : la validation de la qualification du personnel pour l'IUT de l'université Côte d'Azur. Cet exemple bien que non issu d'un domaine critique est contraint par les normes à travers le référentiel Datadock et les pratiques internes de l'université. Cet exemple a également l'avantage de montrer l'intérêt de notre approche sur un cas où les pratiques internes doivent évoluer pour changer de réglementation (passage de RE.S.E.A.U à Datadock) qui entraîne un besoin de (i) capitaliser sur l'existant, (ii) identifier les manquements au regard du nouveau référentiel (iii) produire les justifications manquantes et (iv) valider les justifications ainsi produites. Ces quatre points clés sont, à une plus large échelle, des problématiques des domaines critiques tels que l'aéronautique, le ferroviaire ou le médical. Au regard de nos objectifs, \mathcal{O}_1 correspond à (i) et (ii), \mathcal{O}_2 à (iii) et \mathcal{O}_3 à (ii) et (iv).

La législation impose aux organismes de formation une unification de leur charte qualité menant à six critères portant sur l'identification des objectifs des formations, l'adaptation et suivi pédagogique aux publics cibles, l'adéquation des moyens pédagogiques, la qualification professionnelle du personnel, les conditions d'accès et suivi des formations et la prise en compte des appréciations des stagiaires. Pour cet exemple, nous allons nous intéresser particulièrement à la qualification du personnel (CR4). Pour ce critère, la réglementation identifie trois indicateurs et cinq documents à l'appui comme le montre la Figure 1.1.

CRITÈRES RÉGLEMENTAIRE N°4: LA QUALIFICATION PROFESSIONNELLE ET LA FORMATION CONTINUE DU PERSONNEL EN CHARGE DES FORMATIONS

Indicateurs	Éléments de preuve
4.1 Capacité de l'OF à produire et mettre à jour une base des expériences et qualifications des formateurs	Attestation de l'existence d'une CV- thèque mise à jour de ses formateurs
4.2 Capacité de l'OF à attester des actions de formation continue du corps de formateurs ou du formateur indépendant	Attestation de présence aux formations Descriptif des actions de formation et de professionnalisation des formateurs
4.3 Capacité de l'OF à produire des références (cadre B to B)	Attestation de référence clients, Appartenance ou existence en interne d'un réseau d'experts

FIGURE 1.1 – Critère Réglementaire n°4 mettant en évidence les indicateurs et les éléments de preuve associés

Première partie

État de l'art

Chapitre 2

Modélisation des justifications

Sommaire

2.1 Schémas argumentaires	10
2.1.1 Toulmin	10
2.1.2 Diagramme de Justification (DJ)	11
2.2 Cas d'assurance structuré	13
2.2.1 GSN et CAE	13
2.2.2 SACM	14
2.3 Organisation des connaissances	15

Le besoin de produire un argumentaire pour convaincre qu'un résultat est bien fondé ne se pose pas uniquement dans le contexte de la certification. Au début des années quatre-vingt-dix, avec la croissance des systèmes intelligents tels que les systèmes experts, les systèmes d'aide à la décision et les systèmes de prévision, de nombreux travaux ont insisté sur la nécessité de fournir une explication pour qu'un humain accepte un résultat automatique **ASSOCIATION FOR THE ADVANCEMENT OF ARTIFICIAL INTELLIGENCE [1992]** **ASSOCIATION FOR THE ADVANCEMENT OF ARTIFICIAL INTELLIGENCE [1993]**.

Dans ce domaine, des travaux comme ceux de **CHANDRASEKARAN et collab.** [1989] et **SOUTHWICK** [1991] soulignent le fait que les explications fournies par un système intelligent peuvent être classées en trois catégories. Tout d'abord, une trace des informations, les règles, ainsi que les étapes qui ont permis au système de produire le résultat correspondent à des explications, appelées *traces explicatives*. Deuxièmement, les *explications stratégiques* expliquent pourquoi une information est utilisée avant une autre, comment les différentes étapes du raisonnement sont choisies et comment elles contribuent à atteindre les objectifs principaux. Les explications stratégiques fournissent une explication du fonctionnement général du système. Troisièmement, les *explications profondes* justifient les fondements du système. Ces explications fournissent les théories à partir desquelles le résultat a été généré, les logiques sous-jacentes et les justifications de la base de connaissances.

Dans le cadre de la certification, nous sommes typiquement dans les explications profondes et, partiellement, dans les explications stratégiques. Ce qui est conforme avec **YE et JOHNSON** [1995] qui expliquent que ce sont les explications profondes qui induisent la meilleure acceptation du système par des utilisateurs et la certification vise justement à convaincre une autorité. Il coexiste ainsi différentes modélisations de telles explications que nous présentons dans la suite de ce chapitre.

2.1 Schémas argumentaires

2.1.1 Toulmin

TOULMIN [2003] propose un schéma générique permettant de représenter de façon structurée un argument. Dans ce schéma, tout argument peut-être composé d'une revendication ou conclusion notée (C), "conclusion whose merits we are seeking to establish" et de faits, de données notés (D), "the facts we appeal to as a foundation for the claim". En effet, pour bien argumenter une conclusion, il s'agit de reposer sur des faits. Pour justifier, la transition de données vers la conclusion, les éléments additionnels sont utilisés, parfois de façon implicite, appelé garantie (Q). Une garantie (W) correspond au processus de raisonnement et établit la connexion logique entre les données et la conclusion. Les données et la garantie sont les supports pour l'argument et la distinction entre données et garantie n'est pas forcément facile à rendre explicite. Les garanties sont des règles générales, elles attestent de la puissance de l'argument tandis que la donnée reste une évidence factuelle. Toulmin ajoute le concept d'appui (B) qui se lie à la justification par la garantie. L'appui vient donc supporter la garantie, c'est la justification de pourquoi la garantie est une bonne raison d'accepter la conclusion. Finalement, la revendication n'est pas toujours nécessairement vraie, il est possible d'exprimer des réserves avec le qualificatif modal (R).

Ici, nous n'avons aucune formule logique. Il n'existe pas de système axiomatique qui permette de donner une valeur de validité à la formule logique $D \rightarrow C$. Nous sommes ici dans un cadre rhétorique où nous essayons de définir ce qui est en relation avec un bon argument ou non.

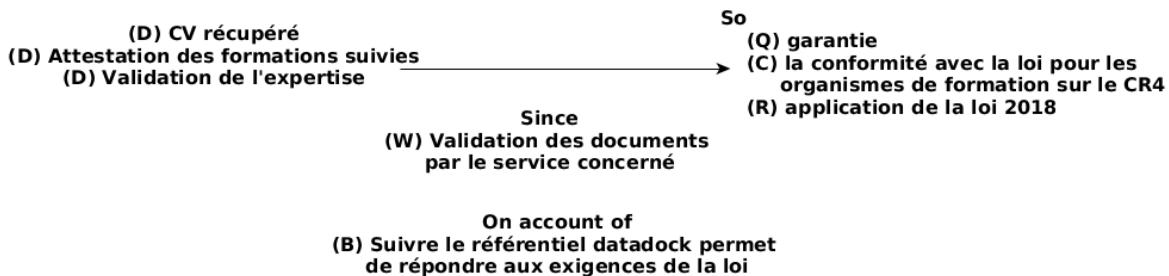


FIGURE 2.1 – Exemple de schéma argumentaire de Toulmin sur l'exemple fil rouge

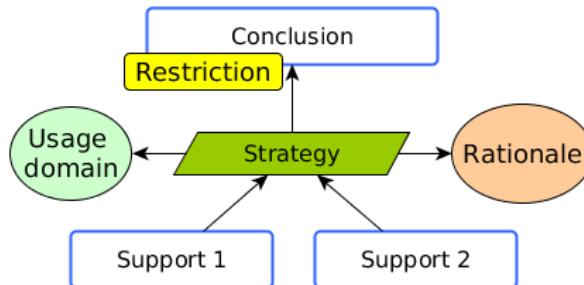


FIGURE 2.2 – Notation graphique d'un pas de justification inspirée de GSN

Exemple fil rouge 2.1.1 : Schéma de Toulmin sur Datadock

Si nous considérons l'exemple fil rouge en Figure 2.1. Nous savons que (D) "CV récupéré", "Attestation des formations suivies", "Validation de l'expertise". Nous nous focalisons alors sur l'argument si (D) est vrai alors on peut conclure que (C) "la conformité avec la loi pour les organismes de formation sur le CR4" est atteinte. Cette inférence à garantir (Q) est basée sur la justification (W) "validation des documents par le service concerné" qui est sous couvert de (B) "suivre le référentiel Datadock permet de répondre aux exigences de la loi". Nous pouvons ajouter que cette conclusion a une portée de validité (R) "application de la loi 2018" qui restreint cette justification au cadre légal de l'année courante.

Nous notons l'absence d'une notation graphique permettant une représentation plus compréhensible. Le modèle de Toulmin nécessite d'être raffiné pour avoir un modèle plus proche des activités de justifications, ce que vont nous apporter partiellement les **Diagrammes de Justification (DJs)**.

2.1.2 Diagramme de Justification (DJ)

Les **Diagrammes de Justification (DJs)** [POLACSEK \[2016\]](#) sont des diagrammes conceptuels permettant d'exprimer comment à partir d'un ensemble de faits on peut déduire une conclusion. Nous ne sommes pas ici dans le monde de la logique formelle, mais dans celui de l'explication. En effet, l'usage des diagrammes de justification est motivé par l'impossibilité d'être dans un monde formel. Ils capturent de bonnes pratiques, un pas de raisonnement accepté entre faits et conséquences, il relève d'une "*bonne rhétorique*" comme qualifiée par [PERELMAN et OLBCREHTS-TYTECA \[1958\]](#). Dérivée des travaux de l'argumentation légale de Toulmin, leur représentation graphique s'inspire de la Goal Structured Notation [KELLY et WEAVER \[2004a\]](#)¹. Dans la Figure 2.2, nous visualisons cette notation à travers le plus petit élément argumentaire valide : un pas de justification. Ces

1. Nous présentons ce concept dans la section 2.2.1

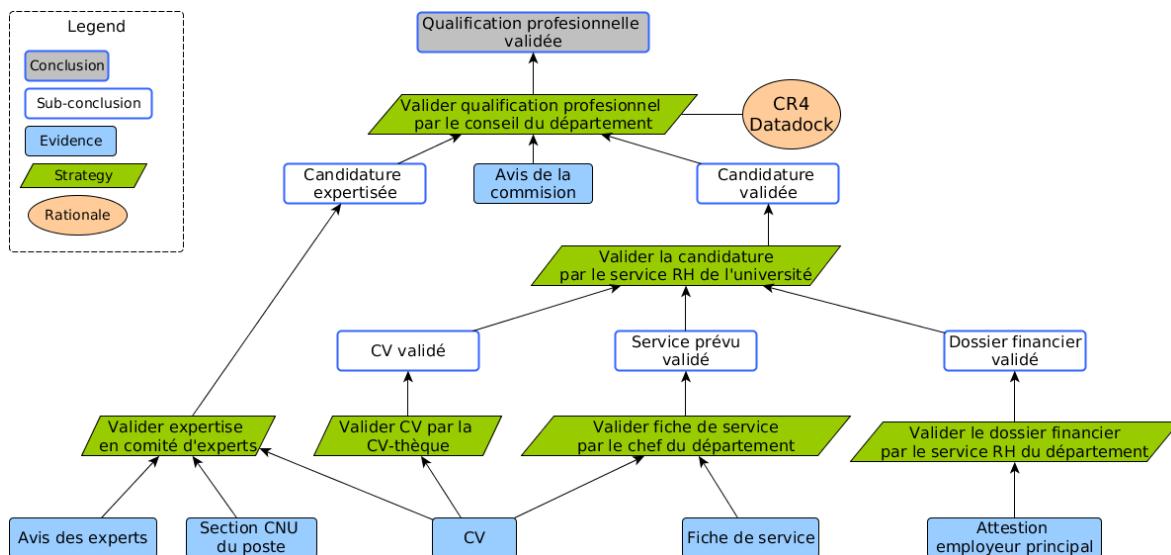


FIGURE 2.3 – Exemple de **DJ** associé au CR4 de Datadock dans le cadre du recrutement d'un vacataire

pas peuvent être chaînés pour ainsi obtenir un **DJ**. En haut, il y a une *conclusion* et, en bas des *évidences* qui amènent à cette conclusion. En fait, un **DJ** est une juxtaposition de pas de raisonnement où chaque pas est la transition de *supports* vers une *conclusion*.

Les principaux concepts des diagrammes de justification sont :

- *Conclusion* explicite ce qui est soutenu par la justification. C'est la conclusion du raisonnement, ce que nous cherchons à justifier.
- *Support* est soit une *évidence* i.e. un fait, une donnée, une hypothèse, etc, soit une *sous-conclusion* i.e. une conclusion d'une autre justification qui vient ici contribuer en tant que support. Ce sont les éléments sur lesquels s'appuie la justification de la conclusion.
- *Stratégie* correspond à la méthode utilisée pour établir la connexion entre les supports et la conclusion.
- *Limitation* est une restriction à la conclusion. Elles sont séparées de la conclusion, car elles n'ont vocation qu'à disparaître soit dans une justification de plus haut niveau, soit plus tard si les diagrammes s'inscrivent dans un processus temporel.
- *Fondement* est une explication de pourquoi une Stratégie est applicable. Si une stratégie consiste à suivre un processus défini par une norme, alors la Stratégie est l'application de la norme et le Fondement est l'explication de pourquoi la norme est applicable.
- *Domaine d'usage* donne les conditions précises d'usage et les limitations de la Stratégie. Si nous prenons l'exemple de l'application d'une norme, alors le domaine d'usage décrit dans quel contexte et pour quel besoin cette norme est applicable.

Exemple fil rouge 2.1.2 : Diagramme de justification sur Datadock

Un exemple concret d'utilisation de **DJ** est donné en Figure 2.3. Dans cet exemple, nous détaillons comment justifier du critère réglementaire n°4 mais appliqué seulement à la qualification des vacataires. Il existe donc autant de **DJs** différents que de types de personnels (e.g., permanent, ATER), mais qui atteignent la conclusion "*Qualification professionnelle validée*". Dans le diagramme pour les vacataires, cette conclusion est atteinte grâce à la stratégie "*Valider qualification professionnelle par le conseil du département*". Ce conseil émet un avis représenté par l'évidence "*Avis*

de la commission" en se basant sur les supports "*Candidature expertisée*" et "*Candidature validée*". Ces supports sont eux-mêmes justifiés par l'évaluation des aspects financiers, de l'expertise, du service prévisionnel par différents services du département ou de l'université. Il est à noter que la stratégie "Valider CV par la CV-thèque" est une validation par un outil qui permet de vérifier que le CV respecte certaines contraintes (e.g., limite de nombre de pages, en français). Nous voyons bien ainsi à travers cet exemple la différenciation au sein d'un même diagramme des activités de justifications humaines et compositionnelles.

Nous retenons de Toulmin et DJ les propriétés suivantes au regard nos objectifs :

- \mathcal{P}_1 : Un cadre sémantique clair reposant sur des concepts clés comme *stratégie* permettant ainsi de capturer et d'expliquer une argumentation voire des exigences de justifications.
- \mathcal{P}_2 : Une notation graphique permettant une meilleure représentation notamment dans l'enchaînement des exigences de justifications par le mécanisme de pas de justifications.

2.2 Cas d'assurance structuré

2.2.1 GSN et CAE

Un cas d'assurance structuré est un argument structuré, supporté par un ensemble d'évidences, permettant de donner un cadre compréhensif et valide de sûreté et d'applicabilité d'un système dans un environnement donné WEINSTOCK et GOODENOUGH [2009]. Différentes modélisations concrètes coexistent comme *Goal-oriented Structured Notation* (GSN) KELLY et WEAVER [2004a] ou *Claim-Argument-Evidence* (CAE) EMMET et CLELAND [2002]. Ces deux modèles visent à expliciter la satisfaction de propriétés sur la sûreté d'un système. GSN et CAE bien que basés sur le modèle de Toulmin abordent celui-ci en rendant optionnel l'expression de la stratégie qui explicite le passage des supports de l'argumentation à la conclusion. Une partie du raisonnement est ainsi perdue alors que c'est essentiel à la justification. De plus, il est difficile d'utiliser ces modèles dans des cadres différents des propriétés de sûreté. C'est pour ces raisons qu'il est apparu crucial pour la communauté d'élever d'un niveau d'abstraction supplémentaire pour ainsi capturer un cadre générique pour les cas d'assurance structurés à travers les *Structured Assurance Case Metamodel* (SACM).

Exemple fil rouge 2.2.1 : GSN sur Datadock

En Figure 2.4, nous reprenons l'exemple fil rouge avec un diagramme justifiant de la qualification professionnelle d'un vacataire. Nous retrouvons les mêmes éléments que dans la Figure 2.3. Ici, les *conclusions* et *sous-conclusions* sont des *goals*, les *supports* sont des *justifications* et des *solutions*, le *fondement* et le *domaine d'usage* sont le *contexte*. Les *stratégies* sont des concepts identiques, mais certaines sont omises ici pour illustrer cet aspect.

Au-delà de la *stratégie*, si nous comparons DJ et GSN sur les deux exemples en Figure 2.3 et Figure 2.4, nous notons que DJ et GSN sont très proches, mais divergent sur les feuilles de leur diagramme. DJ suppose qu'un processus a été identifié pour les activités de justifications et se base sur les évidences produites pendant ce processus. GSN lui aide à identifier les activités du processus liées à l'atteinte de l'objectif

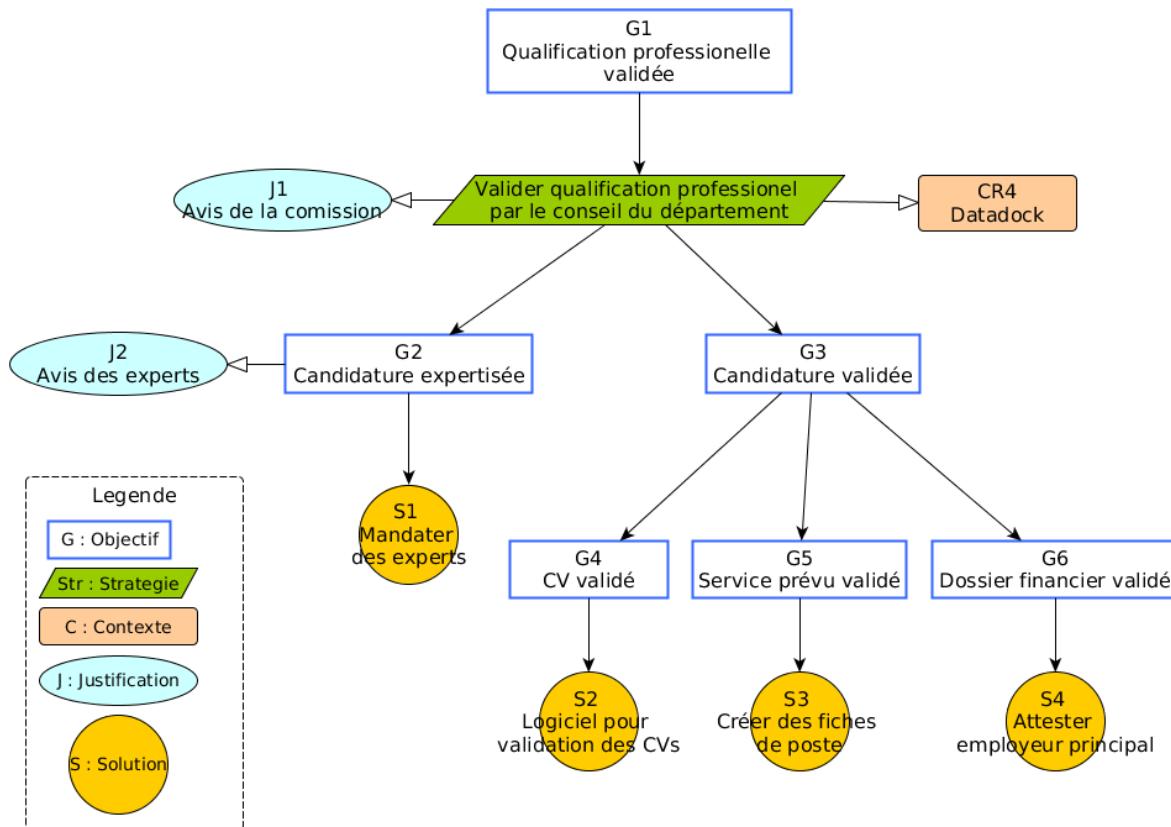


FIGURE 2.4 – Exemple de diagramme GSN associé au CR4 de Datadock dans le cadre du recrutement d'un vacataire

par les *solutions*. En ce point, **DJ** est bien positionné *a posteriori* tandis que GSN est *en a priori*.

Nous en retenons une propriété déjà amenée dans la section précédente \mathcal{P}_2 puisque la notation graphique des **DJs** vient de *GSN*.

2.2.2 SACM

Un des standards pour la modélisation de justification, appelée *Structured Assurance Case Metamodel* (SACM) (OMG) [2013], est une norme portée par l'OMG² qui cherche à établir le méta-modèle des cas d'assurance structurés. Le méta-modèle proposé par *SACM* repose sur 3 aspects distincts : le modèle *argumentaire* détaillé en Figure 2.5, le modèle d'*artefacts* et le modèle de *cas d'assurance*. Le modèle *argumentaire* définit une logique d'assertions permettant de représenter des arguments liés entre eux par des liens typés permettant ainsi d'établir des inférences entre arguments comme l'ajout d'un contexte applicatif à un argument. Le modèle d'*artefacts* permet de formaliser des données factuelles représentées comme des artefacts (événement, participant, technique, ressource, etc.). Ces artefacts sont les éléments basiques sur lesquels des raisonnements peuvent être faits et ainsi mener à une argumentation. Pour finir, le modèle de *cas d'assurance* permet de décorer, enrichir le modèle argumentaire en ajoutant des notes, des contraintes, la gestion de la langue. Notons que le modèle *argumentaire* repose sur des concepts particulièrement intéressants. Il supporte le typage fort des relations entre assertions à travers les concepts dérivés de AssertionRelationship mais il supporte également une forme d'abstraction en raisonnant sur le niveau au dessus des Assertion, les Claim. En

2. Object Management Group (OMG) est un comité de normalisation technologique international

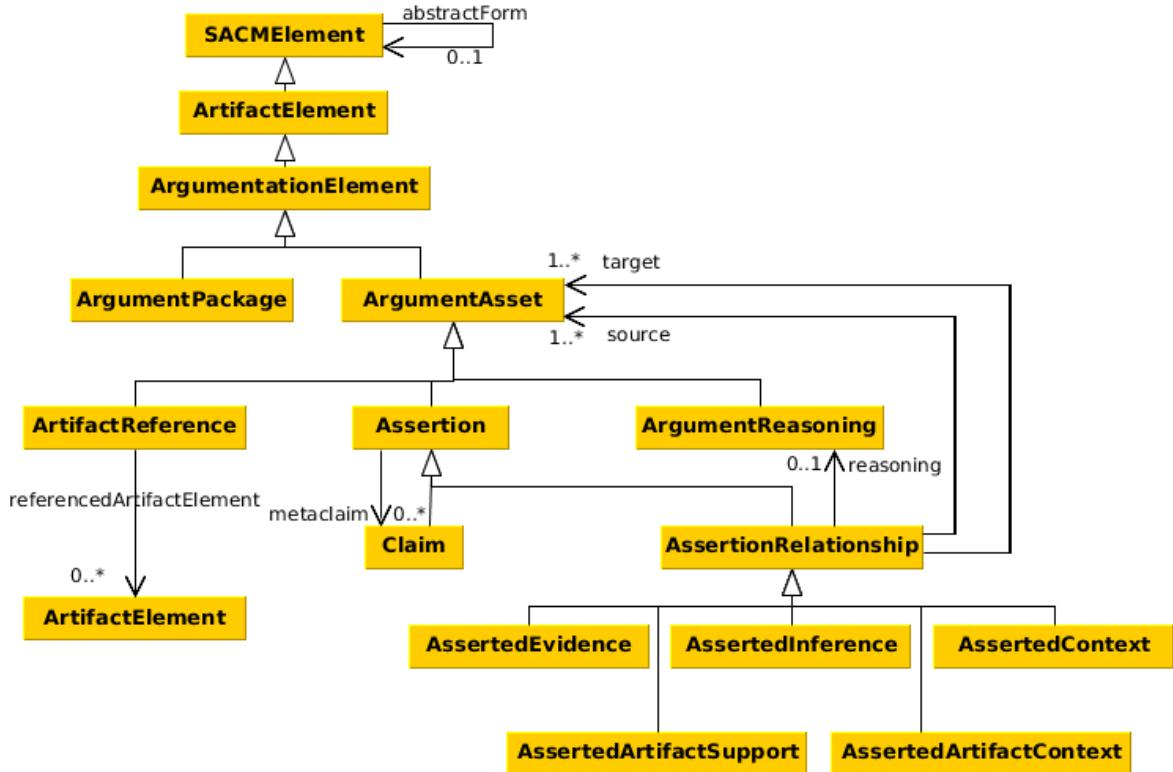


FIGURE 2.5 – Extrait du méta-modèle de SACM (partie argumentation)

ces points, ce modèle *argumentaire* supporte toute la richesse de l'expression des arguments présente dans *GSN* et *CAE*. Ce canevas permet ainsi de définir toutes les couches pour un cadriel³ d'exigences pour la qualité, des faits jusqu'au raisonnement menant à une revendication en passant par la couche de présentation de ce raisonnement. Ce méta-modèle a été utilisé dans diverses approches par la communauté pour en définir un langage utilisable à travers Eclipse pour éditer et visualiser des diagrammes basés sur *GSN* MATSUNO [2014] ou concevoir des exigences pour la certification DE LA VARA et PANESAR-WALAWEGE [2013].

Nous retenons de *SACM* une propriété fondamentale que cette approche apporte :

\mathcal{P}_3 : Une modélisation favorisant les interactions entre notations

2.3 Organisation des connaissances

Dans le domaine de l'organisation des connaissances, un nombre conséquent de méthodologies basées sur l'argumentation existent. Elles ont toutes la même origine : *Issue-Based Information Systems* (*IBIS*) KUNZ et RITTEL [1970]. *IBIS* a été conçu pour supporter et documenter des processus de décision, mais est maintenant utilisé pour organiser la connaissance et les justifications. KIRSCHNER et collab. [2003] présente un ensemble de cas d'études et analyse comment *IBIS* est utilisé dans la pratique. Toutes ces notations provenant d'*IBIS* ont comme point commun qu'elles tendent à capturer le raisonnement derrière une décision prise durant un processus décisionnel, mais également les alternatives qui ont finalement échoué. Par exemple, l'approche *Questions, Options and Criteria* (*QOC*) MACLEAN et collab. [1991] identifie des *questionnements sur la conception* avec des questions menant à des réponses et potentiellement à l'identification d'alternatives.

3. *framework* en anglais c'est-à-dire un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations

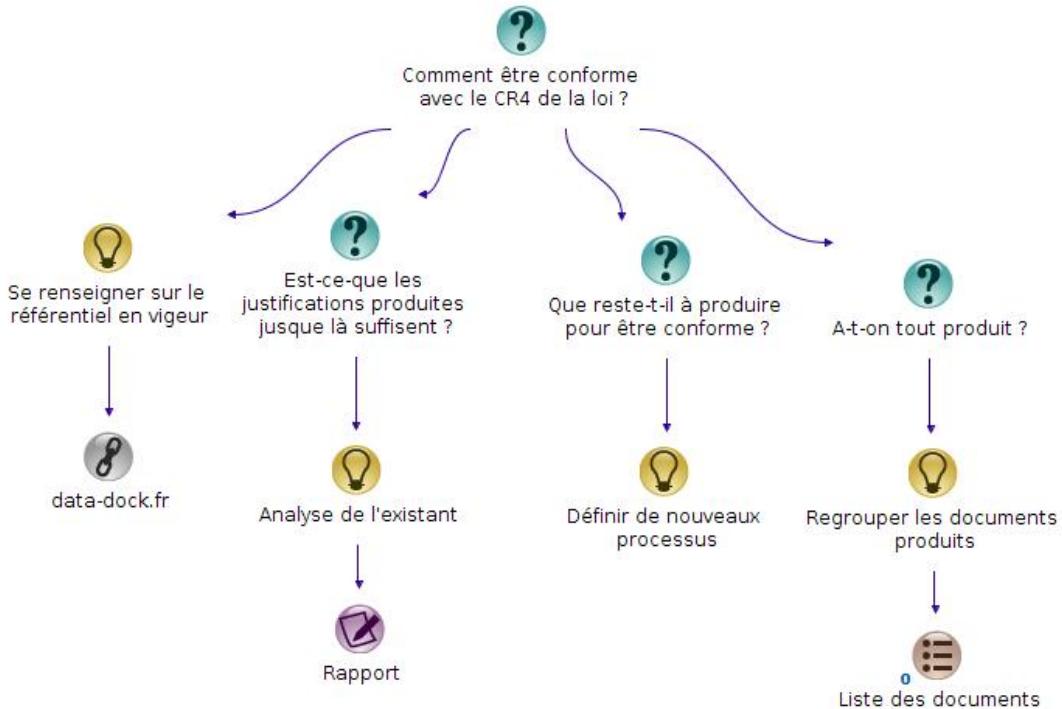


FIGURE 2.6 – IBIS sur Datadock

De plus, *QOC* possède un critère d'évaluation basé sur les besoins et propriétés attendus. Avec ces critères, il est possible de classer les différentes options.

Exemple fil rouge 2.3.1 : IBIS sur Datadock

Dans le cadre Datadock, un exemple est donné en Figure 2.6. La *Question* que l'on se pose est "*Comment être conforme avec le CR4 de la loi ?*". Pour ce faire nous allons dériver cette question vers des questions plus précises et des moyens de réponses jusqu'aux artefacts de ces réponses eux-mêmes (e.g., notes, listings, prise de décision, références). Répondre à la question de haut niveau pour Datadock consiste donc à contextualiser pour l'IUT en i) se renseignant sur le référentiel en vigueur, ii) en analysant les justifications produites jusqu'ici, iii) en identifiant ce qu'il reste à produire, iv) en mettant en place de nouveaux processus, v) en vérifiant si tout est produit comme prévu. Cette approche permet ainsi à la fois d'aider dans la mise en place des activités liées à la question "*Comment être conforme avec le CR4 de la loi ?*" mais également de laisser une trace des questionnements et des décisions prises pour leurs mises en place.

Cette méthodologie permet de s'interroger et à travers ce guide d'aider à argumenter sur les prises de décisions. Cela permet donc de se questionner sur une argumentation dans le but de l'évaluer. Néanmoins, ce processus nécessite une intervention humaine.

Cette méthodologie donne plus une explication sur la liste des tâches à effectuer que sur les exigences de justification. En particulier, les stratégies se perdent dans le flot et les relations entre conclusions intermédiaires sont difficiles à appréhender.

Pour conclure, les approches *SACM* comme *DJ* permettent bien de capturer des exi-

gences de justifications (O_1 : Capturer des exigences de justifications). Ces approches basées sur un cadre sémantique fort permettent d'assurer que la construction des exigences de justifications est conforme d'un point de vue des concepts voire du métamodèle pour SACM. Ceci permet de vérifier des propriétés de conformité, mais uniquement par construction (O_3 : Vérifier et analyser la conformité entre exigences et production des justifications).

Néanmoins, ces approches ne se positionnent pas comme un support pour la production des justifications à travers les diagrammes énoncés par ces approches. Ces méthodes proposent seulement une abstraction sur les éléments de justifications à produire c'est-à-dire la capture d'exigences de justifications. Il en découle que le besoin O_2 : Guider la production des justifications ne peut pas être couvert par ces approches. Il va donc s'agir dans le chapitre suivant d'identifier les processus et activités en jeu pour la justification et ainsi préciser comment la production les réalise.

Chapitre 3

Processus de justification

Sommaire

3.1 Processus de certification	20
3.1.1 Principe	20
3.1.2 Spécialisation par domaine	21
3.2 De l'ingénierie des exigences à l'élicitation d'exigences de justifications	22
3.2.1 KAOS	22
3.2.2 I*	23
3.3 Production des justifications	25
3.3.1 Traçabilité	25
3.3.2 Intégration Continue	27

Les modèles de justifications énoncées dans le chapitre précédent ne se suffisent pas à eux-mêmes. En face, il est nécessaire d'avoir le processus permettant de guider la production des exigences de justifications ainsi que la production des justifications elle-même. En ce sens, des processus sont mis en place permettant de structurer la mise en production des justifications. Les activités de ces processus doivent être effectuées en parallèle de celles du processus de développement de l'entreprise. Chaque activité lors du développement contribue à produire des artefacts de justification qui permettent ensuite de les utiliser dans le processus de justification.

3.1 Processus de certification

3.1.1 Principe

Comme le montre la Figure 3.1, pendant l'ensemble de la durée de vie d'un projet, l'autorité de régulation intervient pour formuler des normes (1) et assurer un suivi réglementaire du projet. En début de projet, elle s'assure que les pratiques internes dérivées des exigences normatives qu'il a identifiées sont valides dans le cadre du projet visé (4). Durant le développement, elle assure le suivi du projet pour vérifier que ses pratiques internes sont respectées avec l'appui des documents attestant le bon déroulement de chaque étape du projet (7). En fin de développement, en vue d'une mise sur le marché, le dossier complet du produit est examiné avec à la clé des demandes de compléments et/ou un audit de certification (10).

AKHIGBE et collab. [2015] identifie quatre tâches essentielles dans le processus de certification : la modélisation, la vérification, l'analyse et la promulgation. La *modélisation de la conformité* consiste à découvrir et formaliser à partir des textes réglementaires quel niveau de conformité est nécessaire. Dans la Figure 3.1, cela correspond aux activités de contextualisation des normes et de production des pratiques internes (2,3). La *vérification de conformité* consiste à assurer que la modélisation proposée capture bien les exigences de l'autorité. L'activité de revue (4) permet de matérialiser ces vérifications. L'*analyse de la conformité* consiste à monitorer la concrétisation de cette modélisation par les justifications concrètes au sein de l'organisation. Ceci est en relation avec le support des pratiques internes pour le développement et les différents audits qui attestent que la production des justifications est en adéquation avec les exigences de justifications (5, 7, 10). La *promulgation de la conformité* consiste à faire évoluer les pratiques dues à des changements organisationnels ou normatifs. Cela correspond à la réaction en chaîne due aux activités de production des normes ou de production des pratiques internes (2, 3).

Exemple fil rouge 3.1.1 : Processus de certification sur Datadock

Si nous reprenons l'exemple de Datadock, le passage de RE.S.E.A.U. à ce nouveau référentiel nécessite de changer de norme et donc de faire évoluer les pratiques internes pour assurer la conformité avec Datadock (*promulgation de la conformité*). Ce changement amène à redéfinir et structurer les exigences de justifications (*modélisation de la conformité*) et vérifier que la capture ainsi proposée permet bien de répondre aux attentes de la norme (*vérification de conformité*). Il vient ensuite le temps où les justifications pour chaque exigence sont produites et où est analysé que la conformité est bien atteinte (*analyse de la conformité*).

Si nous regardons plus à quels niveaux se situent chacun de nos objectifs au regard de ces travaux, nous notons que **O₁ : Capturer des exigences de justifications** se situe sur les tâches de *modélisation de la conformité* et *vérification de la conformité* tandis que

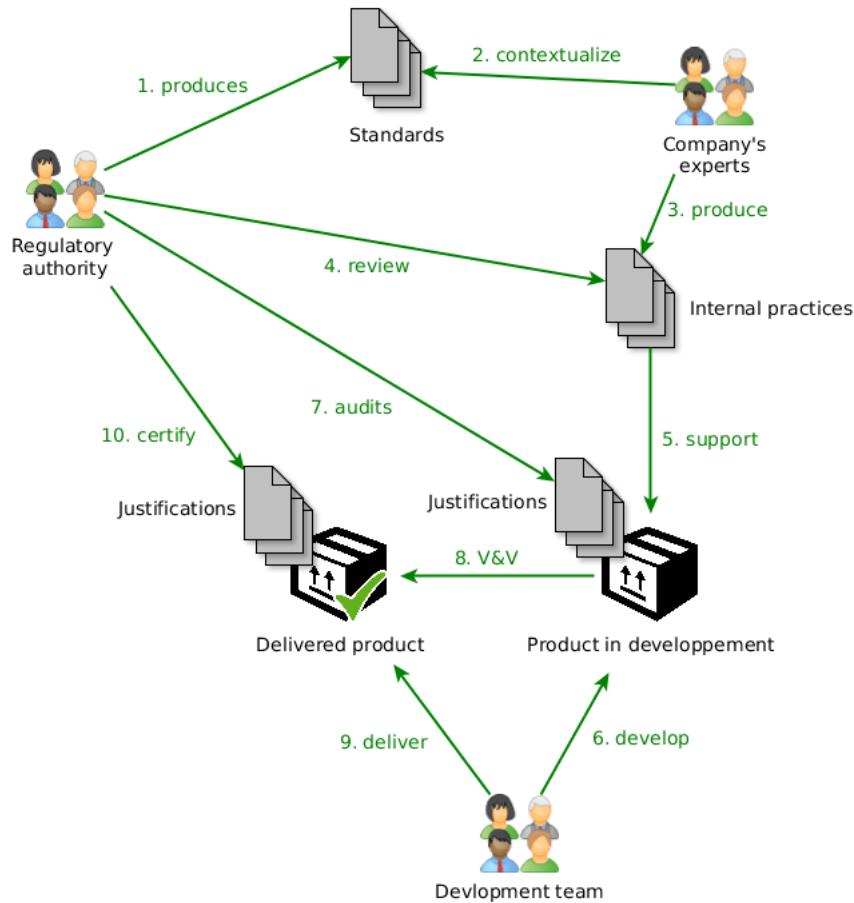


FIGURE 3.1 – Processus de certification

O₂ : Guider la production des justifications et **O₃** : Vérifier et analyser la conformité entre exigences et production des justifications sur les tâches d'*analyse de la conformité*. Les tâches de *promulgation de la conformité* sont un aspect transverse à nos objectifs qui sera précisé lors du raffinement de ceux-ci.

3.1.2 Spécialisation par domaine

Ce processus haut niveau décrit seulement les grandes interactions avec l'autorité de régulation. Pour chaque domaine, les normes sont différentes et, a fortiori, les exigences à respecter aussi. Le contexte de l'entreprise rajoute de la variabilité dans ce processus qui ne peut définitivement pas être générique. Certaines approches ont permis de monter en abstraction sur des domaines et contextes précis.

Par exemple, dans le cadre du développement Agile de logiciel dans le domaine du médical, l'*Association for the Advanced of Medical Instrumentation (AAMI)* a dérivé les points clés de la norme ISO/IEC 62304 pour proposer un cycle incrémental valide dans son guide TIR-45 [AAMI \[2012\]](#). Ce cycle reprend les grandes activités de la norme et lève la croyance que ce domaine critique ne permet pas l'adoption des pratiques Agile. Dans le domaine de l'avionique, Holloway propose d'expliquer la DO-178C - *Software considerations in airborne systems and equipment certification* à travers des diagrammes GSN [HOLLOWAY \[2013\]](#). En effet, cette norme repose sur des éléments implicites où chaque entreprise devait proposer et défendre auprès l'autorité de certification sa propre interprétation. Holloway explicite l'implicite à travers ces diagrammes et propose de les utiliser comme une des pierres angulaires pour la discussion entre experts et l'évolution de la DO-178.

Nous pouvons également citer la tentative de l'OTAN pour généraliser la mise en place des processus de certification dans le cadre des activités de V&V **PDG [2010]**. Pour ce faire les acteurs des domaines critiques ont été réunis permettant ainsi de généraliser cette problématique commune. L'OTAN énonce ainsi une méthodologie générale pour assurer la qualité des modélisations, des simulations et des données.

L'ensemble de ces approches bien que très spécifiques montre bien le besoin des entreprises dans les domaines critiques d'abstraire leurs cas individuels pour raisonner à un niveau plus général dans le but de définir des approches globales à un domaine **HOLLOWAY [2013]**, **AAMI [2012]** voire indépendantes **PDG [2010]**.

Nous retenons ici les propriétés suivantes :

P₄ : Le raffinement d'exigences de justifications haut-niveau vers le moyen de maîtrise de celles-ci permet d'expliciter la contextualisation des normes

P₅ : La capitalisation sur des pratiques existantes en les structurant permet de mieux les comprendre, les discuter et les réutiliser

Pour mener ce processus de certification, on distingue bien dans la Figure 3.1, des activités liées aux exigences de justifications pour la certification (2,3), des activités liées à la production de ces justifications (6,8,9) et des activités de vérification de la conformité (4,7,10). Si nous nous intéressons aux activités d'explicitation des exigences de justifications, il est intéressant d'étudier un domaine plus général : l'ingénierie des exigences. En effet, il est intéressant d'identifier si cette problématique est prise en compte par ce domaine, mais également quelle est sa relation aux exigences d'un produit en général. Nous proposons cette étude en Section 3.2. Par ailleurs, pour les activités de production des justifications, il est intéressant d'étudier par quels mécanismes cette production est assurée. Nous étudions en Section 3.3, les avancées scientifiques sur les préoccupations de traçabilité, mais également sur l'aspect continue de ces activités en étudiant les aspects d'intégration continue.

3.2 De l'ingénierie des exigences à l'élicitation d'exigences de justifications

Le domaine de l'ingénierie des exigences vise à "*eliciting individual stakeholder requirements and needs and developing them into detailed, agreed requirements documented and specified in such a way that they can serve as the basis for all other system development activities*" **POHL [2010]**. Il y est différencié les exigences fonctionnelles (e.g., les fonctionnalités d'un produit) des exigences non fonctionnelles (e.g., qualité de service, politique de sécurité) **CHUNG et collab. [2012]**. Dans le cadre de nos travaux, nous nous intéressons aux aspects qualité d'un produit et sommes donc dans le cadre des exigences non fonctionnelles, mais nous appuyons sur les exigences fonctionnelles également (e.g., L'atteinte de l'exigence fonctionnelle portant sur la portance d'une aile d'avion permettant d'aider à atteindre la sûreté en vol qui est une exigence non fonctionnelle) . Nous allons nous intéresser à deux notations, i* et KAOS, permettant de représenter de telles exigences.

3.2.1 KAOS

L'approche *Knowledge Acquisition in automated specification*, a.k.a *Keep All Objectives Satisfied* (KAOS) **VAN LAMSWEERDE [2009]** est une méthodologie permettant de représenter des exigences logicielles par la capture d'objectifs. Cette approche a permis d'initier des travaux sur la représentation graphique de ces objectifs **HEAVEN et FINKELSTEIN [2004]**, mais également à l'intégration forte avec des méthodes formelles pour le rapprochement entre l'expression d'objectifs et l'expression de spécifications **MATOUSSI**

et collab. [2008]. Cette approche repose sur le concept de Goal, permettant de capturer des exigences fonctionnelles, qui, par raffinement, permettent de préciser les moyens d'atteinte et les freins à cet objectif. Ces objectifs peuvent être qualifiés suivant quatre patrons : *Maintain* qui demande qu'une certaine propriété soit toujours atteinte, *Avoid* qui demande qu'une certaine propriété ne soit jamais atteinte, *Achieve* qui demande qu'une certaine propriété soit éventuellement atteinte et *Cease* qui demande qu'une certaine propriété soit éventuellement atteinte. Le raffinement d'un objectif se matérialise à travers le lien avec les concepts de Sub-Goal, Conflict, Expectation ou Obstacle. Ces concepts recoupent ainsi les patrons d'objectifs.

Pour capturer les exigences non fonctionnelles, le concept de Softgoal a été ajouté. Néanmoins, il est peu formalisé et peu utilisé par la communauté. Preuve en est, un métamodèle capturant les concepts essentiels de KAOS en se basant sur 6 métamodèles existants n'analyse pas ce concept et donc de ne l'intègre pas dans sa modélisation **NWOKEJI et collab.** [2013]. L'analyse de **MATULEVICIUS et HEYMANS** [2005] met d'ailleurs en évidence qu'une recherche autour des Softgoal doit être menée pour mieux les caractériser. Par exemple, il n'est pas caractérisé si ces Softgoal sont régis par les mêmes patrons que les Goal. Néanmoins, il existe plusieurs travaux visant à utiliser KAOS pour capturer des exigences non fonctionnelles comme l'incertitude environnementale **AHMAD et collab.** [2012]. Dès lors, avec une capture plus formelle des exigences non fonctionnelles, il serait possible de représenter des propriétés portant sur la qualité et donc a fortiori de les justifier par cette approche.

3.2.2 I*

I* est un langage qui propose de capturer les besoins très en amont depuis l'intérêt des parties prenantes jusqu'aux différentes variantes possibles du système à réaliser **YU** [1997]. Dans la nouvelle version i* **DALPIAZ et collab.** [2016], un des changements notables est le remplacement des "soft-goal", qui permettaient de capturer des exigences non fonctionnelles, par le concept de "quality". Cette évolution ouvre la porte à une utilisation de i* dans le domaine des exigences qualité. Par exemple, dans la méthodologie *Security Quality Requirements Engineering* (SQUARE) **MEAD et STEHNEY** [2005], cela permet la capture des exigences portant sur la sécurité.

Exemple fil rouge 3.2.1 : Diagramme i* sur la qualification du personnel

Dans notre exemple fil rouge, le diagramme i* en Figure 3.3 comment atteindre l'objectif de "Attestation de qualification du personnel" et la qualité "critère réglementaire n°4", qui correspond au critère énoncé dans la Section 1.4. Dans ce diagramme nous retrouvons donc trois tâches qui nécessitent des ressources internes à l'université pour atteindre l'objectif. Ces tâches produisent des documents permettent de "créer" la qualité "critère réglementaire n°4". À travers ce diagramme nous visualisons bien comment une exigence exprimée sous forme d'objectif et/ou de qualité est raffinée pour identifier les tâches et ressources concernées par leurs réalisations.

Dans la version 2.0 du langage **DALPIAZ et collab.** [2016], les auteurs proposent un métamodèle pour le langage. Le cœur de ce métamodèle, présenté en Figure 3.2, permet de capturer formellement les mécaniques du langage (e.g., abstraction de Goal et Task sous le même concept pour permettre une gestion des Refinement mutualisée, ContributionType capturé par une classe-association permettant une meilleure généralité pour la capture des liens *make*, *help*, *hurt*, *break* entre concepts du langage). Cette contribution formelle a ensuite permis d'enrichir le langage comme le propose **NGUYEN**

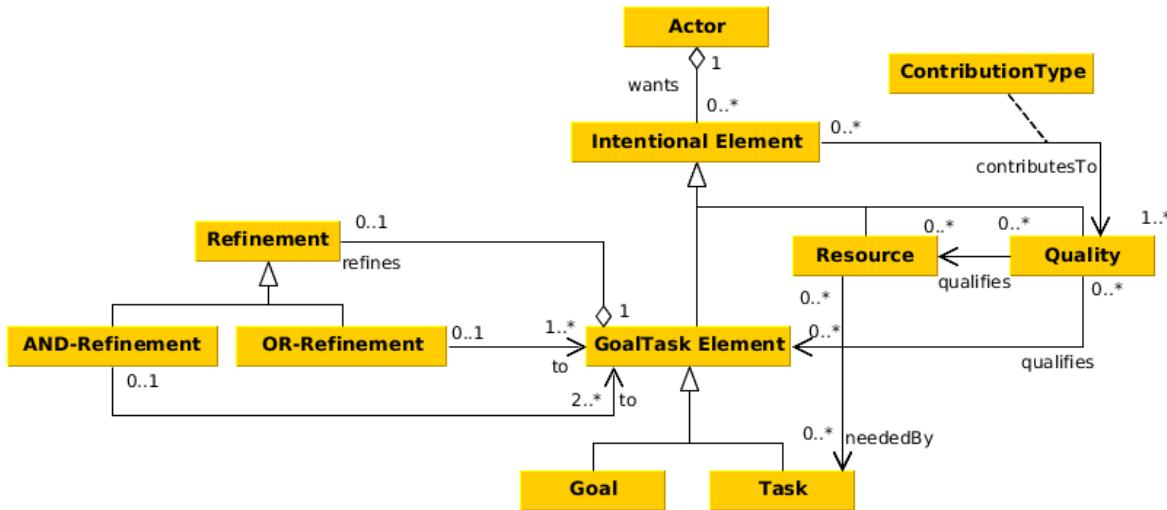


FIGURE 3.2 – Extrait du métamodèle d'i* 2.0

et collab. [2016] en introduisant un lien avec les pratiques internes d'une entreprise, mais aussi Massey MASSEY et collab. [2017] qui transforme ce langage en un langage pour exprimer les besoins d'une autorité de régulation.

Néanmoins, le langage i* reste très orienté processus, puisqu'il manipule principalement les notions d'acteurs, de ressources et de tâches. En ce sens, le langage est plus porté sur le *qui*, *quoi* et *comment* que sur le *pourquoi*. Le *pourquoi* est ce que l'on cherche à atteindre quand on veut justifier. Ainsi, i* permet d'exprimer ce qui contribue ou empêche l'atteinte d'une qualité, mais ne permet pas de justifier cette contribution en s'appuyant sur du raisonnement et des artefacts de qualité. Cependant, pour pallier cette limitation, VAN ZEE et collab. [2016] proposent d'attacher de l'argumentation pour expliciter le raisonnement d'un diagramme i*. Plus précisément, il propose de rajouter une représentation des arguments et contre-arguments proposés par les différentes parties prenantes qui se sont prononcées pour, ou contre, un élément du modèle exprimé en i*. Bien que prometteur, cette contribution relève plus de l'argumentation que de justifications au sens défini en Section 2. Par rapport à ces travaux, les modèles de justification se situent *a posteriori*, il n'y a plus de pour et de contre, mais une explication rationnelle du choix final en vue de le justifier.

Cette étude du domaine de l'ingénierie des exigences permet de mettre en lumière des moyens de structurer de façon formelle l'expression de besoins orientée objectif. L'explication de besoins est la pierre angulaire de tout développement. Ceux-ci sont ensuite raffinés en spécifications menant à une solution technique répondant à ces données d'entrée. Durant tout ce processus, les justifications quant au produit et à la façon de le produire découlent de cette explication. Néanmoins, ces approches sont très larges et introduisent que superficiellement les préoccupations d'exigences de justifications. Il apparaît tout de même intéressant d'être capable de faire un parallèle entre l'expression d'exigences par des méthodologies de l'ingénierie des exigences et leurs utilisations pour justifier. En effet, pour atteindre **O₂ : Guider la production des justifications**, il s'agit notamment d'être capable de construire une approche collaborative avec les domaines connexes de la justification, comme l'est l'ingénierie des exigences. Néanmoins comme le montrent AKHIGBE et collab. [2018] à travers une étude comparative de la littérature sur l'utilisation ou non de la modélisation d'objectifs appliquée aux exigences normatives, l'adoption de telles pratiques reste difficile à généraliser. En effet, cette étude montre que des méthodes non orientées objectifs restent une pratique courante pour capturer l'intention des régulations (*e.g.*, comprendre les lois, représenter leur signification, identifier

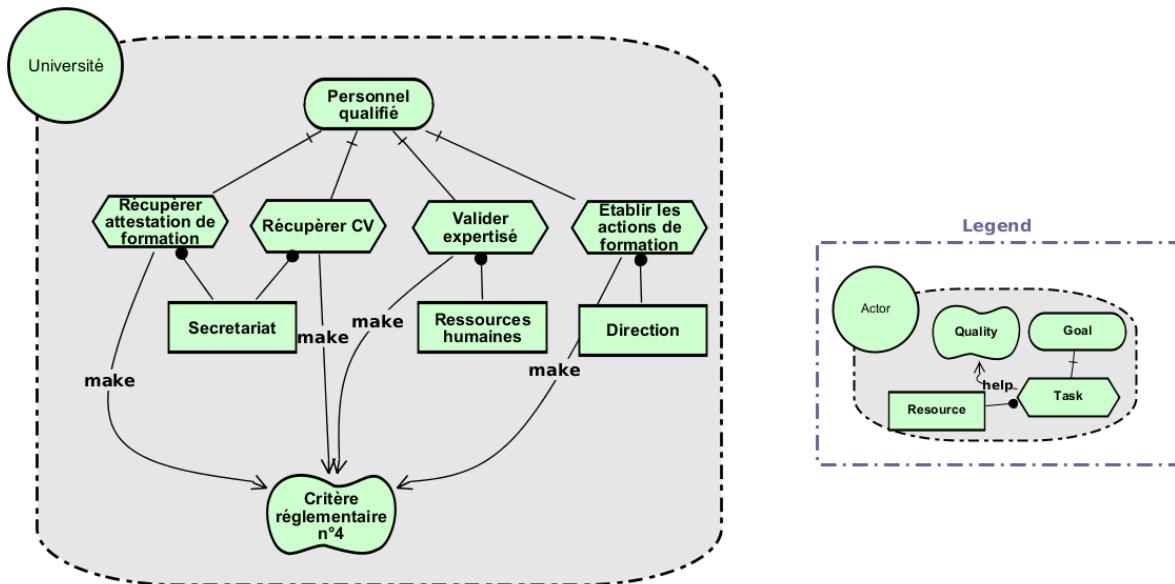


FIGURE 3.3 – Diagramme i* sur l'exemple de Datadock

des obligations et permissions). Une approche orientée objectif est utilisée principalement par les entreprises qui veulent structurer leurs activités autour des normes (*e.g.*, abstraire la sémantique de la loi, identifier les liens entre loi et processus, représenter les dépendances entre lois). Dans nos travaux, nous adressons le deuxième type d'organisation. En effet, nous cherchons à élucider des exigences de justifications jusqu'à la production de ces justifications tout en attestant la conformité entre elles. L'ensemble de ces activités est donc bien non pas intentionnel, mais structurant.

Nous retenons ainsi de KAOS et i*, deux nouvelles propriétés :

\mathcal{P}_6 : La capture conjointe des exigences fonctionnelles et exigences non fonctionnelles, en particulier relatif aux normes, permet une meilleure compréhension des relations qui les lient.

3.3 Production des justifications

3.3.1 Traçabilité

Au regard des pratiques couramment adoptées par les industries hautement régulées, la conformité avec des normes est principalement assurée par la traçabilité **REMPPEL et collab.** [2014]. Dans la pratique, les entreprises définissent des processus qui suivent des préconisations reconnues. La conformité avec ces processus assure, de facto, la conformité avec les normes. Les moyens de conformité sont donc basés sur la preuve que ces processus ont été suivis, que ce qui a été fait est consistant avec ce qui aurait dû être fait. Ces preuves sont donc assurées par traçabilité. Cette traçabilité se matérialise à plusieurs niveaux : organisationnel et stratégique (*e.g.*, suivi de projet, spécifications, décisions critiques), mais également technique (*e.g.*, décisions architecturales, protocoles de tests).

AREL

Un exemple de traçabilité technique nous vient de *Architecture Rationale and Element Linkage* (AREL) **TANG et collab.** [2007] qui est un cadriel pour l'architecture logicielle permettant de tracer des raisonnements sur les choix architecturaux. Ce cadriel propose d'enrichir des modèles *Unified Modeling Language* (UML) **BOOCH** [2005]

pour tracer ces prises de décisions techniques. Depuis des *raisons motivationnelles* initiales jusqu'au *résultat de conception*, il permet de tracer les décisions clés prises durant la phase de conception. Ces décisions architecturales sont classifiées en plusieurs catégories (e.g., Spécifications fonctionnelles, Environnement technologique, Environnement métier) permettant de mieux identifier l'origine du raisonnement. Tracer ses décisions consiste alors à enrichir un modèle *UML* existant en étiquetant les éléments du modèle avec des éléments de décisions architecturales liés au raisonnement y menant. L'intégration de cette approche dans des modèles *UML* permet ainsi de tracer l'impact d'une décision sur l'architecture par parcours du modèle *UML*. De plus, un modèle *UML* peut être amené à évoluer et donc les décisions quant à sa conception aussi. Cette approche permet ainsi de mener à bien conjointement évolution de la conception et traçabilité des évolutions des décisions. Néanmoins, cette approche ne se confronte à aucune exigence de justifications, elle cherche juste à tracer un ensemble de décisions et les impacts sur l'architecture.

Echo

Un exemple de traçabilité stratégique nous vient de *Echo LEE et collab. [2003]*, un logiciel sous Eclipse permettant de tracer des besoins exprimés sous forme textuelle jusqu'aux tâches qui en découlent. Dans un contexte Agile, le processus de raffinement est long et ne mène pas toujours à une fonctionnalité implémentée. Il est alors important de distinguer ce qui est pertinent au milieu du bruit de cette traçabilité intempestive. Depuis le document initial exprimant les besoins, des extraits de phrase sont transformés en fonctionnalités qui sont ensuite raffinées en tâches. L'ensemble de ce raffinement est représenté sous forme de graphe permettant de visualiser les liens d'un extrait de phrase aux tâches. Le rétro parcours de celui-ci permet ainsi d'identifier de quel besoin provient un fonctionnalité ou même tâche. Par cette capture des liens du document initial aux tâches, il est ainsi possible de se concentrer sur les éléments de traçabilité qui sont pertinents : seulement ceux qui ont amené un développement. Au sens d'un graphe, ceci revient à identifier le graphe minimal menant à un sous-ensemble des puits. Dans cet environnement contrôlé, il est également possible de capturer l'évolution des besoins en conservant des versions de cette traçabilité et ainsi pouvoir comparer les graphes entre eux. Grâce à *Echo*, les artefacts ne répondant pas à un besoin réalisé sont facilement ignorables et ne sont conservés que ceux qui sont pertinents. Cette méthodologie permet d'élaguer des éléments tracés ne servant plus et ainsi d'aller dans le sens de \mathcal{O}_2 . En ce sens, cette approche vise l'efficience par le tri. Néanmoins, la séquence d'artefacts n'est pas liée à des exigences de justifications. Ici encore, c'est une méthodologie permettant de tracer seulement les éléments potentiellement utiles, mais cette approche ne cherche pas à atteindre l'atteinte d'exigences de plus haut niveau comme la justification (\mathcal{O}_1).

Exemple fil rouge 3.3.1 : Traçabilité avec Datadock

Dans le cadre de Datadock, une suite d'outils est fournie permettant aux organismes de formation de se déclarer, mais également de fournir l'ensemble des documents de preuve. De ce fait, Datadock lui-même constitue le *Système de Management de la Qualité* (SMQ). L'ensemble des preuves sont ainsi tracées.

Nous retenons ici la propriété suivante :

\mathcal{P}_7 : Intégrer la capture des prises de décision dans les chaînes d'outils des spécifications jusqu'au code permet une traçabilité efficiente

Toutes ces méthodologies et cadriels permettant de tracer certains aspects lors du développement d'un produit ce qui laisse place à un ensemble documentaire conséquent et peu structuré entre eux. Justifier consiste à structurer cet ensemble documentaire en un ensemble d'arguments basés sur ces documents. Il existe donc un fossé entre la production hétérogène des documents par ces méthodologies et la justification basée sur l'argumentaire. Ici aussi, il apparaît donc intéressant de se baser sur les approches de traçabilité pour **Guider la production des justifications** (\mathcal{O}_2).

3.3.2 Intégration Continue

L'intégration continue est une pratique interne, à destination des industriels, pour le développement de logiciels qui vise à fusionner régulièrement tous les artefacts logiciels ensemble **FOWLER** et **FOEMMEL** [2006]. L'intégration continue naît de l'inflation des activités de compilation, tests, déploiement due à l'introduction des pratiques Agile. Elle est la réponse aux problématiques de passage à l'échelle de ces méthodes au quotidien. L'intégration continue vise à l'automatisation totale du processus d'assemblage d'un logiciel. Plus précisément si nous prenons l'exemple des tests, "principle of continuous integration applies as well to testing, which should also be a continuous activity during the development process" **BOOCHE** [1991]. L'Agilité augmente ainsi l'importance et la quantité de tests. Les résultats de ces tests sont des éléments clés pour justifier de la qualité d'un produit. L'intégration continue propose seulement de construire automatiquement ces artefacts de justification. Cela va dans le sens de \mathcal{O}_2 (Guider la production des justifications), mais elle n'a pas vocation à les structurer \mathcal{O}_1 (Capturer des exigences de justifications). Il est tout de même possible de définir des seuils à atteindre (e.g., tous les tests doivent passer, la qualité du code doit avoir une bonne note, le déploiement doit être opérationnel). Ces seuils sont vérifiés en séquence et mènent à définir l'état du projet (e.g., stable, instable, en erreur), en ce sens cela remplit à son échelle \mathcal{O}_3 (Vérifier et analyser la conformité entre exigences et production des justifications).

Dans un contexte de certification, la masse des artefacts produits par l'intégration continue est utilisée comme un support des justifications tout au long de la vie d'un produit. Face à ce constat, il reste un aspect fondamental à prendre en compte : l'évolutivité et incrémentalité des justifications. En effet, à chaque cycle de développement, le nombre de tests et documents de justification qui sont nécessaires de produire augmente. Les éléments de justifications deviennent ainsi complexes à gérer et la confiance dans ce qui est produit est difficile à assurer tant faire le tri dans ce tsunami documentaire est une tâche complexe. Par ailleurs, que ce soit des normes ou des pratiques qui évoluent, ceci impacte sur les exigences de justifications qui doivent voir leur modélisation évoluer. Cette même évolution a alors un impact sur les justifications à produire. Il se pose alors des questions sur ce qu'il est pertinent de conserver et ce qui doit évoluer. Par ailleurs, la production des justifications est faite de façon incrémentale et continue. Au cours d'un projet, il est rare de produire l'ensemble des justifications dès le début du projet comme à la toute fin. Cette production est un processus incrémental qui vise à produire les justifications minimales à chaque grande étape d'un projet. Ce constat est généralisable, hors contexte Agile. Il est donc important de supporter des mécanismes permettant d'avoir confiance dans les justifications produites tout au long du processus de développement notamment de contrôler des mécanismes de consistance partielle des justifications par rapport aux exigences globales de justifications.

Exemple fil rouge 3.3.2 : Incrémentalité avec Datadock

Dans le cadre de Datadock, cette incrémentalité est également présente. En effet, le

recrutement des vacataires se fait de façon continue. De nouveaux vacataires sont recrutés au fil de l'année scolaire et, par conséquent, les justifications à produire également. Il faut donc être capable, à partir d'exigences de justifications de produire les justifications associées à chaque vacataire. Comme la plupart des activités de justifications sont faites par un humain, ceci pose une problématique de passage à l'échelle.

Ce parallèle entre intégration continue et justification met en évidence que l'évolutivité doit être une préoccupation transverse à l'ensemble de nos besoins. En effet, dans le cadre de O_1 : **Capturer des exigences de justifications**, il va falloir supporter l'évolution des exigences de justifications, dans O_2 : **Guider la production des justifications**, l'incrémentalité dans la production des justifications et dans O_3 : **Vérifier et analyser la conformité entre exigences et production des justifications**, des vérifications partielles en fonction de l'état d'avancement des justifications.

Nous retenons ainsi les propriétés suivantes :

- P_8 : L'automatisation de la production de justifications aide le passage à l'échelle
- P_9 : La production incrémentale des justifications est accompagnée de mécanismes de comparaison et vérification partielle de la conformité entre exigences de justifications et justifications produites
- P_{10} : L'évolutivité des exigences de justifications amène à aligner la production des justifications et doit être supporté par des mécanismes d'analyse pour aider ce processus complexe

Chapitre 4

Positionnement et objectifs

Sommaire

4.1 Espace couvert par l'état de l'art	30
4.2 Raffinement des objectifs de la contribution	31
4.2.1 \mathcal{O}_1 : Capturer des exigences de justifications	31
4.2.2 \mathcal{O}_2 : Guider la production des justifications	32
4.2.3 \mathcal{O}_3 : Vérifier et analyser la conformité entre exigences et production des justifications	33

4.1 Espace couvert par l'état de l'art

Nous proposons la Table 4.1 comme la synthèse de l'étude de l'état au regard de nos objectifs. Il est apparu naturel de baser nos travaux sur les DJ et non sur ses fondements plus théoriques comme Toulmin. SACM bien que complet est apparu durant nos travaux. À la naissance de nos travaux, seuls GSN, CAE et DJ co-existaient. Nous avons choisi de nous baser sur DJ. Il n'y a pas eu de justification rationnelle à ce propos, seulement une facilité de collaboration avec l'auteur de l'article fondateur sur les DJs. Néanmoins, si nous analysons ce choix a posteriori, comme les DJs, SACM est basé sur le schéma de Toulmin. Il existe donc des liens forts entre DJ and SACM. En effet, les deux approches visent à capturer le lien entre des faits et la revendication de propriétés en traçant le raisonnement sous-jacent. Néanmoins, si nous comparons la vision que porte SACM et DJ, il existe des différences fondamentales. L'héritage de SACM porté par GSN et CAE les amène à considérer la stratégie (appelé garantie dans Toulmin) comme un élément optionnel. À travers ce choix, un argument peut donc être inféré depuis d'autres arguments sans expliciter le raisonnement qui permet cette inférence. Ceci nous paraît être contraire à l'objectif de ce type de diagrammes : expliquer par la justification. De plus, le cœur du modèle de SACM : le modèle argumentaire, considère possible la multi cardinalité dans les sorties d'un argument. DJ revendique une sortie unique servant de conclusion. Notre analyse ici nous porte à penser qu'un raisonnement basé sur un ensemble de supports ne permet d'atteindre qu'une unique conclusion. En effet, s'il est nécessaire d'avoir plusieurs conclusions ou de couper une conclusion pour la réutiliser partiellement, cela veut dire qu'il y a un raisonnement qui cache des relations implicites qui nécessitent d'être dépliées et expliquées. Ce management des sorties multiples nous apparaît donc comme une faiblesse au regard de l'essence des justifications : expliquer un raisonnement. Ainsi SACM peut être considéré comme un modèle conceptuel pivot, mais qui, pour être utilisé, dans une approche plus formelle nécessite d'être raffiné. Ainsi le choix des DJs reste pertinent dans le cadre de nos travaux à l'heure actuelle.

L'ingénierie des exigences présente de nombreux atouts pour remonter à la raison d'un développement et donc des éléments de justifications : l'expression de besoins. Les paradigmes présentés ont l'avantage d'être formellement définis notamment à travers de métamodèles, mais n'abordent qu'en surface les problématiques de justification. Il apparaît néanmoins nécessaire de pouvoir interagir avec de tels formalismes dans le but \mathcal{O}_2 . De même la traçabilité reste le moyen final d'assurer la pérennité des justifications, mais ne raisonne pas au niveau des exigences de justifications et est trop centrée sur certaines technologies. Pour finir, l'intégration continue apporte, par l'automatisation, une approche pour faciliter le passage à l'échelle et l'incrémentalité dans la production des justifications \mathcal{O}_2 . Cette approche permet également d'obtenir des garanties à travers des vérifications que l'on peut assimiler à de la conformité portant sur la qualité et va donc dans le sens de \mathcal{O}_3 .

En conclusion, l'état de l'art sur la continuité entre l'élicitation des justifications et la production des justifications conformes à ces exigences reste un domaine encore vierge où peu de publications scientifiques sont diffusées. En analysant, un peu plus du côté industriel, il apparaît que c'est en partie dû aux enjeux industriels et financiers des domaines critiques où faire certifier un produit pour sa mise sur le marché reste un savoir-faire à part entière. Pourtant, nous voyons à travers l'état de l'art qu'une approche scientifique sur ce domaine peut être mutuellement bénéfique à la recherche et à l'industrie.

TABLEAU 4.1 – Matrice de comparaison de l'état de l'art

Défi ¹ / État de l'art		\mathcal{O}_1	\mathcal{O}_2	\mathcal{O}_3
Argumentation	Toulmin	Arguments en général	Non	Non
Organisation des connaissances	IBIS	Non	Non	Questionnement informel
Cas d'assurance	JD	Oui	Non	Par construction
	SACM	Oui	Non	Par construction
Ingénierie des exigences	KAOS	Capture des exigences non fonctionnelles	Non	Non
	i*	Qualifie le lien objectif / qualité	Non	Non
Intégration Continue		Non	Plugins exécutables	Build stable/instable
Traçabilité	AREL	Non	Décisions architecturales à travers UML	Restreint à UML
	ECHO	Non	Raffinement de spécifications	Restreint à Eclipse

\mathcal{O}_1 : Capturer des exigences de justifications , \mathcal{O}_2 : Guider la production des justifications , \mathcal{O}_3 : Vérifier et analyser la conformité entre exigences et production des justifications

4.2 Raffinement des objectifs de la contribution

Même si, dans le cadre de la certification, l'usage de diagrammes de justification est prometteur, il faut tout de même souligner qu'à ce jour, il ne s'agit que d'un système de notation. S'ils permettent de clarifier les différents concepts nécessaires à une justification, ils ne sont associés à aucun outil automatique et, par conséquent, toutes les opérations de vérification doivent être réalisées par un être humain. Avoir une définition formelle de propriétés souhaitables permettrait de disposer d'un outil automatique d'aide à la création et à la relecture de diagrammes de justification. Nous allons ainsi ici raffiner, au regard de cet état de l'art, les objectifs de la Section 1.2 en des défis et en idées directrices de notre contribution.

4.2.1 \mathcal{O}_1 : Capturer des exigences de justifications

Les **DJs** étant seulement une notation informelle, il va s'agir sur cet axe de formaliser les différents concepts et leurs mécaniques autour des **DJs**.

C_{1.1} : Préserver la sémantique et les représentations connues adaptées à la justification

La formalisation que nous allons proposer doit reposer sur les notations graphiques existantes. En effet, les **DJs**, mais aussi **GSN**, **CAE** ont prouvé leur intérêt dans leur utilisation actuelle (\mathcal{P}_2). Il est donc préférable pour l'adoption et l'utilisabilité de notre approche de reposer sur ses notations existantes. Cet aspect permet également de rester sur une sémantique dédiée aux justifications (\mathcal{P}_1). Bien que se basant sur les **DJs** nous montrons comment transcrire des justifications de notre formalisme vers **SACM** et donc in fine vers **GSN** et **CAE** (\mathcal{P}_3).

C_{1.2} : Structurer des normes et pratiques internes

Les exigences de justifications sont les éléments concrets pour être conforme avec une réglementation. L'obtention de ces exigences repose sur la revendication de normes et de processus mis en place pour être en conformité avec la législation. Cette réponse argumentée peut donc se structurer par la modélisation de justifications (\mathcal{P}_3). Nous proposons ici de montrer comment utiliser les **DJs** pour capturer des normes et pratiques internes afin de structurer la réponse à une réglementation. Nous montrons notamment comment cette capture explicite mieux ces pratiques (\mathcal{P}_5).

C_{1.3} : Supporter le raffinement des exigences de justifications

L'évolution au cours du temps des exigences de justifications nous amène à devoir prendre en considération le raffinement de ces justifications (\mathcal{P}_4). En effet, établir des exigences de justifications nécessite de les éliciter en les raffinant tout comme nous pouvons faire avec des objectifs en ingénierie des exigences (\mathcal{P}_6). Ce processus permet d'être de plus en plus précis, mais nécessite d'être supporté pour gagner en confiance durant cette éléction. Par ailleurs, une justification peut-être vue, utilisée différemment suivant le besoin (e.g. un document d'identification des risques pour ISO 14971 et IEC 62304). Il s'agit pour nous de pouvoir répondre à ce besoin de raffinement à travers notre formalisme.

C_{1.4} : Supporter l'évolution des exigences de justifications

Au cours d'un projet, les changements sont multiples : (i) des normes peuvent évoluer, (ii) de nouvelles normes sont revendiquées ou (iii) le processus de développement évolue et mène à des changements dans les pratiques internes (\mathcal{P}_{10}). Toute cette évolutivité a un impact sur les exigences de justifications qui doivent évoluer. Ce processus d'évolution doit être pris en compte pour en capturer les fondements et proposer une meilleure maîtrise de ceux-ci dans le but de gagner en confiance. Nous montrons comment notre formalisme permet de supporter cette évolutivité.

4.2.2 O₂ : Guider la production des justifications

Pour être capable de produire des justifications, cela suppose d'établir formellement un lien entre les exigences de justification et leur production. Cette construction doit se positionner dans un écosystème d'outils où les justifications sont éparses et hétérogènes. Il faut donc également être capable d'interagir avec ces outils pour aider dans ce sens.

C_{2.1} : Supporter le passage à l'échelle dans la production des justifications

Une fois qu'un **DJ** définit les exigences auxquelles il doit répondre, il s'agit de produire ces justifications. Cette production est complexe et coûteuse pour une entreprise et nécessite donc d'être efficiente. Elle repose sur la combinaison d'expertise humaine et d'outils pour produire les justifications. Nous identifions et mettons en œuvre des mécanismes pour aider au passage à l'échelle notamment par l'automatisation (\mathcal{P}_8). Pour ce faire nous nous intéressons à différencier ce qui requiert une expertise humaine de ce qui peut être computationnel et donc automatisable.

Nous montrons notamment comment un méta-modèle dérivé de notre formalisme y contribue.

C_{2.2} : Proposer un environnement intégré pour l'utilisation des justifications

Une fois l'ensemble des justifications produites, il reste à récupérer ces justifications pour les valider, stocker et défendre en audit. Historiquement, cette récupération consistait à imprimer ses justifications et les stocker dans des classeurs. Lors d'un audit, les classeurs étaient sortis et parcourus. Les outils informatiques ont petit à petit permis de délaisser cet archivage papier au profit de systèmes de management de la qualité (SMQ) informatique. Cette modernisation demande une plus forte aide dans la recherche et récupération de documents de justifications dans des systèmes informatiques souvent hétérogènes et répartis (\mathcal{P}_7). Nous montrons ici que notre formalisme supporte des mécanismes d'extraction et de validation permettant de pallier à cette problématique.

4.2.3 O₃ : Vérifier et analyser la conformité entre exigences et production des justifications

Durant et surtout après avoir produit l'ensemble des justifications répondant à une exigence de justification, il faut pouvoir vérifier et valider ces justifications au regard de ce qui était attendu pour mesurer la conformité. On distingue alors la confiance que l'on veut obtenir durant la production des justifications et celle que l'on veut avoir lors de la validation finale.

C_{3.1} : Supporter une production des justifications en continu

La production de justifications n'est pas l'activité finale d'un processus de développement. C'est un processus parallèle au développement qui nécessite la production en continu de ces justifications (\mathcal{P}_8). En même temps, nous voulons pouvoir avoir confiance au fil de l'eau et être capable de jaloner le processus de justifications pour vérifier, mesurer le niveau de conformité à chaque instant (\mathcal{P}_9). Nous nous intéressons à identifier les mécanismes en jeu pour supporter la construction et validation de justification en continu. Nous montrons comment notre formalisme supporte ces mécanismes.

C_{3.2} : Faciliter la validation de justifications

Une fois la production des justifications et la récupération de celles-ci au sein d'un DJ, il se pose la question de valider ces justifications (\mathcal{P}_9). Cette validation passe par un ensemble de questionnement : Est-ce que les justifications produites répondent bien aux exigences initiales ? Y avait-il un moyen de l'obtenir plus facilement avec le même degré de confiance ? Est-ce que toutes les justifications nécessaires ont été produites ? Est-ce que les versions des justifications sont les bonnes ? Toutes ces questions se posent lors de revues qualité régulières et les réponses sont difficiles à obtenir tant la formalisation entre les exigences de justifications et les justifications produites est peu maîtrisée. Nous nous intéressons à formaliser des propriétés pour la confiance qui font le pont entre les exigences de justifications et les justifications

produites.

Pour répondre à \mathcal{O}_1 , \mathcal{O}_2 et \mathcal{O}_3 , il s'agit donc de définir un cadre formel pour les [DJs](#) qui capture la définition initiale donnée par Polascek. Il est également question de l'enrichir pour distinguer les exigences de justifications des justifications concrètes à travers la formalisation de ses deux niveaux dans les [DJs](#) (c.f Chapitre 5). Il s'agira ensuite de préciser comment à partir de ce formalisme nous supportons les propriétés d'évolutivité des exigences de justifications (c.f. Chapitre 6). Ensuite, il s'agira de rendre opérationnel notre formalisme pour notamment atteindre l'automatisation de la production des justifications. Nous proposons ainsi un méta-modèle et une architecture l'intégrant (c.f Chapitre 7) Nous finirons par montrer comment ce méta-modèle permet d'interagir avec d'autres formalismes, ici *SACM* et *i** (c.f Chapitre 7.4).

Deuxième partie

Contribution

Chapitre 5

Formalisation des Diagrammes de Justification (DJs)

« *Ce qui est simple est toujours faux. Ce qui ne l'est pas est inutilisable.* »

Paul Valéry

Sommaire

5.1	Une sémantique formelle pour les Diagrammes de Justification (DJs)	38
5.1.1	Concepts de base	38
5.1.2	Patron de Justification	40
5.1.3	Diagramme de Justification (DJ)	42
5.2	La sémantique en actions	42
5.2.1	Principe	42
5.2.2	Modélisation de la sémantique	43
5.2.3	La relation \mathcal{R} en pratique	43
5.2.4	Raisonnement avec les Diagrammes de Justification (DJs)	43
5.3	Discussion	45

Dans ce chapitre, nous adressons les défis **C_{1.1}** (Préserver la sémantique et les représentations connues adaptées à la justification), **C_{1.2}** (Structurer des normes et pratiques internes) et **C_{1.3}** (Supporter le raffinement des exigences de justifications). À travers la proposition d'une formalisation pour les **DJs**, nous montrons comment ils peuvent être utilisés pour structurer les justifications tout en acceptant un niveau de granularité variable des exigences de justifications jusqu'aux justifications concrètes.

5.1 Une sémantique formelle pour les **Diagrammes de Justification (DJs)**

Dans cette section, nous présentons une sémantique formelle pour les **DJs**. Nous nous concentrons sur les éléments clés de la notation graphique existante définie par Polacsek et nous ne détaillons pas ici la *Limitation*, le *Domaine d'usage* et le *Fondement* (voir Section 2.1.2). Concernant le *Domaine d'usage* et le *Fondement*, ils peuvent être vus comme un ajout à la *Stratégie* et donc être contenus, embarqués, dans la représentation que nous allons en donner. Concernant la *Limitation*, il est aussi possible de la voir comme une commodité d'écriture, du sucre syntaxique, et pas comme un élément constitutif du langage. Nous discuterons plus en détail de ce choix en Section 5.3.

5.1.1 Concepts de base

Dans le chapitre 2, nous avons distingué *support* de *conclusion*. Cependant, les **DJs** manipulent des assertions, des allégations, du type “*le système résiste à une panne*” ou “*les tests sont tous positifs*”. Ces assertions correspondent aussi bien à une conclusion, une sous-conclusion, qu'à une évidence. Elles peuvent être exprimées avec une simple phrase en langage naturel ou être exprimées en langage plus formel, comme la logique. Nous noterons l'ensemble des assertions \mathcal{A} .

Afin de pouvoir comparer certaines assertions entre elles, nous définissons une relation, notée \mathcal{R} , que nous qualifierons de relation de conformité. L'idée sous-jacente à cette notion de conformité est celle d'un “*raffinement*”. Ainsi, si deux assertions sont des formules logiques, \mathcal{R} pourra être vue comme “*est un modèle de*”, si ce sont des classes comme “*est une spécialisation de*” et si nous sommes dans l'utilisation du langage naturel, alors la signification de la relation de conformité \mathcal{R} devra être donnée.

Exemple fil rouge 5.1.1 : Illustration du formalisme sur l'exemple fil rouge

Pour illustrer notre formalisme, nous utilisons l'exemple donné en Figure 5.1. Ce cas d'étude présente une justification pour la validation de l'expertise d'un vacataire. La partie *Patron d'expertise candidature* donne le canevas pour justifier ceci, basé sur la stratégie *valider expertise en comité d'experts* supporté par *Avis des experts* et *Section CNU du poste*. Nous raffinons successivement les concepts de justification autour de ce canevas pour les amener dans la réalité du terrain à l'IUT. Par exemple, dans *Pas de justification pour l'IUT pour vacataire en informatique*, cela consiste pour *Avis des experts* à enregistrer cet avis dans un rapport dont le format a été défini par l'IUT, *Rapport issu du template IUT*. Pour *Section CNU du poste*, cela consiste à faire la parallèle entre la section CNU pour l'informatique (27) et un diplôme qu'un vacataire est susceptible d'avoir, *Diplôme Master en informatique*. Dans ce pas de justification concret, les conclusions et supports peuvent être reliés à de réels documents, page web, etc.

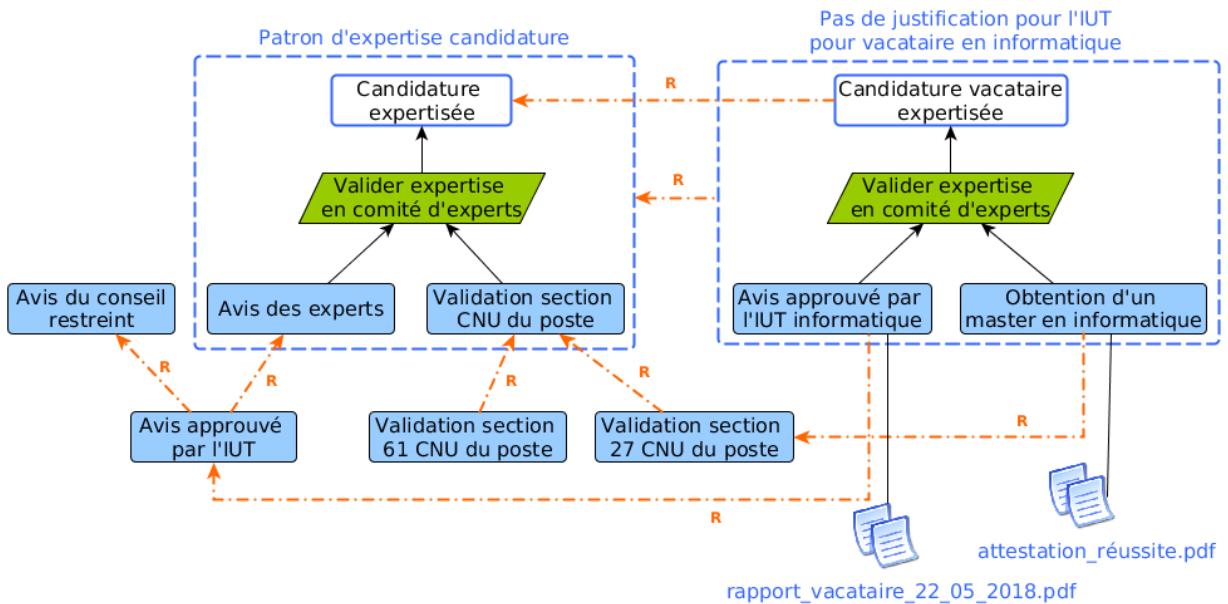


FIGURE 5.1 – Exemple de relation \mathcal{R} , affichée comme une extension de la notation des DJ (e.g., un *Template de réponse IUT* est conforme à un *Avis des experts*)

Définition 5.1.1 : Relation de conformité

Soit \mathcal{A} l'ensemble des assertions, \mathcal{R} est une relation de conformité ssi $\forall a_1, a_2, a_3 \in \mathcal{A}$:

- $a_1 \mathcal{R} a_1$
- if $a_1 \mathcal{R} a_2$ and $a_2 \mathcal{R} a_1$ then $a_1 = a_2$
- if $a_1 \mathcal{R} a_2$ and $a_2 \mathcal{R} a_3$ then $a_1 \mathcal{R} a_3$

$a_1 \mathcal{R} a_2$ se lira a_1 est conforme à a_2 .

Notons qu'une relation de conformité définit un ordre partiel sur l'ensemble des assertions, elle est réflexive, transitive et antisymétrique. D'après cette définition, plusieurs assertions peuvent être conformes à une même assertion. De façon dual, une assertion peut aussi être conforme à plusieurs assertions. Par ailleurs, la relation de conformité n'est pas limitée à 2 niveaux, ce qui est caractérisé par la transitivité. Cela permet de supporter la capture pas-à-pas des normes, pratiques internes et moyen de production des justifications par ce mécanisme de conformité.

Exemple fil rouge 5.1.2 : La relation de conformité sur l'exemple Datadock

La Figure 5.1 illustre de telles conformités hiérarchisées sur un exemple concret.

Validation section 61 CNU du poste et *Validation section 27 CNU du poste* sont conformes à *Validation section CNU du poste*

Avis approuvé par l'IUT est conforme à *Avis du conseil restreint* et également à *Avis des experts*.

Avis approuvé par l'IUT informatique est conforme à *Avis approuvé par l'IUT* et par transitivité à *Avis des experts*, mais aussi à *Avis du conseil restreint*.

Nous définissons un type particulier d'assertions, les assertions terminales. Nous dirons qu'une assertion est un terminal si aucune assertion n'est conforme avec elle. Géné-

ralement, un terminal est un fait concret comme un résultat bibliographique, une bonne pratique établie, un point de la spécification ou le résultat de test.

Définition 5.1.2 : Terminal

$a \in \mathcal{A}$ est un terminal ssi $\forall a' \in \mathcal{A}, a' \mathcal{R} a \Rightarrow (a' = a)$.

Exemple fil rouge 5.1.3 : Les terminaux dans Datadock

Ainsi, dans la Figure 5.1, *Avis approuvé par l'IUT informatique* est une assertion terminale. Cette assertion fait référence à un document unique auquel aucune autre assertion ne se conforme dans le contexte étudié.

Comme chez Toulmin, la stratégie est la pierre angulaire des diagrammes de justification. C'est elle qui explicite, comment, à partir d'éléments de preuve, il est possible de déduire une conclusion. Cette déduction n'est pas de l'ordre d'une déduction formelle, sinon il serait inutile d'utiliser les diagrammes de justification. Dès lors, nous avons choisi d'encapsuler tous les éléments de cette déduction dans un concept *Stratégie*. Nous notons l'ensemble des stratégies \mathcal{W} .

Pour finir, nous allons définir un pas de justification, en d'autres termes donner une sémantique à la notation graphique présentée Figure 2.2. Nous noterons l'ensemble des pas de justification : \mathcal{S} .

Définition 5.1.3 : Pas de justification

Un pas de justification (pas de justification) p est un tuple $\langle supports, strategie, conclusion \rangle$ où :

- $supports$ est un ensemble d'assertions $\subset \mathcal{A}$,
- $strategie \in \mathcal{W}$;
- $conclusion \in \mathcal{A}$

Exemple fil rouge 5.1.4 : Pas de justification dans Datadock

Dans l'exemple en Figure 5.1, *Patron d'expertise candidature* et *Pas de justification pour l'IUT pour vacataire en informatique* sont des pas de justifications .

5.1.2 Patron de Justification

Un patron de justification structure des pas de justification attendus; c'est une abstraction de justifications concrètes et donc une remontée au niveau des exigences de justifications. Avant de démarrer un projet, identifier les justifications nécessaires est crucial. Cependant, caractériser des patrons est une tâche complexe. Il faut avoir une vision globale du processus de développement comme des justifications, une bonne connaissance des normes applicables, connaître l'usage prévu du produit ainsi qu'une connaissance transverse en matière de justification (e.g., projets précédents, pratiques internes standardisées). Par l'utilisation de patrons de justification, les experts raisonnent ainsi sur les types de justifications et d'artefacts (e.g., couverture de tests, besoins) plus que sur de

réelles justifications (e.g., rapport Jacoco des résultats de tests, cahier des charges au format de l'entreprise). Le patron est un canevas plus ou moins strict permettant de donner un fil conducteur au projet en termes de justifications.

À l'aide de la relation de conformité \mathcal{R} , il est possible d'étendre le concept de conformité aux pas de justification. Si toutes les assertions dans un pas de justification sont conformes aux assertions dans un autre pas de justification, alors le premier pas de justification est dit conforme au second. Nous donnons ci-après une définition de la conformité entre pas de justification.

Définition 5.1.4 : Conformité entre pas de justification

Un pas de justification $s = \langle supp, strat, c \rangle$ est dit conforme à un pas de justification $s' = \langle supp', strat', c' \rangle$ ssi :

- $\forall a \in supp, \exists a' \in supp, a \mathcal{R} a'$,
- $\forall a' \in supp', \exists a \in supp, a \mathcal{R} a'$,
- $\forall a, b \in supp, \forall a' \in supp', \text{ si } a \mathcal{R} a' \text{ et } b \mathcal{R} a' \text{ alors } a = b$,
- $strat = strat'$,
- $c \mathcal{R} c'$.

La définition que nous donnons ici priviliege la préservation des concepts encapsulés dans les supports d'un pas de justification plutôt que la cardinalité ferme. En effet, si nous considérons les pas de justification $p_1 = \langle \{s\}, w, c \rangle$ et $p_2 = \langle \{s_1, s_2\}, w, c \rangle$, et $s \mathcal{R} s_1$ et $s \mathcal{R} s_2$ d'après notre définition p_1 est conforme à p_2 . Ceci est conforme à la réalité, où le passage de normes à des pratiques internes amène à éliciter les exigences de justifications et donc à dériver d'un gros grain à un grain plus fin ces exigences en n'en séparant certains aspects en deux justifications disjointes.

Exemple fil rouge 5.1.5 : Conformité des pas de justification dans Datadock

Dans l'exemple présenté en Figure 5.1, *Pas de justification pour l'IUT pour vacataire en informatique* est conforme à *Patron d'expertise candidature*.

Dans un autre exemple, prenons un pas de justification qui permet de justifier d'une simulation numérique à partir des 2 assertions suivantes : “*la température est inférieure à 90 degrés*” et “*la température est plus élevée que 0 degré*”. Prenons maintenant, un pas de justification avec la même stratégie et la même conclusion, mais ici avec pour support “*la température est entre 10 et 50 degrés*”. Si nous considérons que cette assertion est une spécialisation des deux autres, au sens où elles sont liées par \mathcal{R} , alors ce pas de justification est conforme au premier.

Si nous voulons prendre en compte la cardinalité, c'est-à-dire ne considérer comme conformes entre eux que des pas de justification qui ont le même nombre de supports, nous devons considérer que la relation \mathcal{R} est, pour l'ensemble des supports des deux pas de justification, une relation bijective, en d'autres termes, rajouter la condition suivante : $\forall a \in supp, \forall a', b' \in supp', \text{ si } a \mathcal{R} a' \text{ et } a \mathcal{R} b' \text{ alors } a' = b'$.

Notons que la relation de conformité entre les pas de justification, quelle que soit la définition choisie, est, elle aussi, une relation d'ordre partiel. La démonstration est triviale puisque cette nouvelle conformité est simplement basée sur la relation \mathcal{R} entre les assertions. Par commodité nous noterons $s' \mathcal{R} s$, le fait que le pas de justification s' est conforme au pas de justification s .

Propriété 5.1.1 : Propriétés de la relation de conformité

La relation de conformité entre pas de justification est réflexive, transitive et antisymétrique.

Maintenant que nous avons une relation entre pas de justification, nous pouvons introduire formellement ce qu'est un patron de justification.

Définition 5.1.5 : Patron de Justification

$\forall s \in \mathcal{S}$, s est un patron de justification ssi $\exists s' \in \mathcal{S}, s' \mathcal{R} s$ et $s \neq s'$.

De la même façon que les patrons de justification, nous pouvons formellement définir le *raffinement d'un patron* et un *pas de justification terminal*.

Définition 5.1.6 : Raffinement d'un patron de justification et pas de justification terminal

- $\forall s \in \mathcal{S}$, s est un pas de justification terminal ssi $\forall s' \in \mathcal{S}, s' \mathcal{R} s \Rightarrow (s' = s)$.
- Inversement, $\forall s \in \mathcal{S}$, s est un raffinement de s' ssi $\exists s' \in \mathcal{S}, s \mathcal{R} s'$.

Exemple fil rouge 5.1.6 : Patron de justification dans Datadock

Dans l'exemple en Figure 5.1,

- *Patron d'expertise candidature* est un patron
- *Pas de justification pour l'IUT pour vacataire en informatique* est conforme *Patron d'expertise candidature* en cela il raffine le précédent.
- *Pas de justification pour l'IUT pour vacataire en informatique* est un pas de justification terminal

5.1.3 Diagramme de Justification (DJ)

Pour justifier d'une conclusion, il peut être nécessaire de construire plusieurs pas de justification. Nous avons simplement défini un **Diagramme de Justification (DJ)** comme étant un ensemble de pas de justification. Notons qu'un **DJ** peut être composé à la fois de patrons et de pas de justification terminaux.

Définition 5.1.7 : Diagramme de Justification (DJ)

Un **Diagramme de Justification (DJ)** jd est un ensemble de pas de justification \mathcal{S} .

5.2 La sémantique en actions

5.2.1 Principe

La question ici est alors de mettre en œuvre notre sémantique afin de déterminer les relations entre les **DJs**, les absences de justifications, etc. Notre but est également de faciliter la construction des **DJs**. Cette construction peut être manuelle et/ou programmatrice

en fonction du domaine ciblé, de l'environnement de développement et de la justification elle-même. Ainsi lorsque les justifications portent sur des tâches qui échappent au contrôle du système informatique, il appartient aux experts de construire les pas de justification. Inversement, dans un contexte de justification d'un processus pris en charge par le système informatique, il doit être possible de construire les pas de justification de manière automatique. En utilisant les pas de justification comme un guide, les outils doivent alors faciliter la construction incrémentale des pas de justification, permettre l'identification des assertions ou pas de justification manquants, vérifier des conformités.

5.2.2 Modélisation de la sémantique

Pour être capable de supporter une telle approche, il est nécessaire d'avoir des outils. En effet, reposer sur des outils est ici essentiel pour assurer la conformité entre les pas de justifications et donc de renforcer la confiance que l'on peut avoir dans les justifications sur un produit.

Nous proposons ici de capturer notre formalisme à travers un méta-modèle, noyau de nos outils. Ce méta-modèle est présenté en Figure 5.2. Nous y trouvons tous les concepts de la notation des *DJs* qui ont été capturés par notre sémantique. En pratique, un *Repository* contient les *JustificationDiagram*, les *Assertion*, les *Strategy*, et les *Artifact*. Les *Artifact* sont un concept rajouté ici car étant dans un contexte outils, il est important de référencer le ou les documents reliés à une assertion. Ces *Artifact* peuvent être un rapport au format PDF, une page web ou même un résultat d'expérimentation dans un format dédié. Seules les justifications terminales sont concernées par cet aspect puisqu'elles sont les seules à être la concrétisation des justifications et sont donc matérialisées par une trace documentaire.

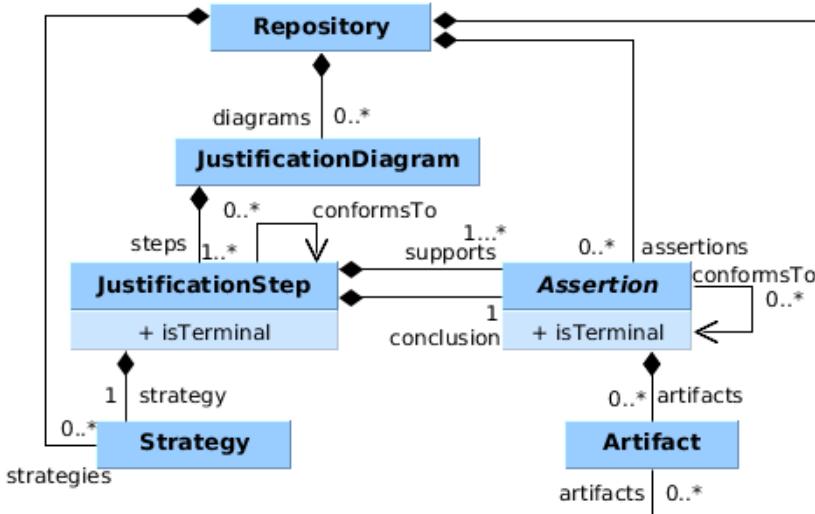
L'association *conformsTo* capture les relations de conformité établies entre les *Assertion* et les *JustificationStep*. Les concepts d'*assertion terminale*, *pas de justification terminal* ont été implémenté avec l'attribut *isTerminal*. En effet, nous travaillons ici dans un espace non clos, de nouvelles justifications peuvent apparaître, nous avons donc dû faire ce choix de rendre cet aspect dynamique.

5.2.3 La relation \mathcal{R} en pratique

La relation de conformité \mathcal{R} est un élément clé de notre sémantique. C'est pour pouvoir rester générique et accepter tout type d'assertion, que nous avons choisi de ne pas caractériser son comportement. Cependant, dans le cadre d'un logiciel, il nous faut faire des choix d'implémentation de \mathcal{R} . Comme nous l'avons précédemment évoqué, la relation \mathcal{R} peut être vue comme un raffinement, une spécialisation, etc. Nous avons choisi de représenter ce lien au travers d'une association *conformsTo*. Une *Assertion* ou un *JustificationStep* peuvent se conformer à 0 ou n éléments. Cette sémantique est forcément dépendante du format choisi pour représenter les assertions. Ainsi, si les assertions sont représentées en logique, il est possible de considérer la conformité comme : "est conséquence logique", si les assertions sont dans un langage contrôlé (boilerplate), la conformité peut être une relation d'instanciation et dans le cadre d'assertion en langue naturelle, la conformité sera un algorithme réalisant du traitement automatique de la langue. La conformité entre les concepts du méta-modèle est seulement l'implémentation des définitions respectives 5.1.1 et 5.1.4.

5.2.4 Raisonnement avec les *Diagrammes de Justification (DJs)*

Basé sur notre sémantique, il s'agit ensuite de proposer des fonctionnalités qui aident dans la gestion des *DJ*. Nous présentons ici un ensemble de fonctionnalités


 FIGURE 5.2 – Noyau du métamodèle de *DJ*

Fonctionnalité \mathcal{F}_1 : Comment justifier une conclusion ?

Considérant une conclusion, cette fonctionnalité permet d'obtenir tous les pas de justifications qui permettent de justifier celle-ci. D'un point de vue de la sémantique cela revient, pour une assertion *goal*, à retourner tous les $s \in \mathcal{S}$ avec $s = \langle S, w, c \rangle$ tel que : $goal \mathcal{R} c$.

Exemple fil rouge 5.2.1 : Usage de \mathcal{F}_1

Si nous reprenons l'exemple de la Figure 2.3, pour aider à la justification de la conclusion *Candidature validée*, la \mathcal{F}_1 donnerait le pas de justification contenant la stratégie *Valider la candidature par le service RH de l'université*

Fonctionnalité \mathcal{F}_2 : Que peut-on justifier ?

À partir d'un ensemble d'assertions, il s'agit de déterminer tous les pas de justification qui peuvent être appliqués. En d'autres termes, pour un ensemble d'assertions *A*, il faut trouver tous les $s \in \mathcal{S}$ avec $s = \langle S, w, c \rangle$ tel que : $\forall s' \in S, \exists a \in A, a \mathcal{R} s'$.

Exemple fil rouge 5.2.2 : Usage de \mathcal{F}_2

Si nous reprenons l'exemple de la Figure 2.3, si nous demandons par \mathcal{F}_2 que ce que l'on peut justifier à partir des assertions *CV* et *Fiche de service*, il sera fourni non seulement le pas de justification contenant la stratégie *Valider fiche de service par le chef du département*, mais également *Valider CV par la CV-thèque*

Notre but étant de faciliter la construction des *DJs* autant manuellement qu'automatiquement, \mathcal{F}_1 et \mathcal{F}_2 sont utilisées ainsi pour supporter à une telle automatisation ou aider les experts.

Fonctionnalité \mathcal{F}_3 : Est-ce qu'une assertion supporte un pas de justification ?

Nous considérons qu'une assertion *supporte* un pas de justification si elle est conforme à un de ces supports ou si elle aide à justifier un de ses supports. Du point de vue de la sémantique, pour une assertion donnée a et un pas de justification donné $js = \langle supp, w, c \rangle$,

a supports js ssi $\exists s \in supp$ tel que $a \mathcal{R} s$ ou $\exists js' \in \mathcal{S}, js' = \langle supp', w', c' \rangle, c' \mathcal{R} s$ et a supports js' .

Exemple fil rouge 5.2.3 : Usage de \mathcal{F}_3

Si nous reprenons l'exemple de la Figure 2.3, \mathcal{F}_3 permet de déterminer que *Attestion employeur principal* supporte le pas de justification contenant la stratégie *Valider le dossier financier par le service RH du département*, mais également par transitivité les pas contenant les stratégies *Valider la candidature par le service RH de l'université* et *Valider qualification professionnelle par le conseil du département*.

\mathcal{F}_3 permet de déterminer quels pas sont basés sur l'existence d'une assertion particulière, mais aussi de déterminer les assertions qui sont inutiles, au sens qu'elle ne supporte aucun pas de justification.

5.3 Discussion

Dans ce chapitre, nous avons défini une sémantique pour les **DJs**. En gardant les mêmes concepts clés et en formalisant leur relation, cela nous a permis d'atteindre **C_{1.1}** (Préserver la sémantique et les représentations connues adaptées à la justification). À travers l'exemple fil rouge, nous avons montré comment ce formalisme peut être utilisé pour structurer des normes et des pratiques internes (**C_{1.2}** - Structurer des normes et pratiques internes). La définition de la conformité par une relation dédiée \mathcal{R} nous permet de supporter le raffinement successif des pas de justification (**C_{1.3}** - Supporter le raffinement des exigences de justifications).

Dans ces travaux, il reste néanmoins certaines limites. En effet, les concepts de *Domaine d'usage* et de *Fondement* ne sont pas définis par notre formalisme, ils sont contenus par la stratégie. Par conséquent la modélisation permet de raisonner sur les assertions, mais difficilement sur le contenu d'une stratégie. Notre vision pragmatique de ce formalisme nous a amenés à faire ce choix, car nous avons voulu proposer un formalisme suffisant pour construire des **DJs** et les analyser. Il a donc été réduit au strict nécessaire pour nos études de cas et ainsi gagné en utilisabilité. Durant nos différentes utilisations du formalisme, nous n'avons pas eu besoin d'exprimer des propriétés liées au contenu d'une stratégie et par conséquent avons choisi sciemment de ne pas le formaliser. Il pourrait être intéressant de formaliser cet aspect pour permettre d'explorer un raisonnement plus fin sur le contenu des stratégies, en monitorer les usages et le rajouter à ce formalisme si son usage se révèle pertinent.

D'autre part, la capture de notre sémantique dans le méta-modèle que nous avons proposé permet de tendre vers une approche outillée. Néanmoins, le modèle proposé capture les éléments clés de la sémantique et leurs relations mais aurait pu être enrichi pour capturer toute notre sémantique. Par exemple, définir le mécanisme de transitivité de *isTerminal* entre un *JustificationStep* et ses *Assertion*. Pour ce faire, l'utilisation d'un langage de contraintes sur modèle comme *Object Constraint Language (OCL)* **WAR**-

MER et KLEPPE [1998] parait être une idée prometteuse permettant de définir un modèle au plus proche de notre sémantique.

La relation \mathcal{R} apporte une grande flexibilité de modélisation de la conformité, bien au-delà d'une relation *type - donnée*. Cependant son usage intensif couplé à cette flexibilité peut engendrer de la *complexité* dans la compréhension des diagrammes. Nous abordons ce point au chapitre suivant en proposant un contrôle du cycle de vies des *DJs*.

Chapitre 6

Cycles de vie des Diagrammes de Justification (DJs)

« *They always say time changes things, but you actually have to change them yourself.* »

Andy Warhol

Sommaire

6.1 Diagrammes-Patron de Justification (DPJs)	48
6.1.1 Concept et sémantique	48
6.1.2 Guider l'évolution	50
6.2 Opérations entre Diagramme-Patron de Justification (DPJ) et Diagramme de Justification (DJ)	53
6.2.1 Formalisation des opérations	53
6.2.2 Modélisation des opérations	56
6.3 Cycles de vies des opérations	57
6.3.1 Cycle de vie itératif pour structurer des exigences de justification et produire les justifications	57
6.3.2 Cycle de vie pour la production et révision des justifications sur la base d'exigences de justification structurées	58
6.4 Discussion	59

Dans ce chapitre, nous adressons les aspects $C_{1.4}$ (Supporter l'évolution des exigences de justifications) et $C_{3.2}$ (Faciliter la validation de justifications). Pour ce faire, nous définissons les opérations qui permettent de contrôler l'évolution des DJ ainsi que catégoriser les cycles de vies qui régissent leur construction. Ceci permet de définir un environnement contrôlé permettant notamment de formaliser les réponses aux questions : comment construire ces diagrammes ? Quand une évolution peut-elle être prise en compte ? Comment analyser et propager l'impact des modifications engendrées ? et ainsi adresser les défis de ce chapitre.

6.1 Diagrammes-Patron de Justification (DPJs)

6.1.1 Concept et sémantique

Chaque pas de justification se construit individuellement, mais c'est l'enchaînement des pas de justification qui construit la justification dans son ensemble. C'est sur cet enchaînement qu'est basée la confiance. En effet, les exigences de justifications décrivent comment atteindre une assertion de haut-niveau à l'aide d'autres assertions. Ainsi l'enchaînement de pas de justifications est le reflet de ces liens, parfois implicites, au sein de ses exigences. L'enchaînement que l'on fait des pas de justifications doit donc être contrôlé pour s'assurer de la capture de ces liens. Au cours d'un projet, les justifications sont amenées à évoluer pour répondre à de nouvelles exigences de justifications ou faire évoluer les moyens de les produire. Nous constatons donc que l'évolution porte 1) sur le pas de justification lui-même (e.g., modification d'un pas qui préserve la conformité) ou 2) sur l'enchaînement (e.g., ajout d'un pas qui casse la relation de conformité). Il s'agit donc d'être capable de qualifier les relations entre DJs pour identifier les relations qui préservent une conformité attendue; nous parlerons de raffinement; des relations qui exigent une nouvelle analyse des justifications.

Définition 6.1.1 : Conformité entre Diagramme de Justification (DJ)

Un DJ $jd = \{s_1..s_n\}$ est conforme à un DJ $jd' = \{s'_1..s'_p\}$ ssi :

- $\forall s \in jd, \exists s' \in jd', s \mathcal{R} s'$,
- $\forall s' \in jd', \exists s \in jd, s \mathcal{R} s'$,
- $\forall s_1, s_2 \in jd, \forall s' \in jd', \text{ si } s_1 \mathcal{R} s' \text{ et } s_2 \mathcal{R} s' \text{ donc } s_1 = s_2$,

Notons que nous avons fait ici à nouveau un choix de cardinalité éventuellement différente. En effet, dans certains cas, on veut pouvoir avoir l'enchaînement de deux pas de justifications conformes à un pas de justification (e.g., une hypothèse de travail utilisé comme évidence à un pas qui n'est plus valide ou à démontrer et qui amène à utiliser un pas de justification pour la justifier). Ce type de conformité permet de *raffiner* des DJs en étant de plus en plus précis, précision qui se traduit par un diagramme dont la taille augmente.

Définition 6.1.2 : Raffinement et Diagramme de Justification (DJ) terminal

Soit \mathcal{D} un ensemble de DJ.

- $\forall jd \in \mathcal{D}, jd$ est une DJ terminal ssi $\forall jd' \in \mathcal{D}, jd' \mathcal{R} jd \Rightarrow (jd' = jd)$.
- $\forall jd, jd' \in \mathcal{D}, jd'$ est un raffinement de jd ssi $jd' \mathcal{R} jd$ et $jd' \neq jd$.

Basées sur ces définitions, nous introduisons un nouveau type de diagramme : **Diagramme-Patron de Justification (DPJ)** pour permettre une meilleure distinction au niveau sémantique entre les exigences de justification (*e.g.*, normes, guides) et les justifications elles-mêmes (*e.g.*, mise en œuvre). En effet, notre sémantique jusqu'ici par la relation \mathcal{R} permet bien de capturer ses deux aspects, mais dans la réalité, il est important d'avoir un vocabulaire les distinguant même si ces concepts au niveau de notre sémantique restent les mêmes. Nous distinguons ainsi les **DJs** : le diagramme qui raisonne sur les exigences de justifications **DPJs** et le diagramme qui raisonne sur les justifications terminales **DJs**¹. Là où un **DJ** est résultat d'un processus de justification au jour le jour, un **DPJ** est un canevas plus ou moins précis qui donne un guide à suivre durant un projet en termes de justifications. Un **DPJ** est ainsi construit à l'aide de patrons de justification, mais également de pas de justification en fonction de l'abstraction qu'il propose (*e.g.*, seulement des exigences de justifications, un mélange entre moyens concrets de répondre à certaines exigences de justifications et le raffinement d'autres exigences). Un patron de justification étant lié à l'existence d'un pas qui le raffine, un **DPJ** est lui aussi contraint à l'existence d'un **DJ** qui le raffine. Ainsi un **DPJ** est relatif à un ensemble de **DJ**. De plus, un **DPJ** ne peut pas être terminal contrairement à un **DJ**.

Définition 6.1.3 : **DPJ** par rapport à un ensemble de **DJ**

Soit \mathcal{D} un ensemble de **DJ** et jd , un **DJ**. jd est un **Diagramme-Patron de Justification (DPJ)** ssi $\exists jd' \in \mathcal{D}, jd'$ raffine jd .

Notons ici que cette définition est un confort de sémantique, nous n'avons rien ajouté au formalisme. Par définition, le premier **DJ** dans l'ensemble \mathcal{D} ne peut pas être un **DPJ** même s'il a vocation à définir des exigences de justifications, il faut attendre la définition d'un autre **DJ** qui le raffine. Notons également que par cette définition d'un **DPJ** nous conservons une propriété souhaitée qui est qu'un **DPJ** peut être composé de patrons de justification et de pas de justification terminaux. Ceci permet d'avoir des **DPJ** plus ou moins précis permettant ainsi de les raffiner successivement pour aboutir à un **DJ** terminal. Ainsi nous supportons, pas à pas, le raffinement d'exigences de justifications jusqu'aux justifications elles-mêmes.

Exemple fil rouge 6.1.1 : **DPJs** sur Datadock

Un exemple de **DPJ** et raffinement de **DPJ** est donné en Figure 6.1. Sur la gauche de la Figure, le **DPJ** donne les exigences de justification liées à *Qualification professionnelle validée* dans le contexte du référentiel Datadock relativement aux pratiques de l'IUT. Le **DPJ** sur la droite de la Figure, est conforme à ce **DPJ**. Tous les éléments attendus par le **DPJ** sont raffinés en des justifications dans ce **DPJ** dans le cadre *Qualification professionnelle vacataire validée*. En effet, les assertions comme *Avis des experts*, *Section CNU du poste* ou *CV* sont respectivement concrétisés par *Rapport issu du template IUT*, *Diplome Master en informatique* et *Page LinkedIn*. Notons que comme la relation \mathcal{R} entre pas de justification n'intègre pas forcément une cardinalité égale sur les supports des deux côtés de la relation, nous pourrions très bien avoir des pas de justifications dans le **DPJ** de droite qui agrège ou complète leurs supports. Par exemple, au lieu de se contenter de la *Page LinkedIn*, il pourrait y avoir un complément d'information avec une *Lettre de recommandation* tout en gardant à la conformité entre ce pas de justification et son patron.

1. par abus de langage, l'approche globale se nomme comme le diagramme contenant les justifications terminales

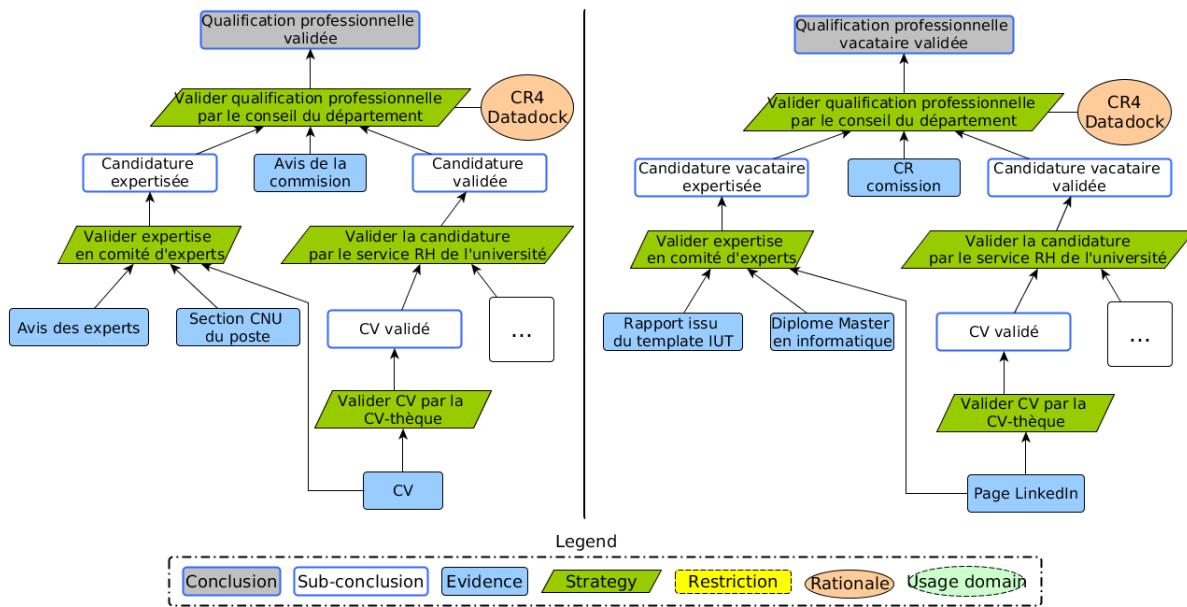


FIGURE 6.1 – Diagramme-Patron de Justification (DPJ) sur la gauche et son Diagramme de Justification (DJ) associé sur la droite sur l'exemple Datadock²

6.1.2 Guider l'évolution

Concevoir un **DPJ** dédié à un domaine permet de structurer les exigences de justifications qui lui sont liées. Ce diagramme est réalisé par des experts qui définissent les patrons de justification en fonction de leurs connaissances, de bonnes pratiques établies, de normes, d'exigences de qualité, etc. Construire un **DPJ** consiste alors, en tenant compte des spécificités du projet cible (*e.g.*, normes spécifiques, usages particuliers, moyens de justification déjà en place dans l'entreprise), à rassembler certains de ces patrons au sein d'un **DPJ**.

Cette tâche est complexe, car elle nécessite d'avoir une vision transverse des besoins de justification du projet et des moyens de réalisation des justifications, ce qui correspond aux stratégies. Il est également souvent difficile de concevoir en amont du projet le **DPJ**. Une façon de construire en toute confiance un diagramme aussi complexe est de partir d'un diagramme proche d'une norme ou guide puis d'itérer avec les parties prenantes (*e.g.*, experts qualité, équipe de développement) pour pas à pas dériver les diagrammes en cherchant la conformité. Cette approche pas à pas facilite la compréhension des équipes et l'interaction entre les parties prenantes. Elle permet aussi de faire évoluer le **DPJ** et ainsi, de rajouter, raffiner ou supprimer des exigences de justifications au cours du projet. Ces évolutions peuvent avoir un impact sur les justifications produites jusque là et donc sur le **DJ**. Il faut donc être capable d'identifier et faire évoluer également les justifications impactées. Par conséquent, à un instant *t*, il peut y avoir une perte de conformité entre les exigences de justification dans le **DPJ** et des justifications dans le **DJ**. En Figure 6.2, nous montrons sur un cas très simple la problématique que cela amène. Comme nous pouvons le voir, l'évolution du **DPJ** par l'ajout d'une évidence *e*₃ sur son seul pas de justification *casse la conformité* avec son prédécesseur et il se pose alors la question de comment transformer le **DJ** associé pour regagner la conformité avec ce nouveau **DPJ**.

Comment avoir alors confiance dans le processus d'évolution des exigences de justifications lorsqu'il mène à une perte de conformité ?

Pour répondre à cette question, nous montrons dans la suite de cette section com-

2. pour des raisons de lisibilité des deux diagrammes en une seule figure nous mettons en “...” tous les éléments de justification qui ne sont pas pertinents pour exemplifier ici

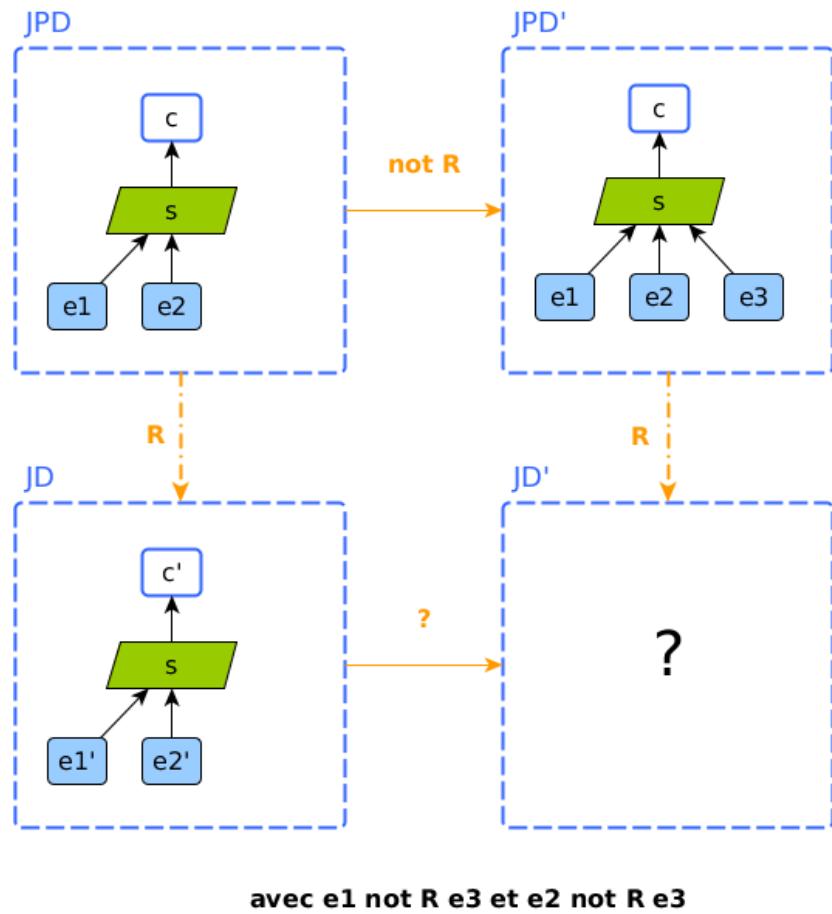


FIGURE 6.2 – Exemple d'évolution d'un DJ et ses impacts

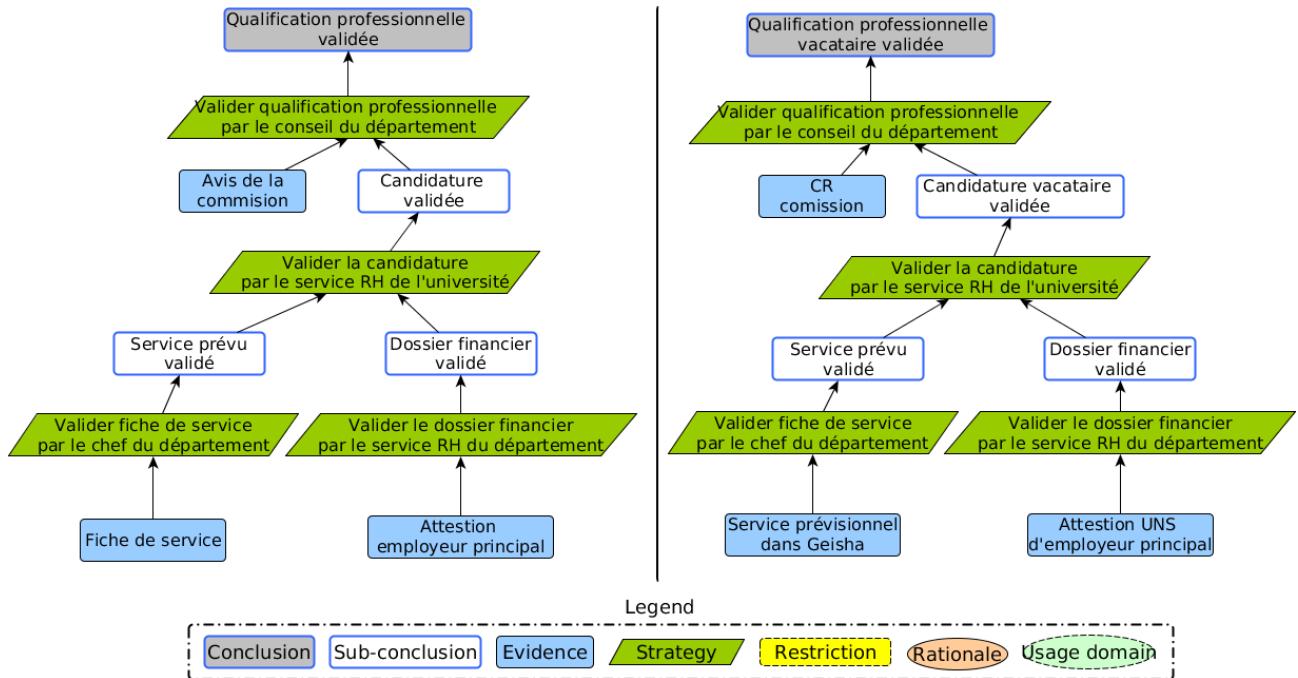


FIGURE 6.3 – Diagramme-Patron de Justification (DPJ) sur la gauche et un Diagramme de Justification (DJ) associé sur la droite pour l'étude de l'existant en vue de préparer Datadock

ment des opérations formellement définies permettent d'encadrer le cycle de vies des diagrammes pour assurer la confiance.

Exemple fil rouge 6.1.2 : Évolutivité avec Datadock

Avant la mise en place du référentiel Datadock au sein de l'IUT, c'est le référentiel "Responsabilité Sociale et Environnementale pour l'Apprenant-Usager" (RE.S.E.A.U.) qui était mis en place. Ce référentiel était supporté par la région PACA. Pour le Critère Réglementaire n°4, il existait un équivalent pour le critère "Gestion des compétences" (Figure 6.4) qui avait pour but de *"s'assurer de l'actualisation et développement des compétences de ses personnels"* ainsi que *"favorise la reconnaissance des compétences et des acquis de ses personnels"*. Ce référentiel est centré sur la formation des personnels déjà recrutés et non sur le recrutement de nouveaux personnels contrairement à Datadock. Passer du référentiel RE.S.E.A.U à Datadock consiste donc à adapter les justifications existantes pour le suivi de formation, mais également à l'introduction de nouvelles exigences de justifications pour le recrutement en se basant sur des pratiques internes existantes. De ce travail découlent les diagrammes en Figure 6.3. À droite, le DJ représente ce qui en pratique était déjà produit dans le cadre du recrutement d'un vacataire par l'université, mais n'était pas revendiqué en tant que justification. À gauche, le DPJ qui représente l'exigence de justification pour Datadock relativement à ce qui peut être produit par l'existant. L'enjeu est alors d'aller pas à pas vers un couple DPJ et DJ concrétisant les exigences de justifications de Datadock comme en Figure 6.1. Nous montrons par les opérations de la section suivante comme y arriver.

2.2 Gestion des compétences

Exigences qualité	Dispositions attendues	Eléments de référence
Le centre de formation assure l'actualisation et le développement des compétences de ses personnels	<p>Le centre de formation :</p> <ul style="list-style-type: none"> définit la structure quantitative et qualitative des emplois et des compétences nécessaire à son activité (et le cas échéant à ses projets de développement). tient à jour l'état des qualifications et compétences détenues par chacun des acteurs (permanents et intervenants réguliers). <p>Le centre de formation calcule le pourcentage de la masse salariale affectée à la formation continue des salariés et veille à sa constante amélioration.</p> <p>Le centre de formation établit et évalue un plan de formation permettant à son personnel d'acquérir les compétences et qualifications requises par son activité et le cas échéant ses projets de développement.</p> <p>Des entretiens d'évaluation individuels sont organisés à échéance régulière et donnent lieu à une discussion sur les compétences acquises et sur les actions à entreprendre pour les actualiser et les développer.</p>	<p>Etat des qualifications et compétences de chaque acteur.</p> <p>Pourcentage de la masse salariale affectée à la formation continue des salariés</p> <p>Plan de formation</p> <p>Attestations obtenues suite aux actions de formation suivies</p> <p>Evaluation du plan de formation</p> <p>Comptes-rendus des entretiens d'évaluation</p>
Le centre de formation favorise la reconnaissance des compétences et des acquis de ses personnels	<p>Le centre de formation met en œuvre les actions d'information et d'appui nécessaires pour permettre à ses personnels d'accroître leurs qualifications et compétences en faisant reconnaître les acquis de leur expérience¹.</p>	Personnels ayant participé à une formation certifiante et attestations de présence

FIGURE 6.4 – Gestion des compétences dans le référentiel RE.S.E.A.U.

6.2 Opérations entre **Diagramme-Patron de Justification (DPJ)** et **Diagramme de Justification (DJ)**

6.2.1 Formalisation des opérations

Durant la production des justifications, remettre en question des patrons pour les adapter à ce qu'il est possible ou plus facile de faire est courant. Ainsi, des mécanismes d'évolution (*e.g.*, améliorer un patron, propager l'impact d'un changement dans un **DJ** pour s'y conformer) doit être méticuleusement pris en compte.

Tout d'abord comme nous l'avons montré dans la section précédente, l'évolution conjointe des exigences de justifications et de la production des justifications doit être précautionneusement étudiée. La traçabilité ascendante entre un **DPJ** et un **DJ**, c'est-à-dire garder une trace de l'origine d'un **DJ**, est un des mécanismes premiers à avoir pour permettre la propagation des impacts lors de l'évolution d'exigences de justifications ou des justifications elles-mêmes. Sans ce lien de traçabilité, la relation \mathcal{R} ne suffit pas. En effet, dès lors qu'une perte partielle de conformité entre **DJ** et **DPJ** est atteinte, il faut être capable d'identifier les impacts que cette évolution peut avoir. Une première opération découle de la relation de conformité entre deux **DJs** et permet à sa création de le *taguer* comme *réaliser à partir de*.

Définition 6.2.1 : Réalisation d'un **DJ** à partir d'un **DPJ**

Soit jpd un **DPJ**, l'opération de *réalisation* consiste à construire un **DJ** terminal jd tel que $jd \mathcal{R} jpd$ et à garder les liens de traçabilité ascendante.

Exemple fil rouge 6.2.1 : Exemple de réalisation

Trivialement, dans la Figure 6.3, le **DJ**, à droite, réalise le **DPJ** à gauche. Chaque pas d'un **DPJ** amène à sa réalisation dans un pas dans le **DJ** où l'on garde la référence du

pas originel.

Cette opération permet de formaliser le lien *ascendant* entre un **DPJ** et un **DJ**. La *réalisation* d'un **DPJ** par un **DJ** permet de rendre concret les justifications en aidant la production à partir du guide que représenter le **DPJ**. Dès lors, construire une justification pour un objectif fixé consiste à construire un **DJ** composé de pas de justification terminaux conformes aux patrons du **DPJ**.

Pour supporter les évolutions qui *cassent* la conformité, il s'agit ensuite de se reposer sur des opérations permettant de comparer et 1) faire remonter des éléments de justifications pertinents du **DJ** vers le **DPJ**, mais aussi 2) faire évoluer le **DPJ** et propager ces changements sur le **DJ**. Dès lors, il faut en premier lieu être capable de comparer les différences entre **DJs**.

Définition 6.2.2 : Comparaison entre DJs

Soit jd_o un **DJ** et jd_t un **DJ**. Soit S_a , l'ensemble des pas de justifications ajoutés, S_r l'ensemble des pas de justifications supprimés, S_m l'ensemble des pas de justifications modifiés.

- si $\exists s_t \in jd_t, \nexists s_o \in jd_o, s_t \not\sim s_o$ alors $s_t \in S_a$
- si $\exists s_o \in jd_o, \nexists s_t \in jd_t, s_t \not\sim s_o$ alors $s_o \in S_r$
- si $\exists s_t \in jd_t, \exists s_o \in jd_o, s_t \not\sim s_o$ et $s_o \neq s_t$ alors $s_t \in S_m$

Ainsi $compare(jd_o, jd_t) = < S_a, S_r, S_m >$.

Procéder à la *comparaison* entre **DJ** consiste alors à établir S_a , S_r et S_m

Exemple fil rouge 6.2.2 : Exemple de comparaison

Si nous considérons le **DPJ** de la Figure 6.1 et celui de la Figure 6.3, la comparaison amènera à déduire que

- S_a est constitué du pas *valider expertise en comité d'experts et valider CV par la CV-thèque*
- S_m est constitué de *valider la candidature par le service RH de l'université et valider qualification professionnelle par le conseil du département*. En effet, ces deux pas ont été complétés par un nouveau support *CV validé* et *Candidature expertisée*, ce que l'on va considérer comme une relation de conformité entre les pas du premier **DPJ** et du second.
- S_r est vide

Nous voyons ainsi que l'ajout d'un pas oblige à raffiner un autre pour permettre de l'insérer dans la justification globale à travers le diagramme. De façon analogue, ce mécanisme régit également les pas supprimés.

Notons que dans le cas particulier où l'on compare deux **DJs** qui se raffinent, la définition 6.2.2 peut être précisée. En effet, le raffinement d'un **DJ** jd ne fait que modifier les pas de l'autre **DJ** jd' . Par conséquent, par définition, $compare(jpd, jd') = < \emptyset, \emptyset, S_m >$ et $m \neq \emptyset$

Cette propriété est utile pour évaluer l'évolution conjointe des **DPJs** et **DJs**, mais ne suffit pas à guider cette évolution. En effet, contraindre un **DJ** à être à tout instant conforme à un **DPJ** est seulement concevable si l'on est absolument certain de la pertinence du **DPJ** au regard de la réalité des pratiques dans l'entreprise. En fonction de la complexité et du

niveau de maturité des systèmes à produire, mais également de l'entreprise, cela peut être utopique (e.g., exigence impossible à remplir, usage du produit qui évolue). Il est donc important de rester souple pour permettre la *dérivation* d'un DJ tout en introduisant un mécanisme d'*alignement* entre un DPJ et un DJ permettant de regagner la conformité. Nous formalisons ces deux nouvelles opérations dans les définitions suivantes.

Définition 6.2.3 : Degré de dérivation d'un DJ

Soit jd_o un DJ et jd_t un DJ. Soit $\text{compare}(jd_o, jd_t) = \langle S_a, S_r, S_m \rangle$. jd_t est dit *dérivé* de degré n de jd_o si $|S_a \cup S_r| = n$

Procéder à la *dérivation* d'un DJ consiste alors obtenir une nouveau DJ en déterminant le nombre n d'ajouts ou suppressions qui ne respectent pas la relation de conformité entre les diagrammes.

Exemple fil rouge 6.2.3 : Exemple de dérivation

Si nous considérons le DPJ jpd_1 de la Figure 6.1 et celui de la Figure 6.3 jpd_2 , jpd_1 est dérivé de degré 3 de jpd_2 . En effet $\text{compare}(jpd_2, jpd_1)$ donne $S_r = \emptyset$, $S_m = \emptyset$ et S_a est réduit à 3 pas de justification : *valider la candidature par le service RH*, *valider la fiche de service par le chef de département* et *valider le dossier financier par le service RH du département*.

Si l'on souhaite atteindre la conformité entre deux DJs non conformes, il convient alors de décider pour chaque pas de justification mis en exergue par l'opération de comparaison ce qu'il faut conserver. Nous parlons alors de l'opération d'*alignement*, celle qui permet de mener à la conformité.

Définition 6.2.4 : Alignement entre DJs

Soit jd, jd', jd^+, jd'^+ des DJs.

$$\text{align}(jd, jd') = (jd^+, jd'^+) \text{ si } jd \text{ not} \mathcal{R} jd' \text{ et } jd^+ \mathcal{R} jd'^+$$

Notons que cette définition ne dirige pas l'alignement sur jd plus que sur jd' . Il est possible d'avoir un alignement menant à des modifications sur les deux. Néanmoins, il est aussi possible de diriger l'alignement sur jd (respectivement jd') alors $\text{align}(jd, jd') = (jd^+, jd')$ (respectivement $\text{align}(jd, jd') = (jd, jd'^+)$).

Exemple fil rouge 6.2.4 : Exemples d'alignement

Considérons dans la Figure 6.5, le DJ jd à droite, le DPJ jpd à gauche. Ici $S_a = \{s_2, s_4\}$, S_r est vide et $S_m = \{s_1, s_1', s_3, s_3'\}$. Étudions deux cas, 1) l'un où tous les ajouts sont acceptés et doivent être propagés dans jd , l'autre où ils sont tous rejetés et l'alignement du jpd sur le jd doit être fait.

Dans le cas n°1, les pas s_1' et s_3' devront ainsi être remplacés dans jd par des pas s_1'' et s_3'' tels que $s_1'' \mathcal{R} s_1$ et $s_3'' \mathcal{R} s_3$. Ceci permet ensuite d'ajouter deux pas s_2'' et s_4'' dans jd tels que $s_2'' \mathcal{R} s_2$ et $s_4'' \mathcal{R} s_4$.

Dans le cas n°2, les pas s_1 et s_3 devront ainsi être remplacés dans jpd par des pas s_1'' et s_3'' tels que $s_1' \mathcal{R} s_1''$ et $s_3' \mathcal{R} s_3''$. Ceci permet de couper les liens vers s_2 et s_4 qui peuvent ainsi être enlevés de jpd

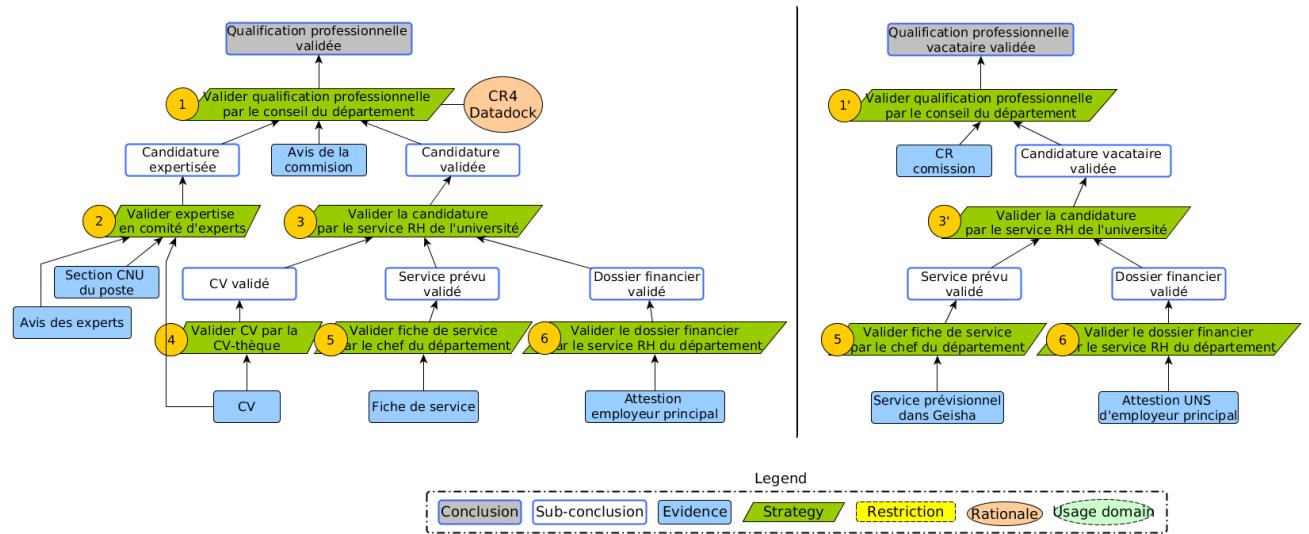


FIGURE 6.5 – Exemple de DPJ et DJ avant alignement

Dans ces deux cas, à la fin de l'alignement est vérifié puisque $jpd\mathcal{R}jd$.

6.2.2 Modélisation des opérations

Afin de nous rapprocher d'une approche outillée, les opérations définies dans la section précédente doivent être introduites dans le méta-modèle des DJs. Il nous faut alors réviser le concept de JustificationDiagram de la Figure 5.2. Cette extension est présentée en Figure 6.6. Notons dans ce méta-modèle, l'absence de distinction au niveau des concepts entre DJ et DPJ. Ceci est directement capturé, comme pour les assertions et les pas de justifications, par *isTerminal* et l'association *conformsTo*. L'opération de *réalisation* consiste finalement à attacher à chaque JustificationStep son origine par l'association *achievedFrom*. L'opération de *dérivation* est matérialisée par l'association *derivedFrom* permettant ainsi d'aborder ce lien de traçabilité. Les opérations de *comparaison*, *alignement* sont elles abordées comme des méthodes d'un JustificationDiagram. Notons ici que S_a , S_r et S_m sont matérialisés par l'énumération ComparisonType de laquelle sont tagués les pas répondant aux critères de ces ensembles dans les méthodes associées.

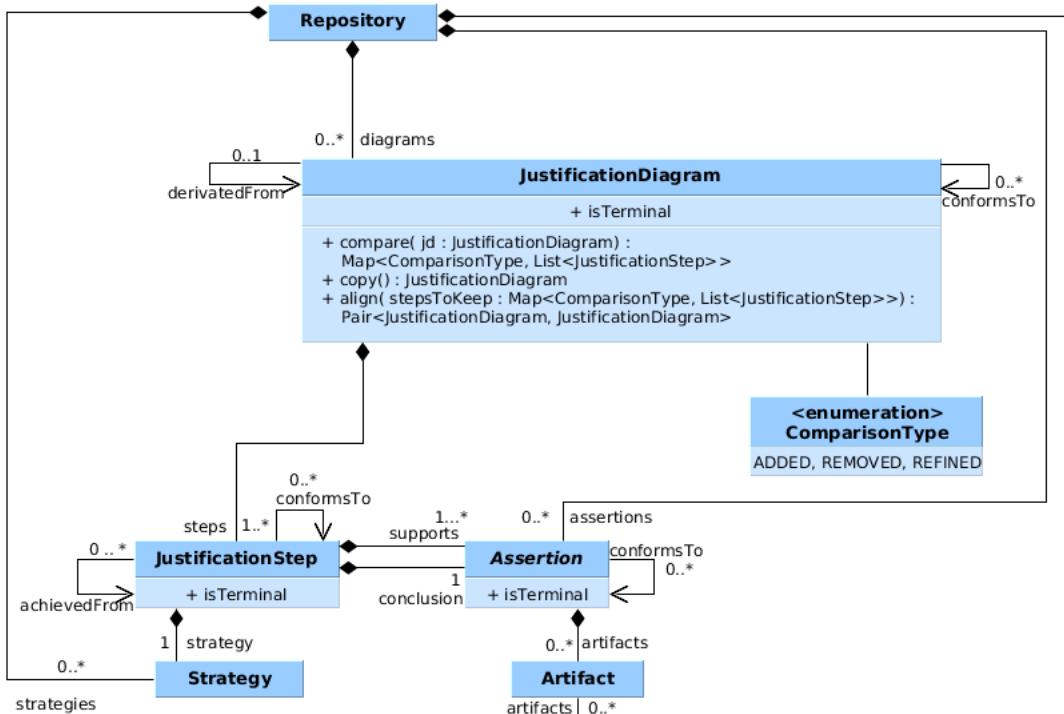


FIGURE 6.6 – Modélisation des opérations

6.3 Cycles de vies des opérations

Dans la réalité, ces opérations sont utilisées de façon très différente en fonction du contexte d'usage. En effet, le niveau de maturité de l'entreprise en termes de justifications (e.g., existence de projets “justifiés”, structuration de pratiques internes), mais également d'organisation autour du projet ciblé (e.g., cycle de développement, réorganisation) ont un impact sur le cycle de vie des DJ. Il existe donc plusieurs cycles de vies pour ces diagrammes. Il ne s'agit pas ici de donner une taxonomie de ces cycles de vies, mais seulement de montrer que nos opérations permettent de répondre à des besoins bien différents que nous avons pu identifier à travers les différents cas d'études que nous avons effectués.

6.3.1 Cycle de vie itératif pour structurer des exigences de justification et produire les justifications

Dans un contexte mouvant entre choix d'organisation, revendication des normes adéquates, expérimentation sur la capacité à produire ces justifications, il est difficile de structurer des justifications. Ceci amène autant à faire évoluer le DPJ que le DJ en fonction de l'évolution des exigences de justifications et des retours du terrain. En Figure 6.7, nous introduisons un flot possible des opérations dans le cas où l'élicitation des justifications est amenée empiriquement pendant le développement. Ce flot montre une façon itérative d'établir un DPJ et un DJ associé pertinent. Ce cycle permet à chaque étape de faire évoluer autant le DPJ que DJ. Itérer sur l'élicitation des exigences de justifications (1. *dérivation*), la pratique sur le terrain pour produire les justifications associées (2. *réalisation*, 3. *dérivation*) et l'alignement entre exigences et retour du terrain (4. *comparaison*, 5. *alignement*) permet d'assurer la convergence vers de telles justifications. En effet, dans ce contexte, il est tout particulièrement important de supporter des *dérivations* sur les exigences de justifications (e.g., norme non identifiée ou mal analysée), mais aussi sur les justifications elles-mêmes (e.g., justification plus facile à obtenir tout en répondant

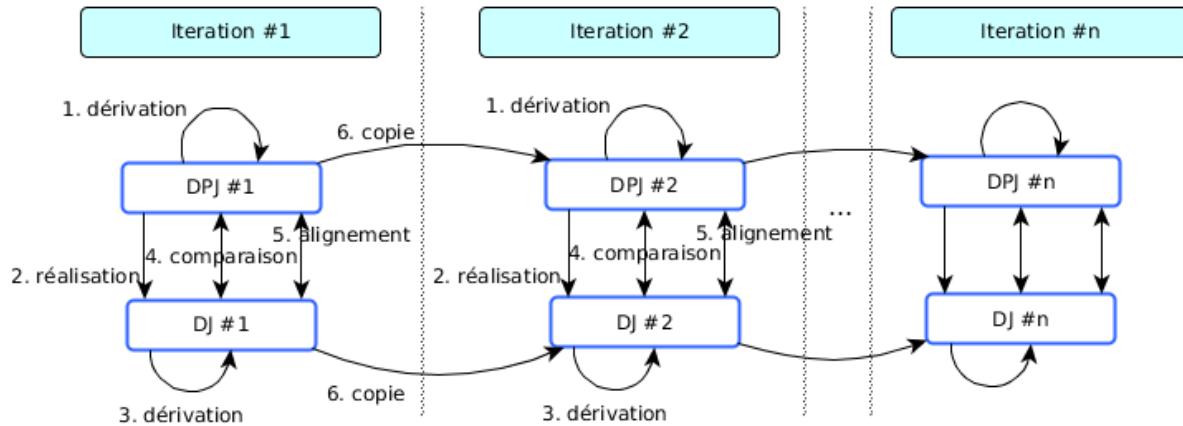


FIGURE 6.7 – Exemple de flot d'utilisation des opérations pour l'évolution conjointe des exigences de justifications et la production des justifications

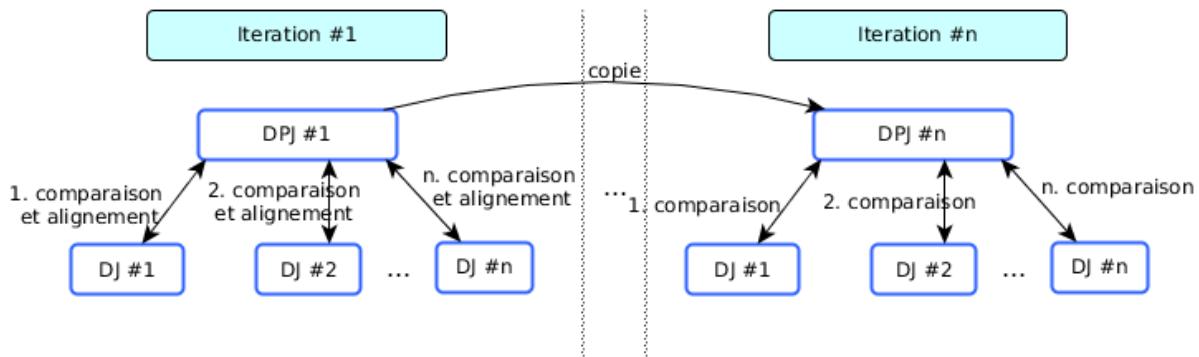


FIGURE 6.8 – Exemple de flot d'utilisation des opérations pour la structuration d'exigences de justifications

à l'exigence de justification, justifications impossibles à produire en réalité). Ainsi le tâtonnement entre les normes qui doivent être revendiquées, le moyen de répondre à ces exigences et les justifications qui ont été réellement produites est structuré par le contrôle du cycle de vie grâce à nos opérations pour assurer in fine la conformité entre le **DPJ** et le **DJ**.

Un manque de maturité peut également être présent dans des entreprises où des justifications sont déjà en place, mais où celles-ci ne sont pas structurées par des exigences de justifications (e.g., pratiques internes non formalisées, absence de norme). Nos opérations permettent de supporter des approches basées sur la pratique, en analysant des justifications existantes par des **DJs** pour ensuite les abstraire et capturer les exigences de justifications communes dans un **DPJ**. Dès lors, il faut successivement *comparer* et *aligner* le **DPJ** avec chaque **DJ** pour capturer des exigences de justifications communes au bon niveau de raffinement. En Figure 6.8, nous visualisons un tel flot. À chaque itération les **DJs** sont *comparés* et *alignés* sur un même **DPJ** jusqu'à ce que la *comparaison* de celui-ci avec chaque **DJ** ne donne que des modifications qui sont des raffinements.

6.3.2 Cycle de vie pour la production et révision des justifications sur la base d'exigences de justification structurées

Dans ce contexte, maîtriser le cycle de vie des diagrammes n'est pas un moyen de structurer avec confiance des justifications puisque le niveau de maturité de l'entreprise le permet déjà. Ainsi, l'intérêt de nos opérations réside alors dans leur faculté à gérer la révision des exigences de justifications et l'incrémentalité de la production des justifications.

Approche linéaire

Dans une approche linéaire (e.g., cycle en cascade, cycle en V), il est simple de déterminer la place des justifications dans le cycle de développement et quand il est nécessaire de produire une justification une fois et pour toute. En amont du projet, après une phase de *dérivation* pour établir le **DPJ** global pour le projet, la production des justifications tout au long du développement amène à remplir le **DJ** associé. De fait, le **DJ** est peu ou pas *dérivé* et il consiste seulement à raffiner les justifications pour atteindre les terminaux adéquats. Le **DPJ** quant à lui peut évoluer si par exemple une norme évolue. Dès lors, un *alignement* dirigé sur le **DJ** doit être conduit.

Approche itérative

Par une approche itérative (e.g., Agilité), il est plus difficile de juger de quand il est nécessaire de produire une justification et il est probable que cette justification sera amenée à évoluer au cours du reste du projet. Même si l'entreprise a une grande capacité à structurer les normes et pratiques internes, il est alors important de restreindre les justifications à ce qui est nécessaire et suffisant pour chaque itération. De ce fait, des **DPJs** sont anticipés à chaque itération et chaque **DJ** associé est produit sans perte de conformité. L'évolution est donc contrôlée lorsqu'à chaque itération le **DPJ** et le **DJ** sont *dérivés* conjointement, et la conformité est assurée par *comparaison* régulière. De telles évolutions conjointes sont fréquentes durant un projet où le besoin de produire au coup par coup seulement ce qui est nécessaire amène à voir la justification comme un processus incrémental au même titre que le développement. Chaque itération vient alors ajouter des justifications ou réviser des justifications existantes. Par nos opérations, nous avons bien montré ici que nous supportons une telle approche.

Notons que dans l'ensemble de ces flots, il existe toujours une phase itérative qui permet aux parties prenantes de mieux appréhender la structuration des exigences de justifications (e.g., itération pour établir les exigences de justifications en amont du projet, itération sur les exigences et la production durant le développement). Cela montre la réelle difficulté, pour des experts comme des débutants, d'atteindre le bon niveau d'abstraction. Ainsi ces opérations sont adaptées à plusieurs approches nécessitant d'introduire ou de faire évoluer de la justification.

6.4 Discussion

Dans ce chapitre, nous avons proposé une sémantique pour distinguer le niveau des exigences de justifications des justifications terminales à travers le concept de **DPJ**. Cette distinction nous a permis d'analyser et capturer les mécanismes d'évolution en jeu dans la justification (**C_{1.4}** - Supporter l'évolution des exigences de justifications). Cette évolution a été caractérisée de telle sorte que des opérations assurent la comparaison des justifications et l'analyse de la conformité durant les phases d'évolution (**C_{3.2}** - Faciliter la validation de justifications). Néanmoins, ces opérations bien que formalisées, reste à intégrer au sein de méthodologies propres. La section 6.3 ne donne que des pistes empiriques de ce que l'on a pu identifier durant l'utilisation de ces opérations sur nos cas d'études. Il serait intéressant de se rapprocher du domaine de l'*ingénierie des méthodes* BRINKKEMPER [1996] pour construire des composants de méthodologie autour de ces opérations, concevoir des méthodologies utilisant ses composants en caractérisant plus finement les contextes d'application de celles-ci, puis évaluer leur pertinence.

Par ailleurs, nous nous sommes intéressés ici à l'évolution unitaire des exigences de justifications ou des justifications elles-mêmes. Or, l'évolution d'un pas dans un **DJ** peut

amener à réviser les pas de justification qui se basent sur celui-ci. Ces évolutions “*par ricochet*” sont capturées sous forme de flot (c.f Section 6.3). L’impact de la modification d’une justification *a* sur la justification *b* sera vu comme deux opérations de *dérivation* successives (dérivation de *a* puis dérivation de *b*). Par ailleurs, une évolution peut également avoir un impact sur les artefacts de justifications et mener ainsi à les réviser ce qui échappe aux diagrammes. Ce type d’évolution n’a pas été caractérisée dans nos travaux, car ce concept d’artefact n’est pas capté par les *DJs* originels de Polacsek. Ceci à juste titre, car l’association d’un artefact à une assertion peut prendre beaucoup de forme et correspond davantage à une problématique de programmation. Dans la pratique, la modification d’un artefact impacte l’assertion associée ce qui nous ramène au point précédent. Il faut alors que les outils ou les humains, produisant ces artefacts, révisent les justifications impactées. A l’échelle des *DJs*, sur ces deux types d’évolution, nous pouvons seulement lever un drapeau permettant de mettre en évidence le besoin de réviser les pas de justification liés à ces évolutions en particulier en utilisant la fonctionnalité \mathcal{F}_3 (Est-ce qu’une assertion supporte un pas de justification ?).

Chapitre 7

Production guidée des justifications

« *Quality means doing it right even when no one is looking.* »

Henry Ford

Sommaire

7.1 Automatisation de la construction des justifications	62
7.1.1 Un environnement homme-machine pour aider à la production	62
7.1.2 Support de la formalisation	62
7.2 Automatisation de la vérification et validation des justifications	64
7.2.1 La revue des justifications : de l'approbation d'un document jusqu'à l'audit	64
7.2.2 Support de la formalisation	65
7.3 Justification Factory, vers une chaîne d'outils	66
7.4 Interactions avec d'autres formalismes	68
7.4.1 SACM et Diagramme de Justification (DJ)	68
7.4.2 I* et Diagramme de Justification (DJ)	71
7.5 Discussion	73

Dans ce chapitre, nous adressons les défis $C_{2.1}$ (Supporter le passage à l'échelle dans la production des justifications), $C_{2.2}$ (Proposer un environnement intégré pour l'utilisation des justifications), $C_{3.1}$ (Supporter une production des justifications en continu) et $C_{3.2}$ (Faciliter la validation de justifications). En effet, à partir du formalisme défini dans les chapitres précédents, nous allons montrer comment notre formalisme supporte des mécanismes de guide autant à la production des justifications qu'à la vérification et validation de celles-ci. Nous proposons ainsi une architecture facilitant l'interaction entre notre sémantique et les outils ainsi qu'avec d'autres formalismes. Il nous est apparu intéressant de montrer comment notre formalisme permet 1) d'automatiser la production des justifications et 2) d'apporter de l'interaction avec les formalismes i* et SACM.

7.1 Automatisation de la construction des justifications

7.1.1 Un environnement homme-machine pour aider à la production

Comme identifiée précédemment, la production des justifications est souvent une activité incrémentale où chaque partie prenante va pas à pas construire les justifications au regard des exigences de justifications. Les cycles itératifs engendrés par le manque de maturité ou un choix de cycle de développement amènent à construire plusieurs fois de mêmes justifications dans des versions révisées. Ceci entraîne une masse de justifications conséquentes à produire, mais aussi à organiser. En s'inspirant de l'agilité, l'automatisation de ce qui peut l'être est un levier pour le passage à l'échelle dans un tel contexte.

Pour aider à produire des justifications, il existe des outils classiques (*e.g.*, analyseurs de code pour la qualité du code, répertoires de binaires pour la gestion du cycle de vie des dépendances externes, logiciels spécifiques pour la documentation technique). Habituellement, les artefacts de justifications, qui sont extraits de ces outils, sont gérés par un humain qui va les lier aux justifications adéquates. Bien que basée sur des outils, l'intervention humaine reste donc indispensable pour l'intégration des justifications. L'automatisation que nous proposons doit ainsi non pas reposer sur la capacité à créer un artefact de justifications, mais bien sûr la capacité à assembler ces artefacts dans des justifications structurées. Par ailleurs, ces outils n'ont pas vocation à embarquer des activités de justification, il faut donc être capable d'interagir avec eux de telle sorte que le processus de justifications soit transparent pour eux.

D'autre part, là où l'automatisation n'est pas possible, c'est-à-dire là où une expertise humaine est nécessaire, il doit être possible de facilement retrouver et construire des justifications à partir des artefacts produits par un outil. En ce point, il y a donc de l'automatisation à acquérir non pas sur la construction, mais sur la récupération des artefacts de justifications dans un environnement intégré.

7.1.2 Support de la formalisation

Le formalisme que nous avons proposé jusqu'ici propose un ensemble de définitions et de fonctionnalités \mathcal{F} qui vont nous permettre d'adresser ces problématiques. Tout d'abord, la production des justifications repose sur des exigences de justifications. Il s'agit ainsi au préalable de disposer d'un DPJ les formalisant. Dans ce DPJ, nous allons nous reposer sur une spécialisation des stratégies en *stratégies humaines* et *stratégies computationnelles* pour distinguer ce qui peut être confié à un outil de ce qui requiert un humain. Dans un pas de justification, une stratégie peut être appliquée dès lors que ses supports sont produits. Indifféremment des stratégies humaines ou computationnelles, c'est donc la coopération entre individus et/ou outils qui permet la construction d'une pas de justification. De ce fait les assertions sur lesquelles reposent un pas de justification doivent être

collectées pour s'adapter aux temps différents que peuvent avoir les parties prenantes et ainsi assurer l'automatisation de la construction. Ainsi il est nécessaire de disposer d'une base d'assertions permettant de regrouper l'ensemble des assertions disponibles.

Définition 7.1.1 : Base d'assertions

Soit \mathcal{B} un ensemble d'assertions et jd un DJ.

Nous définissons deux fonctions $\mathcal{B}_{unused}(jd)$, l'ensemble d'assertions inutilisées dans jd et $\mathcal{B}_{used}(jd)$, l'ensemble d'assertions déjà utilisées dans jd ainsi :

- $\mathcal{B}_{unused}(jd) = \{s \in \mathcal{B}, s \notin jd\}$
- $\mathcal{B}_{used}(jd) = \{s \in \mathcal{B}, s \in jd\}$

Par cette définition $\forall jd, \mathcal{B} = \mathcal{B}_{unused}(jd) \cup \mathcal{B}_{used}(jd)$ et $\mathcal{B}_{unused}(jd) \cap \mathcal{B}_{used}(jd) = \emptyset$.

Ainsi grâce à cette définition, nous pouvons connaître à tout instant, les assertions, déjà présentes dans un pas de justification, qui peuvent être réutilisées pour un autre pas (e.g., conclusion utile en tant que support d'un autre pas, évidence utile dans deux pas de justifications différents) et les assertions pas encore utilisées pour justifier. Nous dissocions alors la création d'une assertion qui est ajoutée à \mathcal{B} qui amènera cette assertion dans $\mathcal{B}_{unused}(jd)$ de son utilisation à partir \mathcal{B} pour justifier au sein d'un pas de justification qui amènera cette assertion dans \mathcal{B}_{used} . Par ce mécanisme nous décorrélons la construction des assertions à partir des artefacts de justifications de la construction des pas de justifications à partir des assertions. Ainsi les outils n'ont qu'à connaître comment construire une assertion et non un pas de justification. Ceci permet de déléguer à un outil central et dédié, que nous nommerons *Justification Factory*, la construction des justifications sous forme de pas de justification inséré dans le ou les DJs appropriés. Pour ce faire, nous définissons un *système de justification* comme le concept qui permet de guider la justification au sein de cette usine (*Justification Factory*).

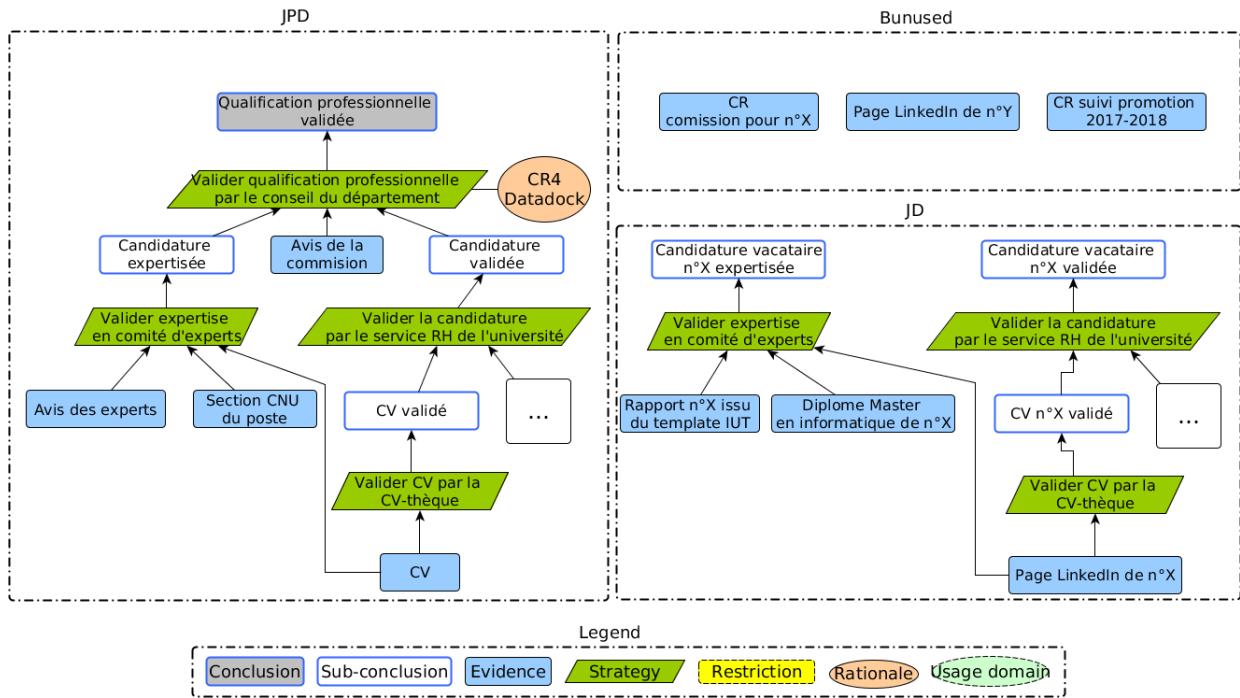
Grâce à nos deux fonctions dans la définition précédente, quand l'ensemble des justifications à produire l'ont été, il est plus facile de se questionner sur si les justifications ont été produites de façon nécessaire et suffisante. En effet, si $\mathcal{B}_{unused}(jd) \neq \emptyset$ cela revient à dire que ces assertions ont été produites pour rien ou qu'il y a un manquement dans les justifications.

Définition 7.1.2 : Système de Justification (SJ)

Un **Système de Justification (SJ)** js est composé d'un DPJ jpd qui définit les exigences de justifications, d'un DJ jd qui structure les justifications produites et de $\mathcal{B}_{unused}(jd)$. Ainsi $js = < jpd, jd >$

Fonctionnalité \mathcal{F}_4 : Production automatique de pas de justification

Ainsi dans un **Système de Justification (SJ)** $js = < jpd, jd >$, à chaque ajout d'une assertion terminale a par un outil dans \mathcal{B} , la *Justification Factory* va faire appel à la fonctionnalité \mathcal{F}_2 (Que peut-on justifier ?) avec la base d'assertions $A = \mathcal{B}$ et l'ensemble des pas de justifications existants $S = \{s \in jpd\}$ pour savoir qu'est-ce qu'il est possible de justifier avec cet ajout. Si $\exists s' \in S$ alors s' est ajouté à jd .


 FIGURE 7.1 – Exemple de **Système de Justification (SJ)** sur Datadock

Exemple fil rouge 7.1.1 : SJ sur Datadock

En Figure 7.1, nous montrons un exemple de **SJ**. Dans ce cas, le **DJ** est en cours de construction et le pas de justification contenant la stratégie *Valider qualification professionnelle par le conseil du département* réunit les conditions pour être appliqué. En effet $\mathcal{B}_{unused}(jd)$ contient la dernière assertion manquante pour construire ce pas. Ainsi \mathcal{F}_4 peut être utilisé pour construire ce pas.

7.2 Automatisation de la vérification et validation des justifications

7.2.1 La revue des justifications : de l'approbation d'un document jusqu'à l'audit

Une fois des justifications produites et structurées au sein d'un **DJ**, il s'agit d'utiliser ces justifications pour 1) vérifier que ce qui devait être produit là bien été et 2) valider qu'au regard des exigences de justification, ces justifications sont suffisantes. Les experts qualité d'une entreprise attestent de ces aspects par des revues et des audits internes ainsi que l'organisme de certification lors d'audits réguliers. Il faut donc rendre compréhensibles et utilisables ces justifications. En restant proche de la notation originelle des **DJs** nous assurons une utilisabilité par les experts qualité habitués à ce type de visualisation. Néanmoins, un **DJ** peut être très gros et la visualisation sous forme de diagramme atteint ses limites. Par ailleurs, l'aéronautique s'intéresse fortement à l'utilisation de diagrammes de cas d'assurance structurée **HOLLOWAY [2013]**; **KELLY et WEAVER [2004b]** et **DJ POLACSEK et collab.**, mais d'autres domaines critiques comme le médical reste sur une structuration des justifications sous forme documentaire. Il est alors difficile de se limiter pour la Vérification et Validation (V&V) à une simple visualisation des **DJs**. Il nous faut donc définir des aides automatiques à la V&V des **DJs**.

7.2.2 Support de la formalisation

Grâce à l'utilisation de la relation \mathcal{R} et la notion de *terminal* nous avons préalablement poser les concepts permettant d'intégrer la V&V des justifications directement dans les **DJs**. Nous donnons ainsi une définition d'un **SJ complet**.

Définition 7.2.1 : Système de Justification (SJ) complet

Un **SJ** $js = < jpd, jd >$ est dit *complet* quand

- jd est terminal
- $jd \mathcal{R} jpd$

Exemple fil rouge 7.2.1 : SJ complet sur Datadock

Dans la Figure 7.1, après avoir utilisé \mathcal{F}_4 pour construire le pas de justification contenant la stratégie *Valider qualification professionnelle par le conseil du département*, les conditions son réunies pour attester que le **SJ** est *complet*. En effet, chaque pas du **DPJ** a un pas conforme dans le **DJ** et chaque pas du **DJ** est terminal.

Une fois un **SJ** complet, il s'agit de le rendre utilisable pour la traçabilité et les audits. Usuellement, la construction d'un document appelé *Master file*, une table des matières navigables vers l'ensemble des justifications, permet une vue globale des justifications. Ce *Master file* est utilisé comme point d'entrée dans le **SMQ** lors des audits. Pour visualiser un **SJ** complet sous forme documentaire, il s'agit de mettre à plat le **DPJ** et **DJ** pour le représenter sous forme d'une table de *mapping* entre exigences de justifications du **DPJ** et justifications du **DJ**.

Définition 7.2.2 : Système de Justification (SJ) sous forme matricielle

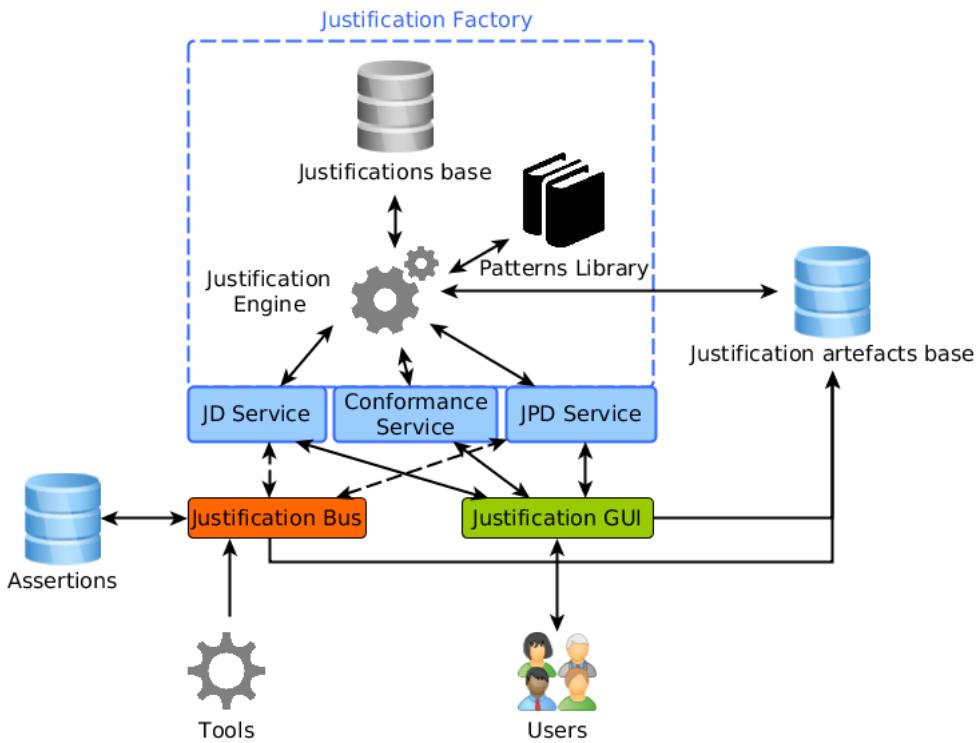
Soit mf un ensemble de couple de patrons de justification et de pas de justification terminaux noté mf . Ainsi, soit un **SJ** complet $js = < jpd, jd >$, $\forall s_{jpd} \in jpd, \exists s_{jd} \in jd$ tel que $s_{jd} \mathcal{R} s_{jpd}$ et $(s_{jpd}, s_{jd}) \in mf$.

Un *Master file* correspond finalement à mf . À partir de cet ensemble mf , il suffit de le projeter sur n'importe quel *template* de *Master file* e.g., tableur excel, page web.

Exemple fil rouge 7.2.2 : Master file de Datadock

Pour Datadock, construire le *Master file* consiste à reprendre le tableau des indicateurs et éléments de preuves de la Figure 1.1. Finalement, cela consiste à avoir un tableur Excel respectant le template donné par cette figure. Ainsi il suffit de mettre la bonne exigence de justification du **DPJ** sur chaque indicateur et la bonne justification du **DJ** sur chaque élément de preuve à l'aide de la forme matricielle.

Grâce à l'automatisation de la V&V et la génération de la visualisation documentaire des justifications, il est plus simple de gérer tout au long du cycle de vie du projet les justifications. En effet, comme vu dans le Chapitre 6, durant un projet, les exigences de justifications comme les justifications sont amenées à évoluer. Ainsi, vérifier et valider des justifications de façon automatique permet de gagner du temps et de la confiance dans ces activités qui sont fastidieuses et sources d'erreur.

FIGURE 7.2 – Écosystème de *Justification Factory*

7.3 Justification Factory, vers une chaîne d'outils

Le méta-modèle présenté et enrichi durant les chapitres précédents est la pierre angulaire du moteur de justification que nous avons développé. Une version détaillée est donnée Annexe A.1. Ce moteur a été exposé au sein d'une usine, appelé *Justification Factory*, qui a été ensuite étendue en exposant des services pour tendre vers le formalisme pour l'automatisation présenté en Section 7.1 et 7.2. Cet écosystème, introduit en Figure 7.2, est présenté pas à pas dans cette section.

Cette usine expose trois services¹, permettant de construire et enregistrer des patrons dans le DPJ d'un SJ (*JPD Service*), de construire le DJ d'un SJ (*JD Service*) et de vérifier des propriétés sur ces diagrammes (*Conformance Service*).

Justification services

Grâce à cette approche orientée service et la concrétisation du formalisme dans le méta-modèle, la connexion avec d'autres outils est facilitée. En effet, il est usuel d'utiliser des services pour exposer de façon interopérable des fonctionnalités. Ces fonctionnalités sont exposées sous forme de méthodes ou ressources utilisables en interagissant avec des objets métiers bien définis, ici les concepts de notre méta-modèle. L'ensemble de ces services étant sans état, c'est-à-dire qu'à chaque appel il n'y a pas de *mémoire* des précédents appels du service, il faut préciser le contexte, la session à utiliser. Il faut donc pouvoir interagir avec en spécifiant, à chaque échange, le SJ visé.

Le *JPD service* permet de renseigner des exigences de justifications à travers des patrons de justifications, mais également de les structurer à travers le DPJ du SJ ciblé. C'est également le service qui, à partir d'assertions, liste les patrons utilisables (\mathcal{F}_2 - Que peut-on justifier ?) ou, à partir d'une assertion, donne les patrons qui permettent de la justifier dans un SJ (\mathcal{F}_1 - Comment justifier une conclusion ?).

1. au niveau technologique, ces services sont des services web REST et le moteur sous-jacent est développé en Java

Le *JD Service* permet de construire des pas de justifications à partir des patrons et de les insérer dans le **DJ** du **SJ** ciblé. Il permet également de vérifier si une assertion supporte un pas de justification (\mathcal{F}_3 - Est-ce qu'une assertion supporte un pas de justification ?) et de récupérer ses justifications sous forme de **DJ**, mais également sous forme transformée comme pour le *Master file*.

Le *Conformance Service* permet de vérifier et valider les justifications du **DJ** au regard des exigences de justification du **DPJ**. C'est également ce service qui gère les opérations formalisées en Chapitre 6 permettant de contrôler le cycle de vie des **DJs**.

Pour plus de détails, l'interface de ces services est donnée en Annexe A.2.

Justification bus

Pour masquer la justification aux outils et automatiser la construction des justifications à partir d'assertions, un bus d'évènements est ajouté. Il permet aux outils de renseigner des assertions et ensuite il orchestre la fonctionnalité \mathcal{F}_4 (Production automatique de pas de justification) à travers le *JD Service* et le *JPD Service*. Puisqu'il peut y avoir plusieurs **SJ** en parallèle dans la *Justification Factory*, ce bus stocke dans une base les assertions qui lui ont été fournies pour aider à construire pour chaque **SJ**, \mathcal{B}_{unused} lors de l'orchestration. Comme les artefacts de justifications peuvent venir de diverses sources, lors de l'ajout d'une assertion par un outil, il va également stocker dans une base dédiée ces artefacts dans le but d'en centraliser l'accès et la pérennité.

L'algorithme d'orchestration est montré en Annexe A.3 sous forme de diagramme de séquence entre le *justification bus* et les services de la *Justification Factory*.

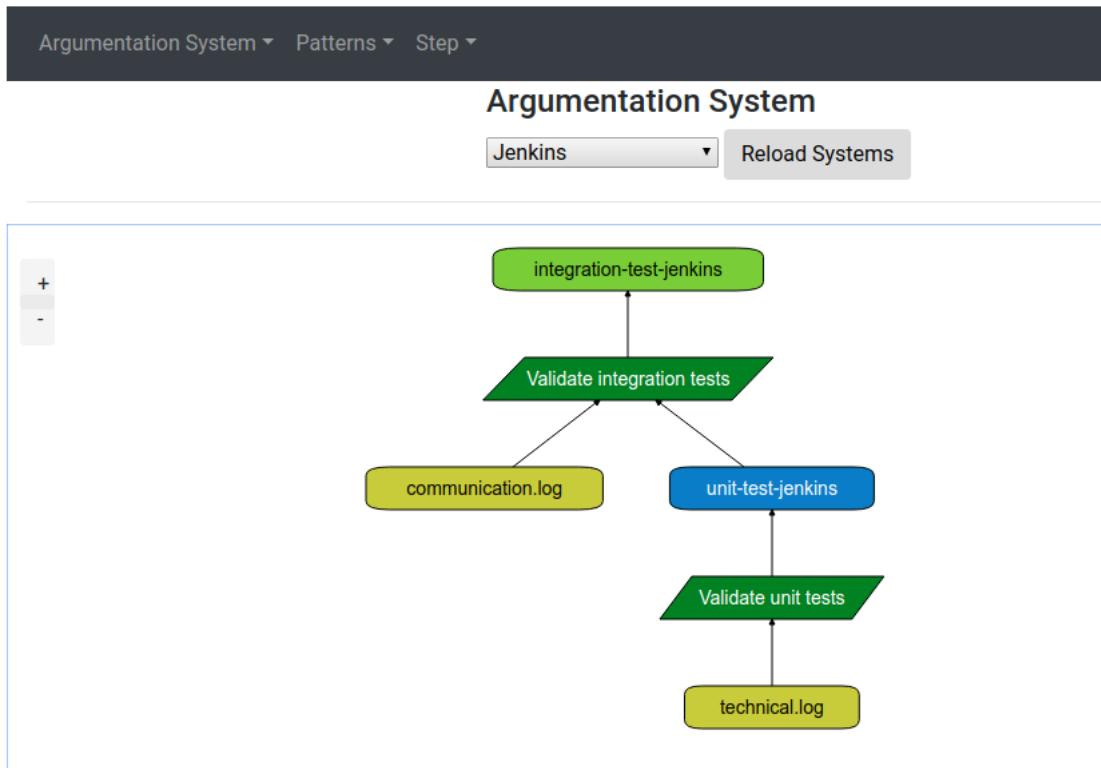
Justification GUI

Pour faciliter la création de patrons de justifications et de leur structuration dans un **DPJ**, nous avons développé un outil *What You See Is What You Get*, nommé *ADEV*. Cet outil permet de visualiser des patrons de justifications et des **DPJs** comme dans la notation de Polacsek.

Cet outil permet également de renseigner des pas de justifications pour permettre à ceux basés sur une stratégie humaine d'être ajoutés à un **DJ** dans un environnement visuel. En fine, cet outil est aussi utilisé pour permettre aux experts qualité de faire la revue des justifications produites.

Cette suite d'outils basée sur notre sémantique permet ainsi de proposer un environnement consacré aux activités de justification. Par cet écosystème que nous avons mis en place, il est ensuite possible d'interagir avec d'autres outils en adaptant la production de leurs artefacts en assertions que *Justification Factory* est capable de traiter pour construire des justifications. Nous montrons des exemples d'extensions d'outils dans ce sens en Partie III. Une capture d'écran de l'outil est donnée en Figure 7.3. Notons que cette visualisation conserve bien la notation originelle des **DJs**.

Par ailleurs, cette approche *justification as a service* permet également de mettre en place des collaborations avec d'autres formalismes. Nous donnons deux collaborations possibles dans la section suivante.

FIGURE 7.3 – GUI pour *Justification Factory*

7.4 Interactions avec d'autres formalismes

Que ce soit avec d'autres formalismes pour la justification comme GSN, CAE ou SACM ou des formalismes traitant des aspects connexes à la justification comme le domaine de l'ingénierie des exigences, notre approche est un noyau pour permettre de mettre en place des collaborations. En effet, la sémantique formelle que nous avons définie nous a permis de proposer un méta-modèle qui peut être utilisé comme un moyen d'interaction avec d'autres formalismes. Pour les autres modèles de justifications cela permet de définir des transformations permettant d'adopter des notations graphiques particulières et ainsi faire profiter *Justification Factory* de *GUI* adaptée suivant le contexte. Pour le domaine de l'ingénierie des exigences, cela permet d'enrichir la justification par les liens implicites qui lient une exigence de justification à son moyen de réalisation et inversement permet au domaine de l'ingénierie des exigences de vérifier l'atteinte de propriétés non fonctionnelles comme l'est la justification. Nous détaillons l'étude de ces deux cas dans cette section.

7.4.1 SACM et Diagramme de Justification (DJ)

Puisque nous voulons proposer notre formalisme comme une approche transposable pour la justification en général, il nous apparaît intéressant de pouvoir traduire des *DJs* vers d'autres formalismes. Ceci a pour but i) de conserver des notations spécifiques déjà existantes *C_{1.1}* (Préserver la sémantique et les représentations connues adaptées à la justification) et ii) d'interagir avec les formalismes existants afin, à l'aide des fonctionnalités de notre formalisme, de pouvoir enrichir notre approche.

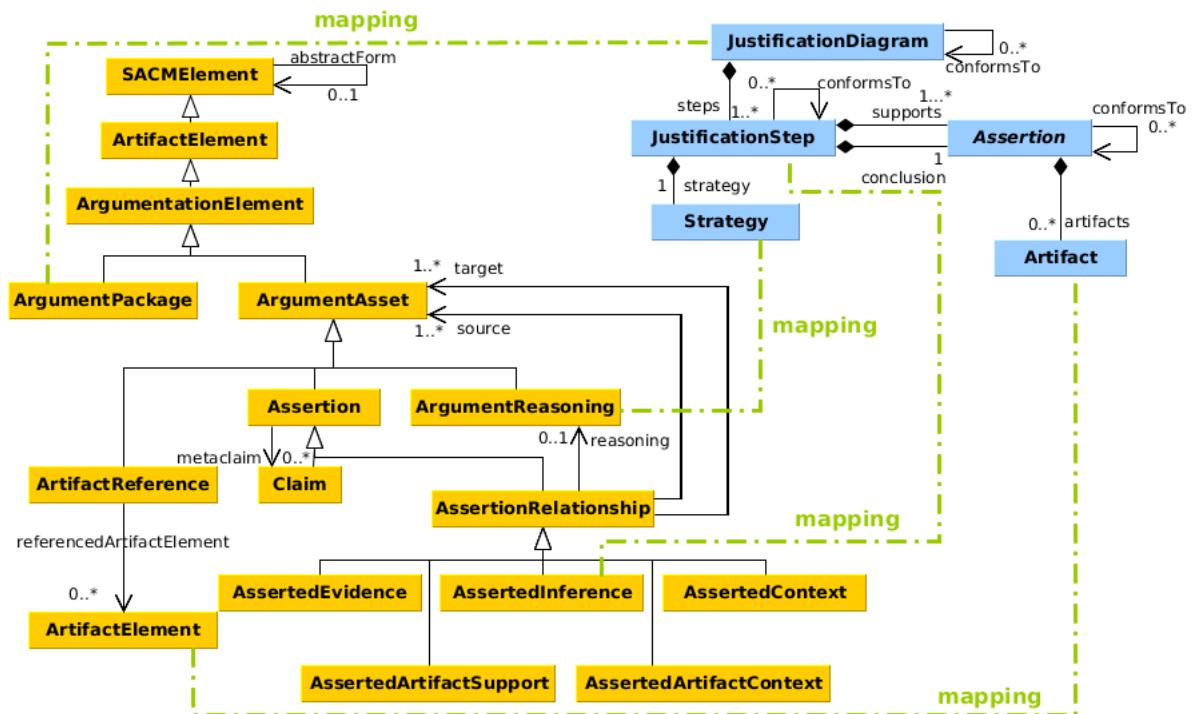


FIGURE 7.4 – Extrait du méta-modèle de SACM et les liens vers le méta-modèle de DJ

Analyse des deux modélisations

En utilisant le méta-modèle de l'OMG et la modélisation des **DJs** présentés en Figure 5.2, nous proposons une transformation de modèle depuis un **DJ** vers SACM.

Dans un **DJ**, quand une assertion est supportée par du raisonnement, cela apparaît en tant que conclusion dans un **JustificationStep** tandis que dans SACM le raisonnement est exprimé à travers la dérivation de **AssertionRelationship**. Cette réification du raisonnement dans **DJ** limite les formes de raisonnement possibles contrairement à SACM. Cependant, ceci permet d'associer une sémantique formelle avec une notation proche du schéma de Toulmin. En particulier, un pas de justification ne peut pas "inférer" plusieurs conclusions dans un **DJ**, contrairement à SACM. Ceci amène à considérer du chaînage partiel entre justifications (e.g., utilisation d'une partie de conclusion d'une justification précédente). Ces choix possibles dans l'utilisation des *sorties* d'une justification cachent le raisonnement qui mène à cette ségrégation et donc amène de l'implicite. Ainsi, par les **DJs**, chaque étape de raisonnement est rendue explicite contrairement à SACM.

De plus, dans SACM, *abstractForm* est défini comme "*an optional reference to another abstract SACMElement to which this concrete SACMElement conforms [...] When +abstractForm is used to refer to another SACMElement, +isAbstract of the SACMElement is false, and the +isAbstract of the referred SACMElement should be true*". Ceci limite l'abstraction des concepts de SACM à un seul niveau. Dans **DJ**, la référence *conformsTo* correspond à la relation \mathcal{R} . Ceci permet à un concept de justification de se conformer à plusieurs éléments simultanément comme discuté dans la Définition 5.1.1. De plus, ceci ne réduit pas le raffinement à un seul niveau, ce qui est très utile pour la construction effective et l'usage des **DJs**.

Transformation de modèle de **DJ** vers SACM

Par conséquent, il est possible de transformer un modèle de **DJ** en limitant la relation de conformité à un niveau d'abstraction seulement.

Certains mapping entre SACM et DJ restent triviaux :

Strategy en ArgumentReasoning, Assertion en Assertion, Artifact en Artifac-tElement, JustificationDiagram en ArgumentPackage (*cf.* Figure 7.4).

D'autres, comme discuté dans la section précédente, sont moins naturels : Justifi-cationStep en AssertedInference

Nous avons implémenté cette transformation en utilisant *Eclipse Modeling Frame-work (EMF)* STEINBERG et collab. [2008] : la modélisation des 2 modèles avec *Ecore* et les règles de transformation avec *ATL* JOUAULT et collab. [2008].

Listing 8.1 montre 2 de ces règles de transformation. Ici, la première règle décrit com-ment transformer un JustificationDiagram en ArgumentPackage la seconde, un Jus-tificationStep en AssertedInference. Le projet complet et exécutable peut être trouvé ici : github.com/JustificationFactory/JustificationDiagramModelToSACM. Actuellement, dû aux limites de SACM énoncées dans la section précédente, il n'est possible de trans-former que des sous-parties de SACM vers des modèles DJ . Notre usage des DJs dirigé par les besoins industriels et le besoin d'une sémantique claire nous amène à favoriser la transformation vers SACM sans interdire la transformation inverse.

Listing 7.1 – Excerpt of DJ to SACM transformation rules using ATL

```
rule JustificationDiagram2ArgumentPackage {
  from
    jd: JD!JustificationDiagram (
      not jd.isAbstract()
    )
  to
    package: SACM!ArgumentPackage (
      content <- jd.name,
      argumentAsset <-
        jd.steps ->
          collect(e | thisModule.Step2Inference(e))
    )
}
lazy rule Step2Inference{
  from
    s: JD!JustificationStep
  to
    inference: SACM!AssertedInference(
      content <- s.name,
      reasoning <- thisModule.
        Strategy2ArgumentReasoning(s.strategy),
      target <- thisModule.
        Assertion2Assertion(s.conclusion),
      source <- s.supports ->
        collect(e | thisModule.
          Assertion2Assertion(e)
    )
}
```

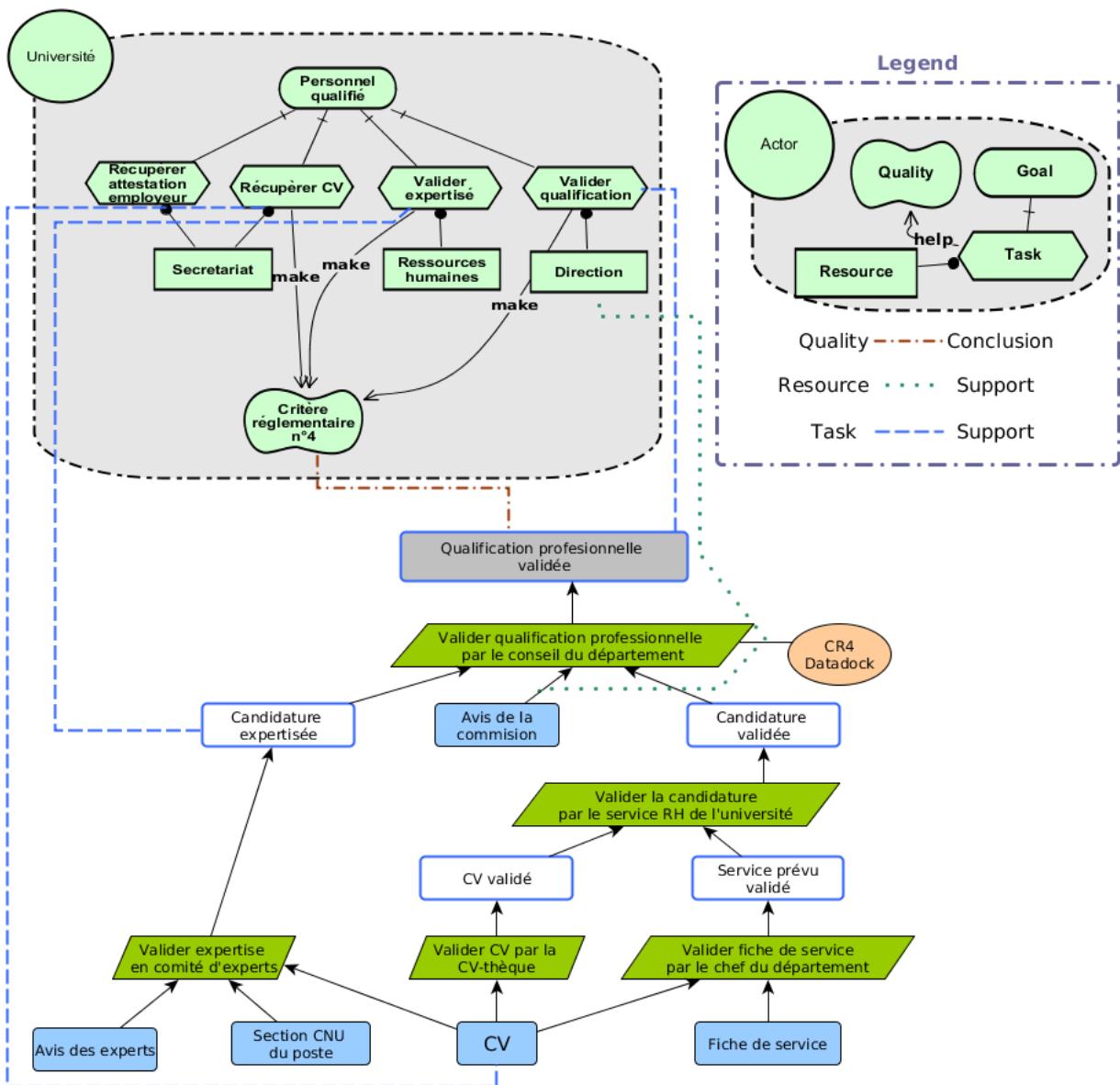


FIGURE 7.5 – Exemple sur Datadock de collaboration entre i* et DJ

7.4.2 I* et Diagramme de Justification (DJ)

Approche pour la justification sur les exigences qualité

Différentes sortes d'exigences peuvent être représentées à travers i* ainsi que les liens entre elles; par exemple, nous pouvons exprimer que la conformité avec la norme ISO 9001 contribue à l'atteinte de la qualité de service. I* capture les contributions pour l'atteinte d'une exigence en qualifiant ce qui la *make*, *help*, *hurt*, *break*. Cependant, comme dit en Chapitre 3, i* ne capture pas le raisonnement, la justification derrière l'attestation de l'atteinte d'une exigence.

Nous proposons de combler ce fossé, en nous intéressant à l'atteinte d'exigences qualité. Dans ce contexte, le but est d'utiliser les **DJs** pour justifier que le niveau de qualité a bien été atteint.

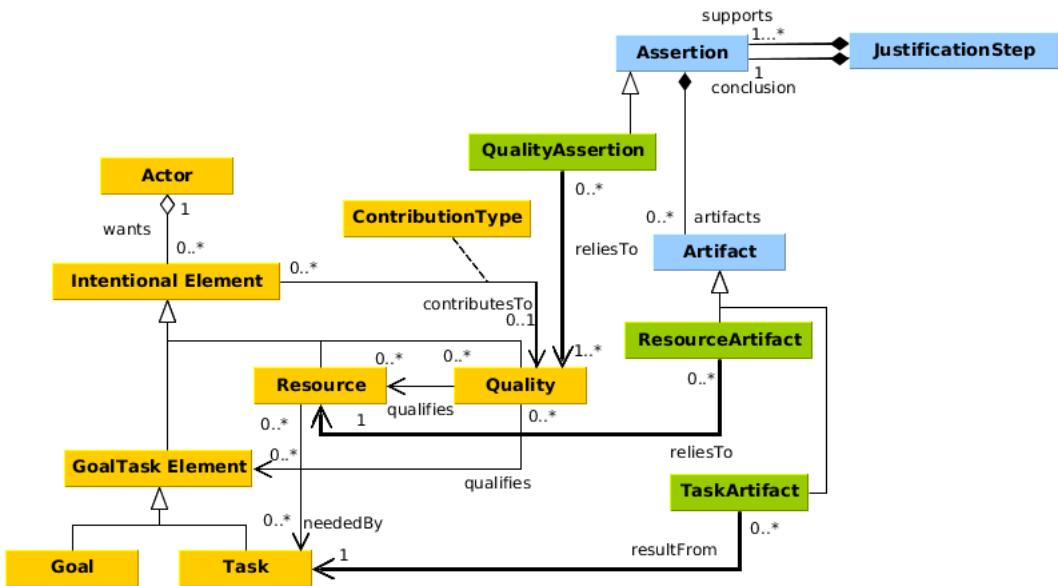


FIGURE 7.6 – Points d’interaction entre DJ métamodèle (en bleu) et i* 2.0 métamodèle (en jaune)

Analyse sémantique entre représentations

Au niveau sémantique, nous avons donc analysé quels étaient les concepts des deux langages que nous pourrions relier. Nous proposons de visualiser ces mapping en illustrant sur l'exemple donné en Figure 7.5. Cette figure présente la collaboration entre un diagramme i* et un DJ . Le diagramme i* présente les exigences à atteindre : l'objectif *personnel qualifié* et prend en compte également une qualité : *Critère réglementaire n°4*. L'atteinte de cette qualité est justifiée par un DJ . Des liens, représentés en pointillés, sont créés entre le diagramme i* et le DJ pour aider et renforcer la justification. Pour des raisons de lisibilité, nous représentons des liens entre assertions et les éléments des *Resource* et des *Task* qui bypass les artefacts de justification. Nous donnons un autre exemple d'un telle collaboration en Annexe A.4. Le diagramme i* donne un point de vue processus pour atteindre des objectifs et qualité tandis que les DJs donne un point de vue justification pour renforcer la confiance dans l'atteinte de ces qualités.

Interactions entre métamodèles

Dans les diagrammes i*, Task représente les actions qu'un Actor veut exécuter pour atteindre une Quality ou un Goal. Par conséquent, dans le DJ correspondant, ceci nécessite des assertions de justifications qui justifient de la bonne exécution d'une tâche. Quand une Resource (une entité physique ou informationnelle) est requise dans i* pour réaliser une Task, nous attendons dans le DJ , une Assertion qui s'appuie sur l'utilisation de cette Resource. Les liens entre ces éléments de justification doivent correspondre à des stratégies qui expriment le raisonnement sur ces éléments.

Pour matérialiser ces liens, nous avons étendu le métamodèle des DJs pour faire correspondre les concepts des DJs sur les éléments intentionnels d'i*. En Figure 7.6, nous proposons l'intégration de ces deux approches au niveau métamodèle. Une QualityAssertion dans un DJ (respectivement ResourceArtifact , TaskArtifact) réfère à une Quality dans i* (respectivement Resource, Task). Pour chaque exigence qualité attendue représentée par une Quality, une QualityAssertion est créée et liée à celle-ci. Nous nous focalisons ensuite sur tous les éléments en relation contributesTo méliorative (*make, help*) dans le diagramme i* liée à cette Quality. Ces éléments, suivant leur type, sont à leur tour liés à des éléments dans le DJ . Par incrémentalité nous atteignons les feuilles des

différents diagrammes.

La sémantique des **Diagrammes de Justification (DJs)**, un support pour la confiance

Nous pouvons ensuite utiliser les fonctionnalités de notre formalisme présentées en Section 5.2.4.

Suivant un DJ et un diagramme i*, nous pouvons vérifier que pour toutes les exigences de qualité attendues (r), il y a un patron dont la conclusion se réfère à r (\mathcal{F}_1) et que chaque élément intentionnel qui contribue à r est lié à une assertion qui supporte directement ou non ce patron (\mathcal{F}_3).

Exemple fil rouge 7.4.1 : I* avec \mathcal{F}_1 et \mathcal{F}_3

Par exemple, dans la Figure 7.5, pour justifier la qualité *Critère réglementaire n°4*, un DJ a été conçu. Nous pouvons vérifier que chaque tâche qui font (*make*) cette qualité sont présentes dans le DJ pour supporter la justification de l'atteinte de cette qualité.

Suivant un ensemble d'assertions lié à un *ResourceArtifact* ou un *TaskArtifact*, nous pouvons également automatiquement construire ou aider la production ce qui peut être justifié (\mathcal{F}_2).

Exemple fil rouge 7.4.2 : I* avec \mathcal{F}_2

Dans l'exemple, après avoir atteint les assertions *Candidature expertisée* et *Candidature validée*, le pas de justification contenant la stratégie *Valider qualification professionnelle par le conseil du département* repose sur la *Direction* qui doit émettre un *Avis de la commission*.

L'alliance entre ces deux types de diagrammes renforce la confiance que l'on peut avoir dans les DJs en termes de consistance et complétude.

7.5 Discussion

Dans ce chapitre, nous avons proposé d'utiliser notre sémantique pour automatiser la production des justifications en se basant sur les exigences de justifications, mais également de vérifier et valider la conformité des justifications $\mathcal{C}_{3.2}$ (Faciliter la validation de justifications). Cette automatisation nous a ainsi amenés à formaliser le flot de création d'un pas de justification et donc à supporter l'incrémentalité dans la production des justifications $\mathcal{C}_{3.1}$ (Supporter une production des justifications en continu). Ces différents mécanismes nous ont permis d'adresser le passage à l'échelle $\mathcal{C}_{2.1}$ (Supporter le passage à l'échelle dans la production des justifications). Par l'architecture proposée, issue de la sémantique que nous avons définie, nous initions un environnement pour la justification permettant des interactions avec d'autres outils $\mathcal{C}_{2.2}$ (Proposer un environnement intégré pour l'utilisation des justifications) et également d'autres formalismes comme i* et SACM $\mathcal{C}_{1.1}$ (Préserver la sémantique et les représentations connues adaptées à la justification). Finalement, nous avons montré comment intégrer des justifications validées dans le SMQ afin d'assurer la transparence de notre outil.

La contribution que nous avons mise en avant ici permet une construction et une validation des justifications très guidées. En effet, la structuration sous forme de diagramme, malgré le raffinement que nous avons défini, fige les exigences de justification et ne permet pas l'introduction de variantes qui seraient non conformes entre elles. Par exemple, si

une exigence de justifications dans un DPJ est finalement inatteignable dans la réalité DJ, il faut alors réviser le DPJ suivant avec les opérations que nous avons définies. Ce processus peut être long et complexe dans le cadre d'un SJ où l'on ne cherche pas à raisonner sur les exigences de justifications (e.g., justifications non basées sur une norme, construction de justifications dans le but d'expliquer). Dans ce cadre-là, la validation de justifications ne se fait pas sur la base d'un DPJ, mais sur la base d'un objectif à atteindre. Cet objectif est finalement une assertion qui à l'aide de pas de justifications va être atteinte. Dans ce cas, nous n'avons pas de connaissances a priori sur l'enchaînement de pas de justifications à faire et donc il n'est pas possible de les structurer dans un DPJ. Dans de tels cas, il nous faudrait alors disposer d'un ensemble de pas de justifications utilisables pour atteindre cet objectif. À partir de cet ensemble, on peut construire un DJ. La validation du DJ consiste alors à vérifier si cet objectif a été atteint. Bien que cet aspect soit présent dans l'implémentation de nos travaux, nous n'avons pas souhaité le formaliser. En effet, la construction seule de pas de justifications sur la base d'un ensemble de justifications *à plat* permet difficilement de guider la production. Dès lors que l'ensemble est grand, cela nécessite la maîtrise d'un grand nombre de pas de justification pour atteindre cet objectif. Pour nous, pour guider cette construction, il faudrait attacher un système de contraintes sur cet ensemble de pas de justifications pour guider la sélection des pas grâce à ses contraintes (e.g., empêcher l'enchaînement de deux pas de justifications, empêcher l'utilisation d'un pas si un autre a déjà été utilisé). La validation du DJ consiste alors en plus de vérifier si l'objectif a été atteint de vérifier l'ensemble des contraintes. Il y aurait ainsi 2 types de systèmes de justifications possibles en fonction du but rechercher par ces justifications.

Troisième partie

Validation

Chapitre 8

Application pour la certification ISO 13485 dans une entreprise de technologies médicales

« *C'est dans la pratique qu'il faut que l'homme prouve la vérité.* »

Karl Marx

Sommaire

8.1 AXONIC : Contexte des technologie médicales	78
8.2 Environnement technique	79
8.2.1 Redmine, un SMQ centré sur l'Agilité	79
8.2.2 Jenkins, une plateforme d'intégration continue pour le système tout en entier	79
8.3 Elicitation des exigences de justifications pour IEC 62304	80
8.3.1 Conduite de l'étude	80
8.3.2 Analyse de l'étude	83
8.3.3 Limites	83
8.4 Production automatisée des justifications pour la IEC 62304	84
8.4.1 Développement des outils pour l'intégration de <i>Justification Factory</i> et Jenkins	84
8.4.2 Conduite de l'étude	86
8.4.3 Analyse de l'étude	87
8.5 Des Diagrammes de Justification (DJs) vers l'intégration dans le SMQ d'AXONIC	88
8.5.1 Développement des outils pour l'intégration de <i>Justification Factory</i> et Redmine	88
8.5.2 Conduite de l'étude	89
8.5.3 Analyse de l'étude	90

8.1 AXONIC : Contexte des technologie médicales

AXONIC est une jeune entreprise fondée en 2014 composée d'un douzaine de personnes. Elle développe des **Dispositifs Médicaux Implantables Actifs (DMIA)** plus précisément des neurostimulateurs pour adresser différentes pathologies liées au système nerveux central comme périphérique. Par exemple, les pathologies de Parkinson, les douleurs fantômes dues à une amputation, les fonctions urinaires et motrices pour les tétraplégiques sont des pathologies ou des dysfonctionnements moteurs qui par neurostimulation ponctuelle ou chronique peuvent être soulagés, voire soignés. Pour adresser ces pathologies, on distingue alors les neurostimulateurs implantables, portables (sur le patient en externe pour une durée déterminée) et sur table (pendant une opération).

En accord avec le type de pathologie ciblé et le **DMIA** à développer, le processus de développement d'AXONIC doit être conforme avec différentes normes. En effet, un **DMIA** étant implanté dans un corps tous les aspects sécurité électriques doivent être pris en compte pour par exemple résister à une défibrillation. A l'inverse un dispositif portable peut mettre à disposition des interactions avec le patient ou son entourage, les aspects utilisabilité sont alors à prendre en compte. Les normes en jeu sont donc des plus générales avec la ISO 13485 qui décrit les exigences haut-niveau pour le développement de produit dans le médical, ISO 14971 qui impose une gestion des risques jusqu'aux plus précises comme la norme IEC 62366 qui décrit un canevas général qui a pour préoccupation l'utilisabilité du dispositif. De telles normes doivent être identifiées, revendiquées et appliquées par AXONIC. Par exemple, pour la IEC 62366, il est nécessaire de prendre en compte des activités comme l'évaluation sommative, la formation des utilisateurs et la documentation utilisateur. Ce cadre général pour le médical est ensuite affiné pour, par exemple, spécifier les exigences pour l'utilisation du **DMIA** au domicile du patient à travers la norme auxiliaire IEC 62336-1. Celle-ci introduit notamment les aspects erreur humaine dans l'utilisation, e.g., sécurité physique pour prévenir l'utilisation par un enfant du dispositif. AXONIC doit aussi être conforme à des normes "*métier*" relatives au développement *hardware* et *software*, e.g., ISO 60601 qui décrit les exigences essentielles pour la sécurité electro-magnétiques et IEC 62304 qui donne un canevas de développement pour le logiciel dans le médical.

Dans cette validation, nous allons nous focaliser sur la conformité d'un produit en phase de développement chez AXONIC. Nous cherchons ainsi à atteindre pas à pas la conformité, pour le logiciel, dans le cadre d'un produit destiné à des études cliniques. A cette fin, un ensemble de documents doit être produit pour assurer la confiance dans le produit et le processus de développement dont il est issu. Des guides pour le développement logiciel en médical permettent de raffiner le IEC 62304 en un cadre plus proche des pratiques de l'entreprise comme le guide TIR-45 **AAMI [2012]**. En effet, dans cette validation, nous appliquons nos travaux dans un contexte Agile. Ce guide spécifie un cadre de conformité pour ces pratiques. Ainsi, le développement d'AXONIC découpé en itérations, appelées *sprints* en Agilité, est conforme à IEC 62304. Plus généralement, ces guides compilent des pratiques éprouvées et donnent des outils pour aider les entreprises à produire la preuve appropriée de conformité, i.e. les justifications.

Par ailleurs, les normes de plus haut-niveau (e.g., ISO 13485, ISO 14971) sont raffinées dans les normes "*métier*". Par exemple, ISO 14971 est raffinée pour le logiciel dans IEC 62304 mais également dans IEC 62366 pour l'utilisabilité. Ceci mène à des activités de justifications complexes et entrelacées puisque les exigences de justification relatives à la gestion des risques se retrouvent diffuses dans plusieurs normes. De plus, AXONIC étant centrée sur des projets de Recherche et Développement Avancé, beaucoup de projets prospectifs sont menés. Il leur faut être capable de capitaliser sur l'aspect normatif entre les projets pour gagner du temps tout en gardant confiance dans ce qui est réalisé.

Adapter et reproduire des patrons de justifications de projets en projets sont donc de réels enjeux pour eux.

Nous montrons à travers trois études comment nos travaux ont répondu pour AXONIC aux objectifs fixés au Chapitre 4. Pour ce faire, nous étudions 1) comment nous avons structuré les exigences de justifications de la norme IEC 62034 et les pratiques internes d'AXONIC avec nos travaux, 2) comment nous avons partiellement automatisé la production des justifications associées, 3) comment nous avons aidé à valider les justifications produites et intègre notre approche dans le [SMQ](#) d'AXONIC.

8.2 Environnement technique

8.2.1 Redmine, un [SMQ](#) centré sur l'Agilité

Redmine est un logiciel de gestion de projet souvent associé aux pratiques Agile. Il permet de centraliser par projet la *roadmap*, les tâches qui en découlent sous forme de tickets et la documentation sous forme de page web (Wiki). Ainsi il peut être utilisé comme outil de traçabilité de la définition des besoins jusqu'à la traçabilité des produits livrés. AXONIC a ainsi choisi Redmine comme son outil principal de [SMQ](#). On y retrouve ainsi le *Master file* de chaque projet et les justifications associées sous forme de page Wiki. Redmine intègre pour ses pages un versionning automatique permettant de collaborer sur l'écriture de page Wiki tout en supportant l'historique des modifications. Enfin, ces pages peuvent être verrouillées par le chef de projet pour bloquer les modifications. Ainsi, AXONIC a défini un template et cycle de vie pour les documents de justifications. Dans un premier temps, un auteur rédige la justification, qui est ensuite revue et validée par des approbateurs puis, après validation par chacun, est verrouillée par l'expert qualité affecté au projet. Ainsi nous retrouvons une cartouche dans chaque page montrant l'état d'une justification (rédigé, approuvé, verrouillé). Un audit d'un projet consiste ainsi à partir du point d'entrée de Redmine avec le *Master file* et d'aller petit à petit vers des éléments plus techniques comme un document des bonnes pratiques pour le code du logiciel ou un document de revue de bugs.

8.2.2 Jenkins, une plateforme d'intégration continue pour le système tout en entier

AXONIC a défini un *plan de développement logiciel* qui contextualise la IEC 62304 et TIR-45 et met en face les pratiques et outils permettant d'y être conforme. Ce plan de développement est principalement orienté autour des outils de gestion du code (Gitlab), d'intégration continue (Jenkins) et de stockage d'artefacts logiciels (Nexus). Jenkins est la plateforme d'intégration continue avec la plus grosse communauté active. Il permet de configurer des *jobs* décrivant les tâches à exécuter, quand et où les exécuter. Ces tâches peuvent tout aussi bien être la compilation et les tests d'un code source comme des backups de machine. L'objectif est de disposer d'une plateforme de planification qui permet d'exécuter des tâches sur des machines dites *esclaves*. Un *job* quand il est exécuté mène à instance de celui-ci, appelé un *build*, lequel est définie par l'état d'exécution des tâches : *stable*, *instable* (e.g., tests en échecs, analyse statique de code pointe des problèmes) ou *en erreur* (e.g., impossible de récupérer un artefact logiciel, compilation en erreur). AXONIC a centré sa stratégie autour de cet outil, le code de pilotage d'un neurostimulateur comme le code embarqué sont compilés et testés des *jobs* dans Jenkins.

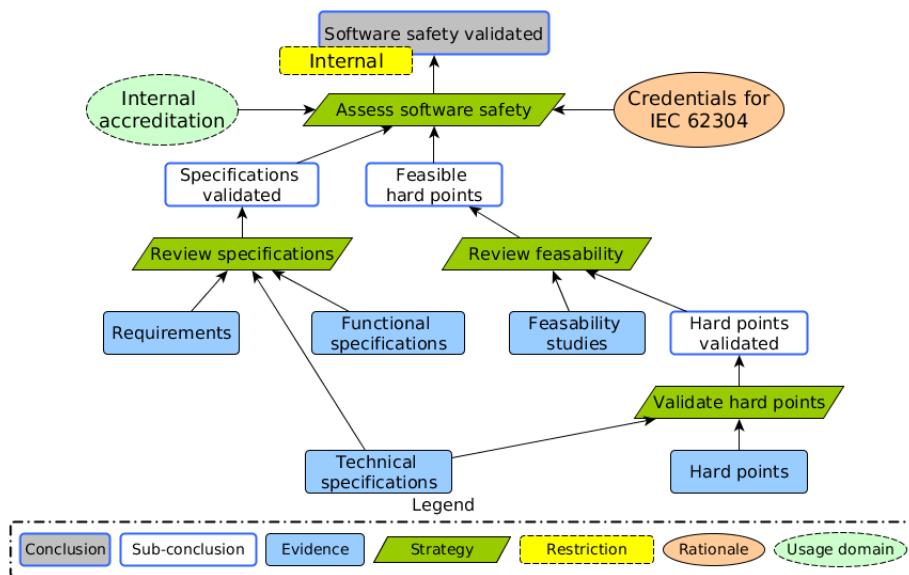


FIGURE 8.1 – Ce DPJ illustre le résultat en termes d'exigences de justifications du *kick-off* du nouveau projet. Les besoins doivent mener à des spécifications fonctionnelles et techniques qui doivent être revues et validées. Pour raffiner les spécifications techniques, des points difficiles doivent être identifiés et il est nécessaire de conduire des études de faisabilité pour montrer qu'il est possible de les dépasser.

8.3 Elicitation des exigences de justifications pour IEC 62304

8.3.1 Conduite de l'étude

Dans ce cas d'études, les parties prenantes impliquées dans les justifications sont : un chercheur/praticien¹, l'équipe d'experts qualité (2 personnes), et les responsables techniques (3 personnes). L'équipe d'experts qualité est responsable de la veille réglementaire, la conception des procédures internes, la préparation des audits, etc. Les responsables techniques assurent que les justifications prévues dans les procédures sont produites correctement et défendent ces justifications lors des audits. Pour conduire cette étude, le chercheur a été intégré dans les activités au quotidien de l'entreprise, des réunions (*e.g.*, suivi de projet, analyse des risques, réunion technique, audit) jusqu'à la production du produit lui-même (*e.g.*, développement, utilisation des outils).

Après un an d'implication dans AXONIC, le chercheur a conçu un premier DPJ pour un nouveau projet, basé sur les pratiques de l'entreprise (voir Figure 8.1). Puis, durant l'année suivante, en parallèle de l'avancement du projet, l'équipe d'expert qualité et les responsables techniques ont itéré pour produire les justifications, améliorer le DPJ et prendre en compte de nouvelles normes applicables. Pour construire le bon DPJ à chaque étape de développement du projet, l'équipe a itéré sur la base du processus suivant :

- 1 Avant le début d'une nouvelle étape de développement, le chercheur conçoit un DPJ en accord avec les exigences des experts qualité et des responsables techniques. Ce nouveau DPJ est *dérivé* du DPJ courant, pour, par exemple, prendre en compte une nouvelle norme applicable. La Figure 8.2 est une *dérivation* de la Figure 8.1. A cette étape, pour *assess software safety*, il est obligatoire d'avoir l'architecture du code validée. Comme montrée dans le diagramme, la validation passe par une revue qui correspond à l'ajout de la stratégie : *Review architecture*. Cette revue d'architecture requiert d'utiliser une justification précédente à propos des études de faisabilité. De plus, pour être conforme à la norme ISO 14971, de nouveaux pas de justifications doivent être ajoutés pour assurer le management des risques. IEC 62304 et

1. moi

ISO 14971 sont entrelacées en termes de management des risques logiciels. Grâce au *JPD*, nous avons identifié les justifications précédentes qui peuvent être modifiées pour répondre à ces nouvelles exigences, mais aussi identifier les justifications manquantes.

- 2 Les responsables techniques (i) identifient les éléments de justification qui doivent être produits et (ii) développent et/ou configurent les outils pour les produire (*e.g.*, extraire des documents de la chaîne de production, ajouter un plug-in aux outils existants) **DUFFAU et collab. [2017c]**. Basé sur le **DPJ**, un **DJ** est créé. Si un **DJ** de l'étape précédente existe, on procède à l'*alignement* dirigé par le nouveau **DPJ**. S'il n'existe pas de **DJ**, on procède à la *réalisation* à partir du nouveau **DPJ** c'est-à-dire à la production de toutes les justifications durant cette étape de développement.
- 3 Durant l'étape de développement, les responsables techniques définissent éventuellement de nouvelles activités, telles que la validation par un expert, la revue en comité ou des tests externes. L'équipe de développement produit quant à elle les justifications nécessaires dans le **DJ**. Évidemment, durant le développement de nouvelles justifications ou manières de justifier peuvent apparaître. Par conséquent, elles sont ajoutées au **DJ** courant. Notons qu'à ce moment, nous avons perdu la conformité entre les exigences de justifications capturées par le **DPJ** et la production réelle des justifications capturées par le **DJ**.
- 4 A chaque fin d'étape de développement, les experts qualité analysent les différences entre le **DPJ** et le **DJ**. Cette *comparaison* aide à identifier (i) les justifications manquantes qui auraient dû être produites et (ii) des justifications produites dans le **DJ** qui n'étaient pas attendues et qui doivent être potentiellement introduites dans le **DPJ**. Après analyse de la raison de ces différences, un *alignement* est effectué, dirigé par le **DPJ** si on veut réfuter les retours du terrain ou dirigé par le **DJ** si on veut accepter les évolutions amenées par le terrain. Par exemple, durant le projet, AXONIC a introduit une nouvelle norme, IEC 62366, dans ses pratiques. Durant la production des justifications, nous avons noté que les documents produits pour la ISO 14971 et pour la IEC 62366 étaient en fait plus facile à produire dans une justification conjointe que de manière séparée. De ce fait, la propagation de ce retour du terrain dans le **DPJ** a été effectuée c'est-à-dire un alignement dirigé par le **DJ**.

Finalement, cette conduite de l'étude se rapproche fortement du cycle de vie général des **DJs** que nous avions décrit en Section 6.3.1.

Suite à cette structuration des normes et pratiques internes sous forme de **DPJ**. Nous avons souhaité étudier la compréhension d'une telle représentation par rapport à l'existant. Étudier la compréhension d'un **DPJ** par rapport aux normes, guides et pratiques internes textuelles nécessite de faire appel à des personnes extérieures à ses aspects pour mettre en évidence leur utilité. Néanmoins, il fallait avoir une bonne connaissance en développement logiciel pour être à même de comprendre le métier que cherche à capturer ses normes. Pour éviter les biais dans cette étude, nous avons donné à un nouvel arrivant en développement logiciel chez AXONIC, l'ensemble documentaire norme IEC 623014, guide TIR-45 et la procédure de développement logicielle d'AXONIC et en parallèle le **DPJ** représentant IEC 62304, guide TIR-45 et un **DPJ** raffiné de celui-ci représentant la procédure de développement logicielle d'AXONIC. L'objectif était ici de qualitativement mesurer si les **DPJ** aident à rentrer dans l'environnement qualité de l'entreprise. Nous détaillons les résultats de cette étude préliminaire sur l'utilisabilité des **DJs** dans la section suivante.

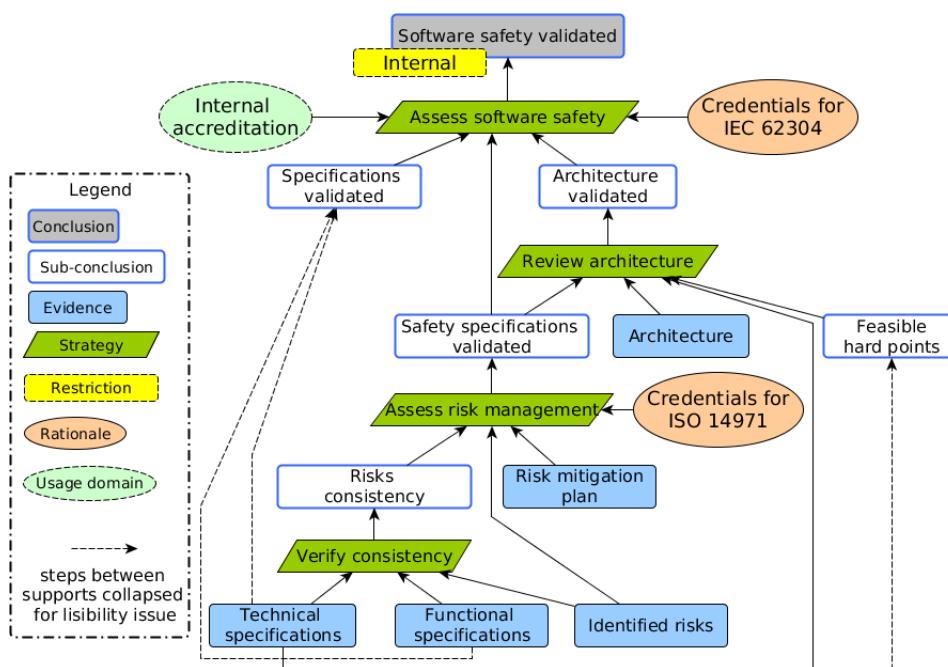


FIGURE 8.2 – Quelques étapes du projet plus loin, durant la phase de conception que représente ce DPJ, nous gardons les supports définis dans les étapes précédentes, mais nous dérivons ce DPJ pour répondre aux exigences de IEC 62304 à cette nouvelle étape.

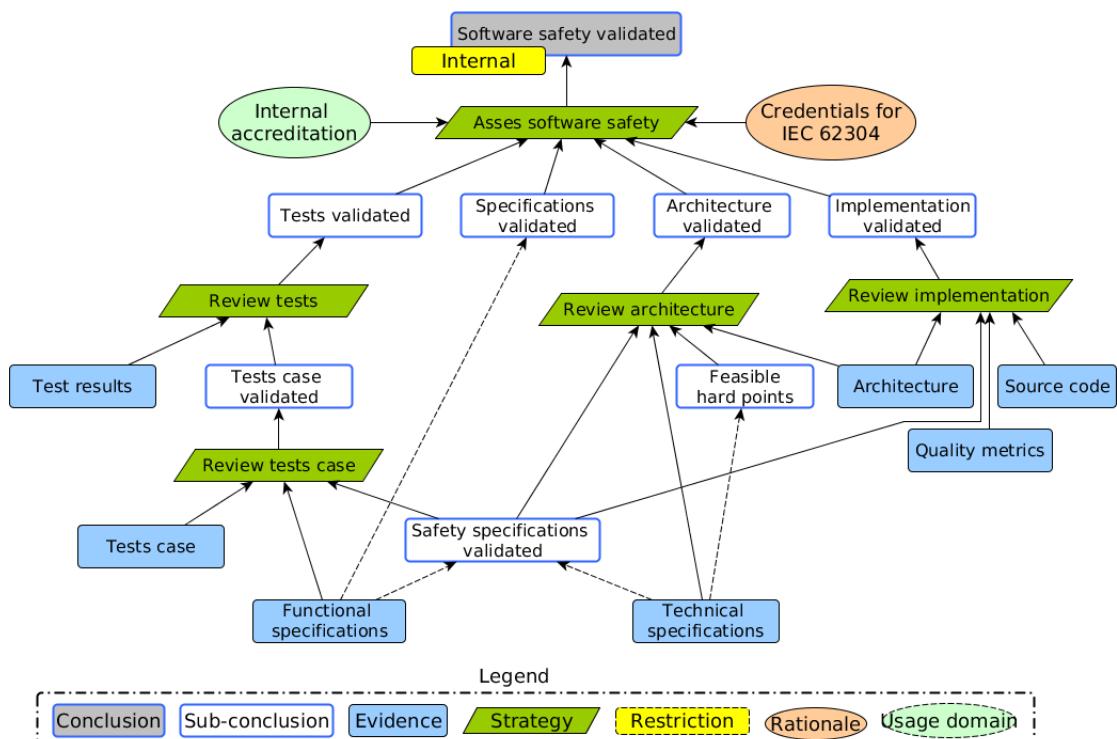


FIGURE 8.3 – A la phase de développement apparaît les aspects code et tests qui nécessitent la réalisation d'exigences de justification spécifiques

8.3.2 Analyse de l'étude

Durant cette étude, nous avons conçu 5 DPJs pour les cinq étapes majeures du projet. Le dernier DPJ contenant l'ensemble des exigences de justifications est donné en Annexe A.5. Nous avons utilisé ses DPJs pour, à chaque étape, guider la production d'un DJ. Nous avons versionné chaque SJ de chaque sprint de développement de la même façon que l'on peut faire sur du code. En effet, il est utile de pouvoir attester de la conformité aux exigences de justifications à chaque itération du projet et il fallait donc être capable de retrouver des états passés. Dans le futur, ces DPJs seront réutilisés comme la première entrée pour de nouveaux projets.

Cette étude visait à mettre en application nos travaux dans un contexte industriel. Ceci nous a permis en parallèle d'étudier les objectifs que nous nous sommes fixés dans \mathcal{O}_1 : Capturer des exigences de justifications. En effet, les diagrammes proposés respectent la notation graphique des travaux sur laquelle nous nous sommes basés. En ce sens, nous remplissons C_{1.1} (Préserver la sémantique et les représentations connues adaptées à la justification).

En observant, comment le nouvel arrivant a abordé l'appropriation de cet ensemble documents et DPJs, nous en avons établi quelques enseignements². Un DPJ est plus simple pour avoir une vision globale de ce qui est attendu par les normes. Le raffinement de DJ permet de rendre plus tangible la matérialisation des normes dans les pratiques internes. In fine, une fois cette première appropriation faite grâce aux DPJs, il est plus facile de parcourir l'ensemble documentaire et guider ses recherches par l'utilisation des DPJs. Ces enseignements ont été validés par l'équipe d'experts qualité et l'équipe logicielle d'AXONIC déjà en place en les questionnant, puis en constatant qu'ils les utilisent toujours. Le questionnaire de cette étude est donné en Annexe A.6. Par conséquent, nous avons bien rempli C_{1.2} (Structurer des normes et pratiques internes) dans le contexte d'AXONIC. En effet, cette étude préliminaire tend à attester que la connaissance diffuse et des normes, guides et pratiques internes et également les liens implicites entre les exigences de justifications sont bien capturés et plus compréhensibles dans des DPJs. Dans le cadre d'AXONIC, il n'a pas été possible de pousser davantage cette étude pour des questions d'effectifs de l'équipe développement logiciel (1 seul recrutement sur l'année en cours). Il serait intéressant d'étudier à plus large échelle l'utilisabilité et la compréhensibilité des DPJs avec une évaluation utilisateur dédiée.

D'autre part, dans le cadre de l'étude de cette compréhension nous avons montré que notre approche permettait de capturer la norme IEC 62304 dans un DPJ qui était ensuite raffiné pour rendre plus proche des pratiques internes d'AXONIC. Ce raffinement, n'aide pas seulement la compréhension. En effet, le raffinement conserve, par la définition formelle que l'on a donnée, la conformité ainsi nous remplissons bien C_{1.3} (Supporter le raffinement des exigences de justifications).

Lors de cette étude, le processus que nous avons suivi pour les établir de manière confiante a mis en jeu les opérations que nous avons définies permettant de supporter l'évolution des exigences de justifications. En ce sens, nous remplissons C_{1.4} (Supporter l'évolution des exigences de justifications).

Ainsi l'ensemble des défis que nous nous étions fixés dans \mathcal{O}_1 : Capturer des exigences de justifications ont été validés par cette étude.

8.3.3 Limites

Après cette première étude, AXONIC a souhaité mettre en place notre approche à plus large échelle pour la gestion du dossier de conception de leur projet tout entier. Ce dos-

2. Nous les relatons ici, même si une évolution formelle n'a pas été suivie et reste l'objet de perspectives

sier est composé de l'ensemble des justifications sur un produit ayant atteint la fin de son développement et capture donc l'ensemble des normes à revendiquer pour un tel produit. Pour AXONIC, cela rassemble 6 normes. Ceci nous a amenés à concevoir un DPJ beaucoup plus volumineux contenant une centaine de pas de justifications. Ce DPJ est présenté en Annexe A.7³. Ici, l'utilisabilité de ce diagramme reste très discutable et nous amène à penser qu'il est nécessaire de repenser l'outil de visualisation de la *Justification Factory* pour permettre d'étendre et réduire des pas de justifications dans un DJ et ainsi aider à minimiser le focus sur certaines justifications. Le problème n'est, en effet, pas tant la taille du DJ que la séparation d'un DJ en plusieurs. Il faudrait sans doute plusieurs SJ pour le dossier de conception. Chaque SJ représenterait les justifications par une norme et il faudrait pouvoir établir des liens entre eux pour pouvoir représenter les justifications conjointes entre les normes. Ainsi nous pourrions capturer dans le formalisme les relations possibles entre SJ pour permettre une meilleure séparation des préoccupations de justification et donc une utilisabilité accrue. Néanmoins, cet effort de capture des exigences de justifications du dossier de conception pour AXONIC a permis de mettre en place notre approche d'automatisation au sein du SMQ que nous détaillerons en Section 8.5.

8.4 Production automatisée des justifications pour la IEC 62304

8.4.1 Développement des outils pour l'intégration de *Justification Factory* et Jenkins

Après identification des artefacts de justifications qui peuvent être produits automatiquement pour l'entreprise, il s'avère que tous ceux qui sont produits automatiquement sont récupérables par Jenkins. Ainsi, pour permettre la production automatique des justifications à partir de DPJs pour AXONIC, il a fallu dans un premier temps développer des plug-ins pour les outils Jenkins capables d'interagir avec le *Justification Bus* de la *Justification Factory*.

Pour Jenkins, la plateforme étant fortement extensible, il a suffi de respecter l'API d'extension de la plateforme pour créer un plug-in qui permet de créer des assertions à partir d'artefacts de justification produits durant un *build*. Cela consiste finalement à décrire dans le *job* qu'après les tâches principales à exécuter, il faut assembler certains artefacts dans des assertions dédiées (e.g., *UnitTestResult*, *IntegrationTestResult*, *CodeAnalysisResult*, *CodeCoverage*). À la fin de l'exécution d'un *job*, ces assertions sont construites avec les artefacts et sont envoyées au *Justification Bus*. Ainsi l'ensemble des justifications liées

3. Pour sa visualisation, nous l'avons décomposé en sous-arbre

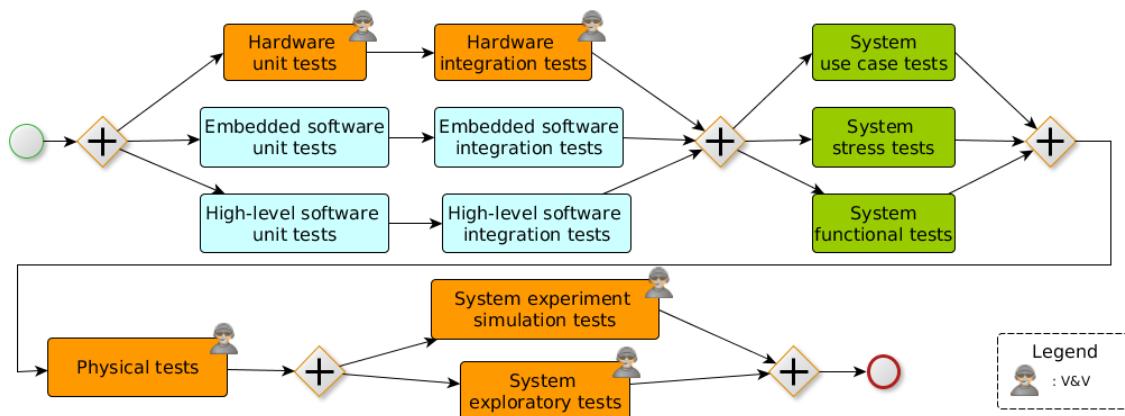


FIGURE 8.4 – Extrait du processus de V&V d'AXONIC

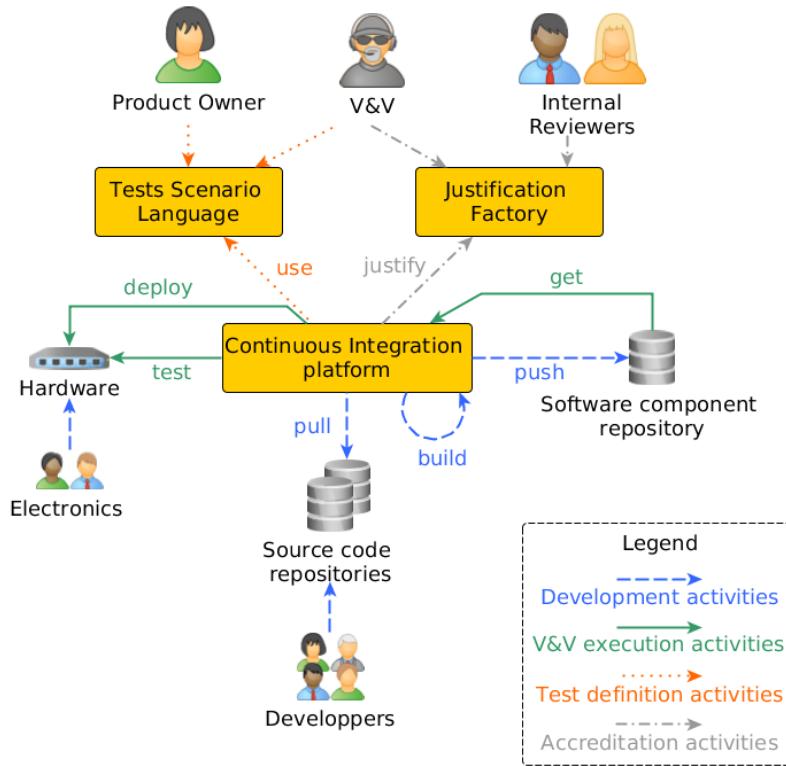


FIGURE 8.5 – Écosystème de la chaîne d'intégration continue

au code (*e.g.*, tests stables, couverture de test satisfaisante, qualité du code convenable) sont produites automatiques par remontée d'informations de la part de Jenkins dans la *Justification Factory*.

De plus, en mettant en place cette chaîne d'outils pour AXONIC, nous avons mis en évidence que leur chaîne d'intégration pouvait être encore plus automatisée et permettrait ainsi de couvrir encore plus de justifications. Comme vu précédent, nous pouvons automatiser les activités V&V liées au logiciel, mais dans le cadre de logiciel embarqué, il faut être capable d'aller jusqu'à la plateforme d'exécution, c'est à dire le *hardware*. En effet, la V&V consiste également à faire des tests sur le système de bout en bout c'est-à-dire avec le logiciel de pilotage, le logiciel embarqué et le matériel (*e.g.*, tests de performance, tests de procédure de sécurité). Ces tests étaient jusque là faits à la main de l'exécution jusqu'au rapport de tests associé. Pour aller, vers un environnement qui aide autant dans les activités de V&V que dans les justifications, il s'agit pour nous d'améliorer la chaîne d'intégration continue pour permettre l'automatisation de ses tests et de la production des justifications.

Nous montrons en Figure 8.4, les activités de la chaîne de V&V pour AXONIC. Par cette figure nous visualisons le passage progressif vers l'automatisation au maximum possible de cette chaîne. Les activités bleues représentent les tâches déjà automatisées par l'entreprise avec des pratiques et outils courants (*e.g.*, Junit, CMock, Unity, Mockito). Les tâches en orange qui dépassent le cadre de l'intégration continue (*e.g.*, nécessite de faire appel à un prestataire, tests cliniques avec des êtres vivants, activités exclusivement sur le matériel) Les activités en vert correspondent aux activités de tests sur le système mis bout à bout. Pour les exécuter, il est nécessaire de disposer d'un dispositif physique, de tests système écrits par la V&V et de la pile logicielle issue des activités bleues précédentes.

L'approche générale permettant de mettre en œuvre ces tests système est donnée en Figure 8.5. En bas de la figure, les *activités de développement* suivent des pratiques courantes et utilisent les outils classiques de Jenkins. Ces activités correspondent finalement

aux activités bleues de la Figure 8.4. En milieu de figure, les *activités de V&V* correspondent à la récupération des artefacts logiciels produits auparavant, et au déploiement de ceux-ci. Pour le code embarqué, nous avons développé notre propre outil permettant de déployer le code sur le dispositif physique. À ce stade l'environnement du système entier est opérationnel et contrôlé (*e.g.*, version des codes, vérification du lancement). Cet environnement est mis en place pour chaque job dans Jenkins qui vise à lancer des tests liés aux activités en vert sur la Figure 8.4. Pour l'exécution des tests à proprement parler, nous avons développé un *Domain Specific Language (DSL)* nommé *Test Scenario Language* qui permet de définir des scénarios de test d'acceptation d'une fonctionnalité, des tests de charge, etc. Ce langage a été développé pour être adapté à l'utilisation de l'équipe V&V . De ce fait, nous avons masqué les détails techniques pour nous concentrer sur le langage métier de la neurostimulation. Ainsi en début d'une nouvelle itération de développement, durant la phase d'analyse des spécifications, les tests systèmes sont écrits par la V&V (*activités de définition des tests* dans la Figure 8.5). Suivant l'approche *Test Driven Development BECK [2003]*, ces tests sont exécutés durant l'ensemble de l'itération par Jenkins où le but est de passer d'un statut *instable* à *stable* durant l'itération. À la fin de l'itération, pour la livraison, nous nous assurons de la stabilité de ces tests. Comme pour les autres activités automatiques de V&V, nous reportons alors à la *Justification Factory* des assertions qui sont associées aux tests pour construire les justifications globales de la V&V (*activités de justification* dans la Figure 8.5).

Grâce à cet écosystème pour l'intégration continue intégrant mieux la V&V sur un système bout-en-bout, nous avons observé une réduction des coûts des activités de V&V tout en améliorant la qualité du produit final (*e.g.*, ressources nécessaires pour la V&V , plus de scénarios tests exécutés dans une itération) **DUFFAU et collab. [2017c]**.

De par le développement de ces extensions des outils existants, nous atteignons dans le cadre d'AXONIC le défi **C_{2.2}** (Proposer un environnement intégré pour l'utilisation des justifications). Nous étudions dans la suite de cette section les aspects production automatique des justifications.

8.4.2 Conduite de l'étude

Pour mesurer l'impact de ce passage de la construction et vérification manuellement des justifications à un environnement où ce qui peut être automatisé l'est, nous avons mesuré le facteur suivant avant et après.

$T_{m\text{production}}$ Nombre de justifications à produire manuellement par rapport à l'ensemble des justifications à produire, , par sprint de développement. Ce métrique vise à valider **C_{2.1}** (Supporter le passage à l'échelle dans la production des justifications). En effet, minimiser le nombre de justifications à produire à la main permettrait de diminuer le temps de production des justifications par notre solution et ainsi de supporter le passage à l'échelle.

Avant, pour IEC 62304, cela consistait notamment à récupérer manuellement les artefacts de justifications dans Jenkins (*e.g.*, page web d'un build, lien vers l'analyse statique du code, lien vers le binaire produit, page de JavaDoc), les vérifier (*e.g.*, build stable, pas de nouveaux bugs critiques découverts par l'analyse statique du code, binaire manquant) et les assembler dans des documents de justifications (*e.g.*, page Wiki attestant de qualité du code de la dernière livraison). Avec notre approche, l'ensemble de ce processus est automatisé. Depuis Jenkins, les artefacts de justifications sont mis dans des assertions et, par la *Justification Factory*, sont assemblés pas à pas pour atteindre la conclusion de haut-niveau souhaitée dans un **DJ**. Notons le rôle fort des stratégies qui permettent de mettre en évidence ce qui est complètement automatisable par un outil de ce qui nécessite l'intervention d'un humain.

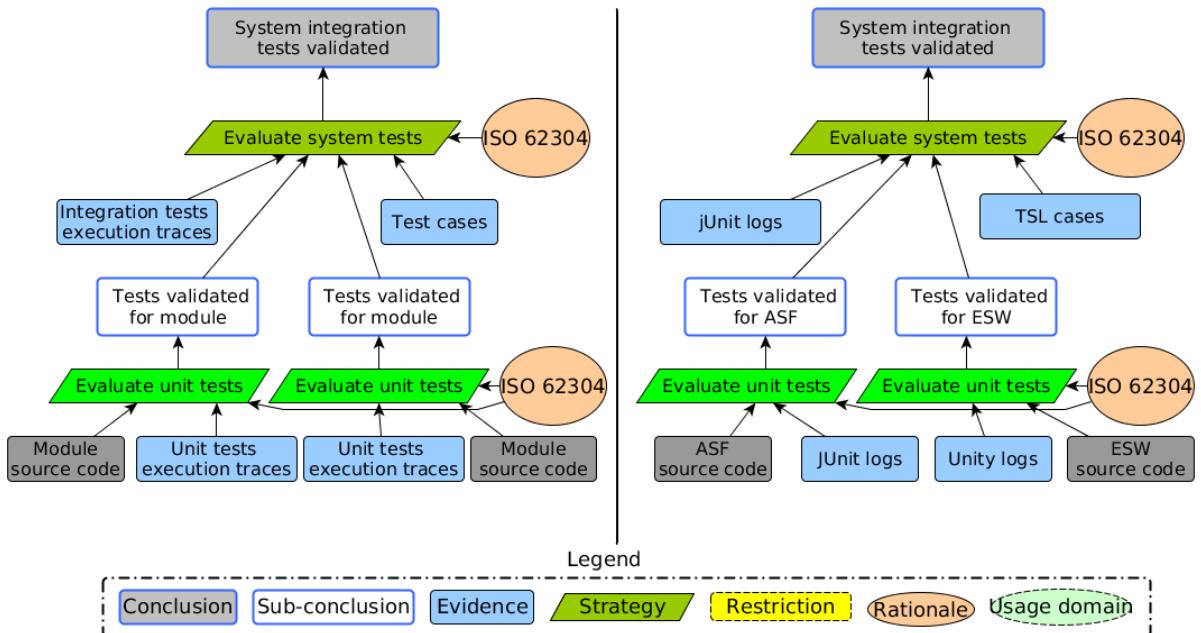


FIGURE 8.6 – DPJs pour valider l'exécution des tests d'intégration logicielle

Nous donnons un extrait de DPJs matérialisant ces assertions et justifications en Figure 8.6. À gauche, le DPJ capturant les exigences de justification de IEC 62304 et à droite, le DPJ contextualisant ces exigences par les pratiques internes d'AXONIC⁴. Ce dernier DPJ donne un aperçu des techniques utilisées par l'entreprise pour la gestion des tests unitaires et d'intégration logicielle. On retrouve l'utilisation de la pile technologique Java avec jUnit pour le test unitaire du logiciel de pilotage nommé ASF et la pile technologique C avec Unity pour les tests unitaires du logiciel embarqué nommé ESW. Pour les tests d'intégration du système bout-en-bout, nous retrouvons le DSL Test Scenario Language (TSL) pour la description des cas de test. Ces cas de tests sont ensuite exécutés par un moteur utilisant Java et donc les logs sont produits par jUnit. Notons que dans ce dernier DPJ, l'ensemble des assertions peuvent être construites automatiques par Jenkins. Chaque pas de justifications correspond finalement à un *job* dans Jenkins et chaque *build* mène à la production d'un pas.

Nous avons conduit cette étude durant les 30 sprints de développement d'un nouveau projet pour AXONIC avec l'équipe de développement logiciel et l'équipe V&V d'AXONIC. Ce lapse de temps intègre le passage vers l'utilisation de TSL et la chaîne de justifications qui va avec. Ainsi le passage vers l'automatisation des justifications présenté ici s'est fait pas à pas.

8.4.3 Analyse de l'étude

Les données brutes de cette étude sont données en Annexe A.8.

À la fin de l'étude, parmi les 82 artefacts de justification pour attester de la validation d'un livrable de la fin de sprint de développement, 59 sont issus de Jenkins et les justifications les utilisant ont été automatisées par notre solution. $T_{m,production}$ a ainsi passé de 100% à 28%. Les éléments restants consiste à compléter les justifications produites et rédiger le rapport final. Pour la personne en charge de rédiger le document final pour la validation, visualiser les justifications déjà produites dans un outil spécifique à la justifi-

4. Nous ne donnons pas ici de DJ, car trop technique. Le DPJ pour les pratiques internes reste néanmoins très proche du DJ. Les seules différences sont de l'ordre des versions des logiciels, de l'infrastructure informatique d'AXONIC et des projets de l'entreprise

cation (*justification GUI*) sur lequel baser son rapport est donc apparu comme un levier pour éviter les erreurs lors de la collecte de ces justifications. Ce rapport reste à rédiger à la main, car 23 artefacts de justifications restent des activités humaines (e.g., tests manuels, vérification de tests faux-positifs). Nous parlons ici seulement de ce qui se passe en fin de sprint. Au cours d'un sprint qui dure 2 semaines, les *jobs* dans Jenkins sont lancés entre 1 et 10 fois par jour chacun. Ce qui correspond à 500 tests sur l'intégration du système par jour. L'ensemble des justifications au jour le jour est produit et structuré dans des **DJ**. Ceci permet au cours d'un sprint de gagner en confiance tout du long et également de contrôler la convergence vers les justifications attendues. Finalement, la fin de sprint consiste à prendre le dernier lié à la livraison. Nous voyons bien à travers ces métriques, les besoins de passage à l'échelle et l'incrémentalité des justifications auxquels nous avons tâché de répondre par nos travaux en s'interfaçant avec la plateforme d'intégration continue.

De cette étude découle que nous avons ainsi bien répondu aux défis que nous nous étions fixés dans le cadre de l'objectif **O₂** : Guider la production des justifications.

8.5 Des Diagrammes de Justification (DJs) vers l'intégration dans le SMQ d'AXONIC

8.5.1 Développement des outils pour l'intégration de *Justification Factory* et Redmine

Pour Redmine, nous ne cherchons pas à atteindre ici l'automatisation de la production des justifications. Les justifications non automatisables sont produites par AXONIC sous forme de document dans Redmine. Cette documentation est constituée des minutes des réunions, des matrices d'analyse de risques, des comptes rendus de revue d'architecture, etc. La production de ces justifications nécessite alors l'analyse d'un expert ou d'un comité pour attester de certaines propriétés que l'on retrouve dans ces documents qui servent de justifications. L'intérêt pour nous ici est donc d'assembler l'ensemble des justifications qui sont diffuses dans Redmine dans un **DJ**. Ainsi nous aidons l'évaluation de la conformité globale du dossier de conception d'AXONIC au regard des exigences capturées préalablement dans le **DPJ** de la section 8.3.

Il est possible d'écrire des plug-ins également, mais ils sont trop limités dans leur exécution. En effet, cela reste de simples plug-ins pour l'interface graphique permettant d'enrichir de l'enrichir avec de nouvelles préoccupations métiers. Ici, ce qu'il est nécessaire de faire pour nous, c'est de moniturer les pages Wiki pour en remontée des changements d'état (rédigé, approuvé, verrouillé). L'état qui nous intéresse pour insérer une justification dans un **DJ** est donc quand elle est verrouillée. De ce fait, nous avons développé un service dédié (*Redmine monitoring service for Justification Factory*) qui notifie les apprivateurs quand un document est rédigé et signé par son auteur, et l'expert qualité quand les apprivateurs ont tous validé le document. Plus concrètement, ce service regarde si un pas de justification peut être produit à partir de l'assertion contenant le document, si ce n'est pas possible, il analyse les assertions manquantes pour pouvoir le faire et notifier. Ainsi le flot pour arriver à l'état verrouillé est accéléré et ne nécessite plus de monitoring humain. Une fois cet état atteint le service construit des assertions dédiées (e.g., document dans Redmine, approbation d'un document) permettant ainsi d'envoyer ces éléments au *justification bus* pour la construction du **DJ**. Cette automatisation est transparente pour les humains qui n'ont pas besoin de comprendre comment les justifications sont assemblées et se concentre seulement sur la production des éléments critiques. Il est ensuite possible d'évaluer ce **DJ** en utilisant le *conformance service* pour vérifier la complétude de ce **DJ**. In fine, le *Master file* est généré au format d'une page Wiki pour éviter cette tâche source d'erreur et automatisable grâce à notre approche. Étant donné la criticité du *Master file*, AXONIC n'a pas souhaité que l'écriture de cette page se

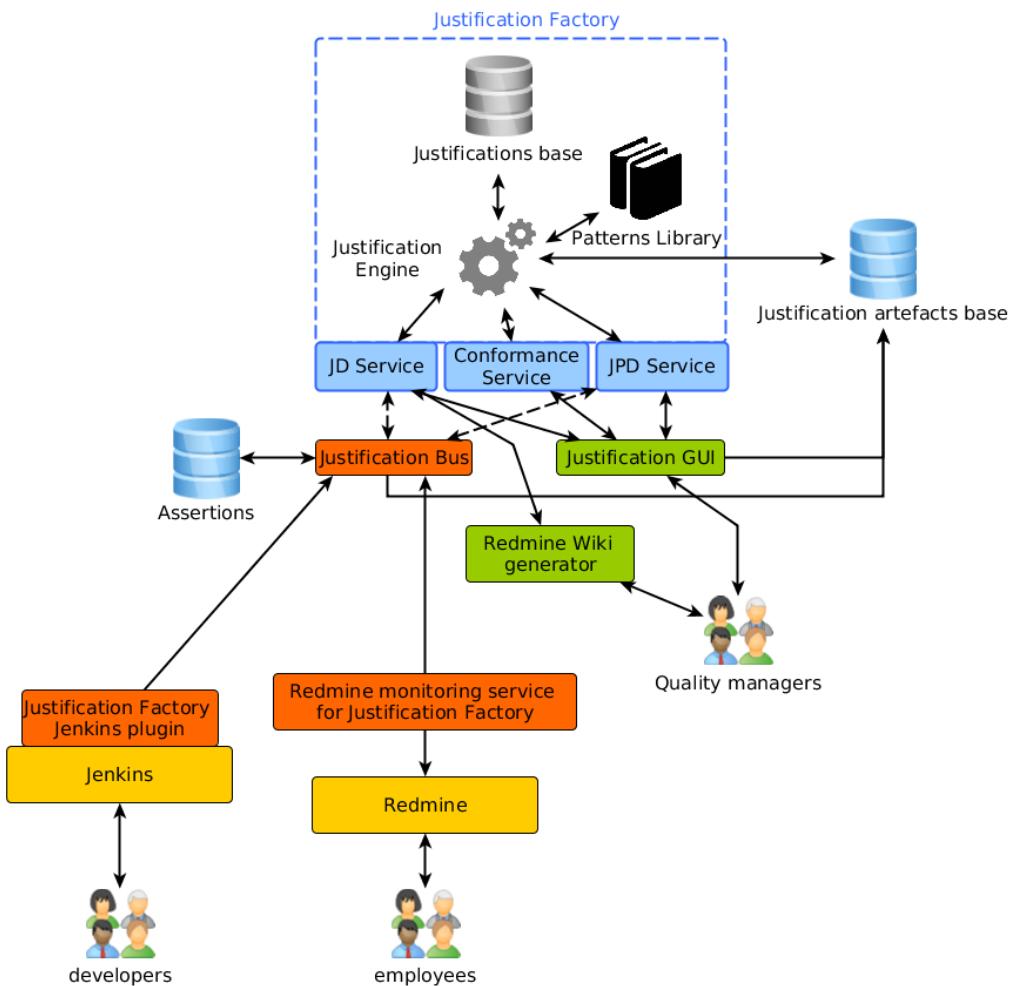


FIGURE 8.7 – Extensions de *Justification Factory* pour les outils d'AXONIC

fasse de façon automatique dans Redmine. Nous avons développé le générateur pour le *Master file* (*Redmine Wiki generator*) à côté qui permet à un expert qualité de vérifier cette génération avant de l'insérer dans Redmine. A l'introduction de ce générateur, ce travail de vérification a permis de mettre en évidence des manquements dans le *DPJ* (e.g., exigences oubliées dans le *DPJ*) du dossier de conception et après quelques itérations, l'outil a été suffisamment éprouvé pour avoir confiance.

Nous donnons l'architecture globale pour AXONIC autour de la *Justification Factory* en Figure 8.7. Notons qu'il ne s'agit ici que d'extensions sous forme de plug-ins ou services. Ces extensions ont été d'abord conçues pour être au même niveau d'abstraction que la *Justification Factory*, c'est-à-dire qu'elles peuvent être réutilisées dans d'autres contextes que cette étude. Elles restent seulement liées aux outils avec lesquels elles font le pont (e.g., *Justification Factory* ↔ Redmine), mais pas avec le métier et les spécificités d'AXONIC.

8.5.2 Conduite de l'étude

Au préalable de cette étude, il nous a fallu concevoir un *DPJ* capturant les exigences de justifications pour ce dossier de conception.

Pour mesurer l'impact de ces extensions autour du suivi et validation des justifications dans Redmine, nous avons mesuré les facteurs suivants avant et après.

T_{suivi} Temps, par sprint de développement, de l'expert qualité pour assurer le suivi des justifications (temps pour analyser, notifier pour approbation et verrouiller des do-

cuments). Par nos travaux, les activités de suivi correspondent à la vérification qu'un pas de justification est valide. Ce temps vise à valider **C_{3.1}** (Supporter une production des justifications en continu). En effet, minimiser ce temps permet de supporter une production en continu des justifications, car le suivi devient gérable à l'échelle du sprint.

T_{validation} Temps, par sprint de développement, de l'expert qualité pour mettre à jour le *Master file* quand de nouvelles justifications ont été verrouillées. Par nos travaux, les activités de validation correspondent à vérifier et valider qu'un **SJ** est complet et le projeter sur un *Master file*. Ce temps vise à valider **C_{3.2}** (Faciliter la validation de justifications). En effet, minimiser ce temps avec notre solution permettrait de montrer qu'il est plus simple de valider des justifications.

Te_{suivi} Taux d'erreur dans la forme des justifications (approbation valide). Cette métrique vise à mesurer si les optimisations de temps de production et V&V des justifications se font au détriment de la qualité de celles-ci.

Te_{validation} Taux d'erreur sur l'agrégat des justifications (justification manquante). Cette métrique vise à mesurer si les optimisations de temps de production et V&V des justifications se font au détriment de la qualité de celles-ci.

8.5.3 Analyse de l'étude

Les données brutes de cette étude sont données en Annexe [A.9](#).

Nous avons conduit cette étude sur les 4 derniers sprints du nouveau projet d'AXONIC. Nous avons d'abord analysé le résultat des 25 sprints précédents. Malgré un temps conséquent de l'expert qualité pour suivre ces justifications, sur ces 25 sprints de nombreux documents présentaient des erreurs de validation (*e.g.*, document non approuvé, mais version suivante déjà écrite par l'auteur, problème dans le format de la date ou signature électronique d'un document).

Avec notre solution, le temps de suivi se résume à verrouiller les documents quand notre service notifie l'expert qualité que les conditions sont réunies pour le faire puisque ce service s'occupe du suivi d'un document de son écriture jusqu'au verrouillage. De plus, de l'aveu même de l'expert qualité, ce travail rébarbatif et source d'erreur l'amenait à sauter des suivis sur certains sprints. Au final, ce suivi était fait tous les 2 ou 3 sprints et donc n'était pas régulier comme prévu. Avec notre solution, chaque semaine, les personnes concernées par des documents en erreur sont ainsi notifiés des documents à problème. Ainsi nous passons de **T_{suivi}** de 1 heure à quelques minutes. En effet, au cours de notre étude nous sommes passés de 10 minutes avec notre solution à 1 minute au bout des 4 sprints. Cela vient tout simplement de la prise de confiance en l'outil par l'expert qualité dans notre approche. Au début, il vérifiait les résultats qui lui étaient transmis et au fur à mesure que l'outil a fait ses preuves, il a pu se détacher de ces vérifications. De plus, la régularité du suivi a été regagnée.

Suite à ce contrôle systématique du bon format des approbations en vérifiant si un pas de justification peut être construit dans l'état du document actuel, nous avons ainsi permis de pointer les erreurs pour les corriger et ainsi anticipé les défauts de conformité. De ce fait, **Te_{suivi}** est passé de 33,9 % à 16,3 %. Ce taux d'erreur résiduel provient des transitions entre les personnes concernées par ces activités et des documents déjà verrouillés et qui ne peuvent pas être *ré-ouverts* pour être mis à jour. En effet, cela reviendrait à éditer et donc revalider toutes ses justifications passées, ce qui n'est pas acceptable pour l'expert qualité. Hormis cet historique sur lequel nous ne pouvons rien faire, dès qu'une erreur est introduite, une notification est envoyée aux personnes concernées. Au delta que les personnes notifiées tiennent compte des actions correctives à effectuer sur les documents, en fin d'étude, **Te_{suivi}** ainsi affiné est estimable à 0.

Pour la validation des justifications dans leur ensemble à travers le **DJ** qu'a construit le service associé à Redmine, nous avons matérialisé la notion de **SJ** complet dans *Justification GUI* ce qui permet d'assurer la conformité de ce qui a été produit lors de la revue par l'expert qualité. Une fois cette revue effectuée et les éventuelles corrections effectuées, il s'agit pour cet expert d'obtenir une version revendicable de ces justifications sous forme de *Master file*. $T_{validation}$ qui consistait à vérifier et mettre à jour le *Master file* à la main devient outiller de bout en bout. Ainsi, nous avons diminué $T_{validation}$ de 30 minutes à quelques minutes. En effet, nous avons également observé, que dès l'introduction de notre solution pour valider les justifications, $T_{validation}$ est passé à 5 minutes et qu'au bout des 4 sprints de l'étude, $T_{validation}$ est passé à 2 minutes. Cela vient de la prise de confiance dans l'outil. Tout comme T_{suivi} , notons que l'expert qualité sautait certains sprints par manque de temps avant l'introduction de notre approche.

Si nous regardons $T_{validation}$, avant la mise en place de notre solution, il arrivait que certains documents n'étaient pas verrouillés en temps voulu et donc non introduit dans le *Master file* ou des documents non verrouillés présents dans le *Master file*, avec notre solution de notification et de vérification de la conformité, de tels manquements dans le *Master file* ne peuvent pas être générés. Ainsi, avant notre étude, le *Master file* contenait 22 documents erronés, soit 13,8 % des documents, que nous avons tous éliminés par notre solution.

Pour conclure, nous avons donc considérablement réduit les temps T_{suivi} et $T_{validation}$ tout en réduisant T_{suivi} et $T_{validation}$. Grâce à l'automatisation du suivi et validation des justifications et la prise de confiance dans notre approche par AXONIC, l'entreprise envisage désormais de verrouiller automatiquement par notre outil les documents dès lors que l'approbation des vérificateurs a été effectuée, mais également de générer automatiquement le *Master file* par notre approche, sans intervention humaine. Cette étude valide donc notre approche sur l'objectif **O₃ : Vérifier et analyser la conformité entre exigences et production des justifications** avec les défis **C_{3.1}** (Supporter une production des justifications en continu) et **C_{3.2}** (Faciliter la validation de justifications).

Pour conclure ce chapitre, nous avons démontré l'utilité de nos contributions scientifiques sur le cas d'étude d'AXONIC au cours d'un réel projet développé par l'entreprise. L'approche a été déployée au cœur du processus de justification de l'entreprise sur ce projet. Il reste néanmoins à mesurer l'apport sur d'autres projets pour mesurer notamment la réutilisabilité des **DPJs** de projet en projet. D'autre part, nous avons présenté ici principalement des métriques quantitatives sur l'apport de notre solution. Des analyses basées sur des questionnaires pour valider ou non l'utilisabilité d'une visualisation sous forme de diagramme, du confort ou non de faire confiance à un outils pour la gestion d'un sujet aussi critique que les justifications, etc. En prenant du recul sur cette étude, nous notons que la façon dont AXONIC souhaite utiliser ces travaux de recherche est très proche d'une approche processus, mais dirigé par les justifications. En effet, utiliser des **DJs** sur les différentes activités du flot d'un document de son écriture jusqu'à son insertion dans le **SMQ** permet, en toute confiance, de contrôler le processus de justification. Cette utilisation nous amène à penser qu'une étude approfondie de la relation des **DJs** et des modélisations de processus métiers par exemple avec *Process Management Notation (BPMN)* (OMG) [2011] est une perspective à explorer.

Chapitre 9

Conclusion et perspectives

« *Every new beginning comes from some other beginning's end* »

Seneca

Sommaire

9.1 Autres cas d'application	94
9.1.1 Évaluation de la charge de travail sur une chaîne de production d'avion	94
9.1.2 Justification automatique sur plateforme d'expérimentation	95
9.2 Leçons apprises et limites	99
9.2.1 Les DPJs élicitent les exigences de justification	99
9.2.2 Les DJs aident à gérer les justifications	100
9.3 Perspectives	101
9.3.1 Recherche	101
9.3.2 Industrie	104
9.4 Synthèse	105
9.5 Liste des publications	107

Dans ce chapitre, nous allons d'abord étendre la portée de nos travaux sur deux cas d'application bien différents de ceux présentés jusqu'à présent. Nous dressons ensuite des conclusions sur nos travaux sous forme de leçons apprises et de limites de notre contribution. Nous ouvrons ensuite sur des perspectives pour la suite de ces travaux. Pour conclure, nous établissons une synthèse de notre contribution au regard des objectifs que nous nous étions fixé.

9.1 Autres cas d'application

Jusqu'ici durant nos travaux, nous avons considéré un exemple fil rouge basé sur un référentiel qualité pour les organismes de formation et sur un cas d'application centré sur les normes dans le biomédical. Nous avons également initié des travaux pour étudier jusqu'où peut être utilisée notre approche. Pour ce faire, nous avons collaboré sur i) une étude sur l'utilisation des **DJs** dans un domaine critique et ii) sur une étude sans normes en jeu : l'évaluation de la charge de travail sur une chaîne de production d'avion et dans un domaine non critique qui cherche à expliciter des connaissances : les plateformes d'expérimentation. Nous montrons pourquoi et comment nous avons introduit les **DJs** sur ces deux cas.

9.1.1 Évaluation de la charge de travail sur une chaîne de production d'avion

Pour assurer que les coûts de production ne sont pas pris en compte trop tardivement lors de la conception d'un avion, il est crucial d'effectuer une analyse de ces coûts au plus tôt de la conception. En effet, le calcul de l'évaluation de la charge de travail est très critique. Beaucoup de décisions peuvent être prises à partir de cette information comme des choix de conception, mais aussi les besoins en main-d'œuvre, l'évaluation des coûts, etc. Pour effectuer une évaluation fiable de la charge de travail, il est nécessaire d'établir une forte interaction entre les ingénieurs responsables de la conception et les équipes de production **POLACSEK et collab. [2017]**.

Historiquement, l'évaluation de la charge de travail est basée sur une conception très avancée et requiert plusieurs calculs longs et complexes. Pour avoir une estimation plus rapidement, mais moins détaillée et donc moins fiable, un nouveau processus a été défini. Pour rester proche des pratiques de l'avionneur, ce processus doit rester intégré dans un cycle en V. Cela commence au tout début de la conception et termine avec une conception avancée en réutilisant entre chaque étape autant que possible les résultats obtenus précédemment. La confiance est ainsi gagnée à chaque étape du processus. Au début, avec beaucoup d'incertitude, mais une évaluation rapide avec peu d'informations. À la fin, avec une évaluation lourde, mais très fiable.

Nous avons eu l'opportunité de suivre cette application utilisant une approche basée sur les **DPJ** conduite par Polascek. Le regard sur ces travaux, nous a permis de prendre du recul sur l'application présentée en Chapitre 8. Dans cette étude, à la différence d'AXONIC, il n'y avait aucune exigence de justifications provenant de l'organisme de certification. Ici, l'objectif est de gagner en confiance sur cette évaluation de la charge de travail par la justification, seulement à des fins internes. Ainsi, à contrario d'AXONIC, où l'on est parti des exigences de justifications pour guider la production des justifications, Polascek et l'avionneur ont effectué l'étude dans l'autre sens. Ils sont partis de justifications construites sur des projets passés qu'ils ont représentés sous forme de **DJs** pour ensuite remonter au niveau d'un **DPJ** commun pour en élucider les exigences de justifications implicites. Grâce à cette étude, l'entreprise dispose à la fin d'une suite de **DPJ** capturant les exigences de justifications permettant pour chaque étape de la conception d'un nouvel avion d'effectuer cette évaluation. Finalement le processus pour la mise en place de ces

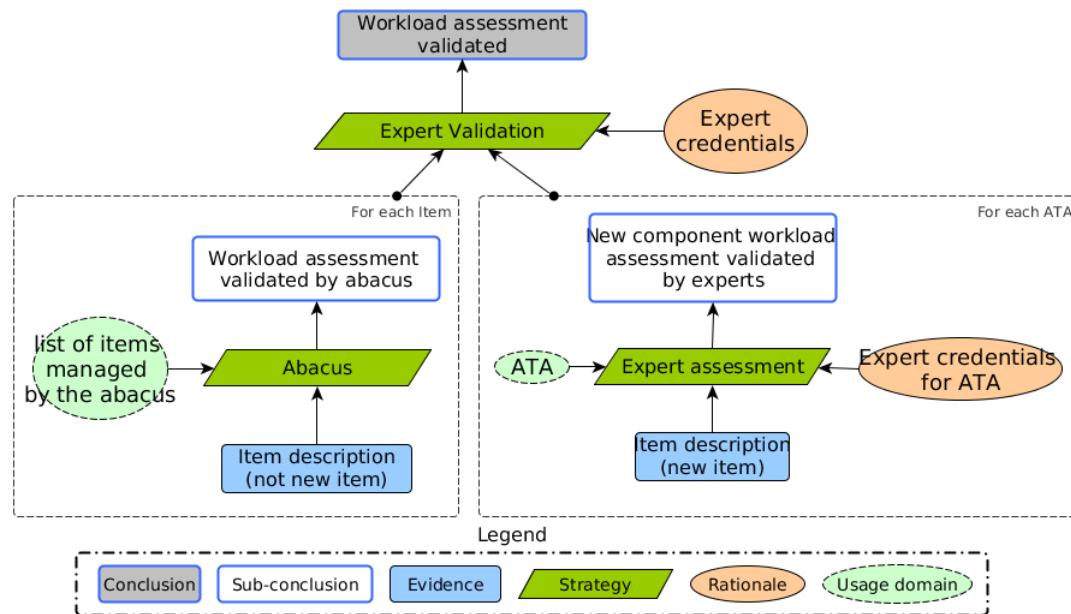


FIGURE 9.1 – Diagramme-Patron de Justification (DPJ) à l'étape de conception préliminaire

DPJs est un exemple concret du flux d'opérations que nous montrions en Figure 6.8 du Chapitre 6.

En Figure 9.1 est présenté un des DPJs conçu durant cette étude par Polascek. Dans ce diagramme, il est intéressant de noter que cette étude a fait apparaître le besoin d'utiliser des patrons de justification intégrant un mécanisme de répétition *for each*. En effet, dans un DPJ de très haut niveau, certains éléments de justification sont non dénombrables. Ici, les ATAs¹ ne sont pas dénombrables car nous cherchons à capturer les exigences de justifications pour l'ensemble des projets de l'entreprise. Ainsi à cette étape, il n'est pas possible de connaître les spécificités du projet est donc les ATA concernés. Cette cardinalité est précisée dans les DJ qui raffine ce DPJ. Il est nécessaire d'observer l'usage d'un tel patron de justification sur d'autres cas pour identifier si ce mécanisme est redondant et doit être remonté dans notre formalisme. Si tel est le cas, cela nécessite d'introduire des patrons de justifications particulier et la relation \mathcal{R} associée à ces patrons.

Ce cas d'étude montre, que même, à des fins internes sans norme en jeu, construire un DJ pour établir les justifications pour chaque grande étape d'un projet permet de gagner en confiance et en compréhension.

9.1.2 Justification automatique sur plateforme d'expérimentation

Motivations : construction d'un portfolio d'algorithmes de *Machine Learning*

D'un point de vue utilisateur bétien, la construction de workflows de Machine Learning (ML) est complexe par l'immense variabilité des algorithmes, des preprocessing et des stratégies de paramétrisation FERNÁNDEZ-DELGADO et collab. [2014], mais également par la difficulté de faire les bons choix; le meilleur algorithme n'est pas le même pour chaque problème WOLPERT [1996], il dépend entre autres de la taille et de la nature des données et de ce que l'utilisateur veut apprendre de ces données. Pour répondre à cette problématique le projet ROCKFlows² a été initié. En fonction du jeu de données de l'utilisateur et de ses objectifs, l'approche vise à générer le workflow de ML le plus approprié

1. les chapitres ATA sont définis par la *Air Transport Association of America*. C'est une norme commune pour tous les avions commerciaux. Un chapitre ATA représente un domaine de l'avion comme *Air Conditioning & Pressurization* (ATA 21), *Electrical Power* (ATA 24) ou *Pneumatics* (ATA 36).

2. rockflows.i3s.unice.fr

CAMILLIERI et collab. [2016]. La plateforme ROCKFlows repose sur la construction d'un portfolio **LEYTON-BROWN et collab. [2003]** de workflows de ML, associé à une Ligne de Produits Logiciels (LPL). Le portfolio se présente comme un recueil d'algorithmes et de jeux de données à partir desquels des expérimentations sont menées. Sur la base des résultats obtenus, la plateforme construit des modèles prédictifs permettant de sélectionner en fonction d'un problème donné les workflows possibles et de prédire leurs performances. Les informations se trouvant dans le portfolio sont remontées par abstraction au niveau de la ligne de produits dans des termes utilisateurs. Cette remontée est nécessaire pour réduire les choix de l'utilisateur et supporter le processus de configuration, tout en favorisant une évolution rapide de la ligne. L'étape de configuration d'un produit consiste alors à fournir le jeu de données ou les méta-informations qui lui sont associées, à sélectionner les principaux objectifs puis en fonction des prédictions proposées à choisir un workflow. Celui-ci est alors généré automatiquement. Il ne s'agit donc pas d'une approche d'assemblage de workflows, mais bien d'une génération à partir d'objectifs de haut niveau.

Au niveau de l'interface utilisateur n'apparaissent évidemment que les informations pertinentes en fonction du problème posé. Les prédictions pour chacun des workflows possibles, *e.g.*, la précision et le temps d'apprentissage résultent d'un processus complexe d'apprentissage. Dans ce contexte, comment justifier à l'utilisateur qui le souhaite la pertinence des prédictions? La justification repose sur les jeux de données sur lesquels le système a appris, des protocoles d'expérimentations utilisés, de la manière dont le modèle de prédiction a été construit, etc.

Construction incrémentielle des justifications Afin de répondre à cette question, dans le futur, il serait intéressant de travailler à produire automatiquement et de manière incrémentielle un **DJ** qui trace les activités permettant de construire la connaissance utilisée au niveau de la LPL. Cela suppose non seulement de prendre en compte les "ajouts" comme de nouveaux résultats d'expérimentation ou l'ajout d'un algorithme, mais également de filtrer les éléments non pertinents, par exemple, des jeux de données qui n'apportent pas de plus-value aux résultats et donc à la justification.

Cohérence par construction des justifications. Chaque activité (*e.g.*, expérimentation, analyse, abstraction) peut faire l'objet d'une justification. La manière dont celle-ci est construite doit être cohérente par rapport à l'activité elle-même, mais également par rapport à l'ensemble de la justification. Par exemple, l'ajout d'une nouvelle expérimentation doit vérifier que toutes les informations relatives à son contexte d'exécution sont bien données tandis que la construction des modèles de sélection ne doit se faire qu'à partir de résultats ayant suivi les mêmes protocoles d'expérimentations (*e.g.*, une évaluation par *cross-validation sur 10 folds*). Ces contraintes dans la construction de la justification sont elles-mêmes évolutives en fonction de nos apprentissages et des biais relevés dans la littérature, par exemple, des comparaisons entre des algorithmes sur des données préparées différemment ou en n'utilisant pas exactement les mêmes méthodes d'évaluation **FERNÁNDEZ-DELGADO et collab. [2014]**.

Utilisabilité de la justification Construire la justification et assurer sa cohérence ne suffit pas. Cette justification doit pouvoir être présentée aux utilisateurs de la LPL ou aux experts pour validation et discussion. Ainsi, les experts métier doivent être en mesure de la lire facilement, en particulier en naviguant entre les arguments et les artefacts expérimentaux.

Comme nous le voyons à travers ce cas d'utilisation, il ne s'agit pas ici de justifier pour

attester de la conformité avec des normes, mais bien de justifier dans le but d'expliquer des connaissances et permettre à une personne extérieure d'avoir confiance dans ces connaissances acquises par expérimentation. Il existe d'autres exemples où ce même objectif de la justification existe. Par exemple, dans le cadre d'AXONIC, avant la mise sur le marché d'un produit, des essais cliniques sont conduits permettant de tester le produit sur d'abord des animaux puis des patients où il est recherché pas à pas la thérapie pour une pathologie permettant d'assurer efficacité et sûreté. L'atteinte de ces deux propriétés doivent être justifiées auprès de comités éthiques, promoteur de l'étude, etc dans le but de montrer que ces expérimentations vont ou ont été menées convenables. Notons ici, que nous ne cherchons pas à concevoir un DPJ pour capturer les exigences de justifications puisque il n'y a pas ou peu d'exigences dû à l'aspect empirique du domaine des expérimentations. Cela rejoint le point que nous avons relevé dans la section 7.5 sur le besoin d'avoir plusieurs façons de guider un SJ. Pour conduire l'étude guidée par ces deux cas applicatifs, nous avons donc généralisé cette problématique pour les systèmes expérimentaux.

Systèmes expérimentaux et justifications

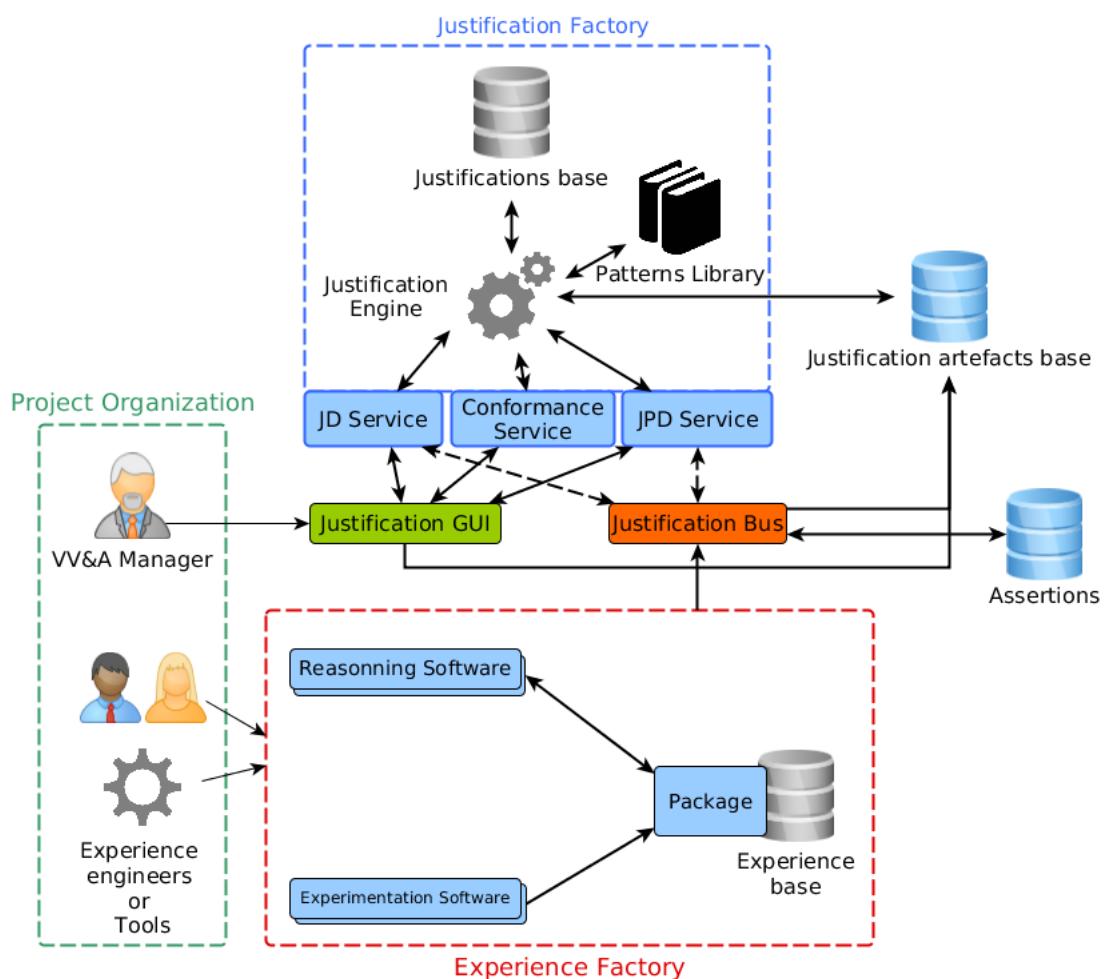
Un *système expérimental* a été défini par Rheinberger **RHEINBERGER [1997]** comme *a basic unit of experimental activity combining local, technical, instrumental, institutional, social, and epistemic aspects*.

Dans **BASILI et collab. [1994]**, Basili décrit comment l'organisation d'un projet se construit autour d'un système expérimental pour conduire des expérimentations empiriques. Une *Experience Factory (EF)* est alors une infrastructure conçue pour supporter le management des expériences sur des logiciels par la répétition de ces expériences et réutilisabilité des résultats. Cette définition initiale peut être étendue à tout système d'informations conçu pour gérer des expériences même autres que logicielles (e.g., logiciel pour des études cliniques). Une EF supporte une collection de *logiciel d'expérimentation*, l'analyse de ces expérimentations par des *logiciels de raisonnement* et assemble ces expérimentations et leur analyse dans une *base d'expériences* comme montré en bas de la Figure 9.2. Alors que EF s'intéresse à capitaliser sur les connaissances acquises sur des expérimentations pour les réutiliser sur d'autres, nous nous focalisons nous, dans ce contexte, sur la justification durant l'analyse de ces expérimentations dans les *logiciels de raisonnement*.

L'approche EF seule n'est pas suffisante pour les besoins de justification car ne supportent pas les justifications permettant de faire le lien entre des artefacts et ainsi les structurer pour construire une justification globale. Les EFs supportent seulement une collection d'artefacts (e.g., documents, statistiques, données d'expérience, conclusions) utilisée pour les expérimentations et le résultat de l'analyse des expérimentations dont ils sont issus [**WOHLIN et collab., 2012**, Fig.6.5].

Il existe néanmoins quelques travaux rapprochant EF et justification. Dans **LARRU-CEA et collab. [2016]**, les auteurs décrivent la chaîne d'outils issue de l'EF pour suivre le processus décrit dans la norme ISO/IEC29110³ pour les aspects de certification. Cette approche est principalement basée sur la vérification d'exigences de justifications de la norme et non, comme nous, sur le raffinement et la conformité à prendre en compte avec les pratiques internes et les moyens de production. Par exemple, si nous prenons la propriété *Attester que le propriétaire du code est formellement identifié*, nous ne cherchons pas seulement si elle est vérifiée, mais également comment cette propriété a été obtenue (e.g., analyse des fichiers pour trouver des métadonnées sur les auteurs).

3. Norme pour l'ingénierie de systèmes et l'ingénierie du logiciel - les processus de cycle de vie pour les très petits organismes

FIGURE 9.2 – Architecture entre *Experience Factory* et *Justification Factory*

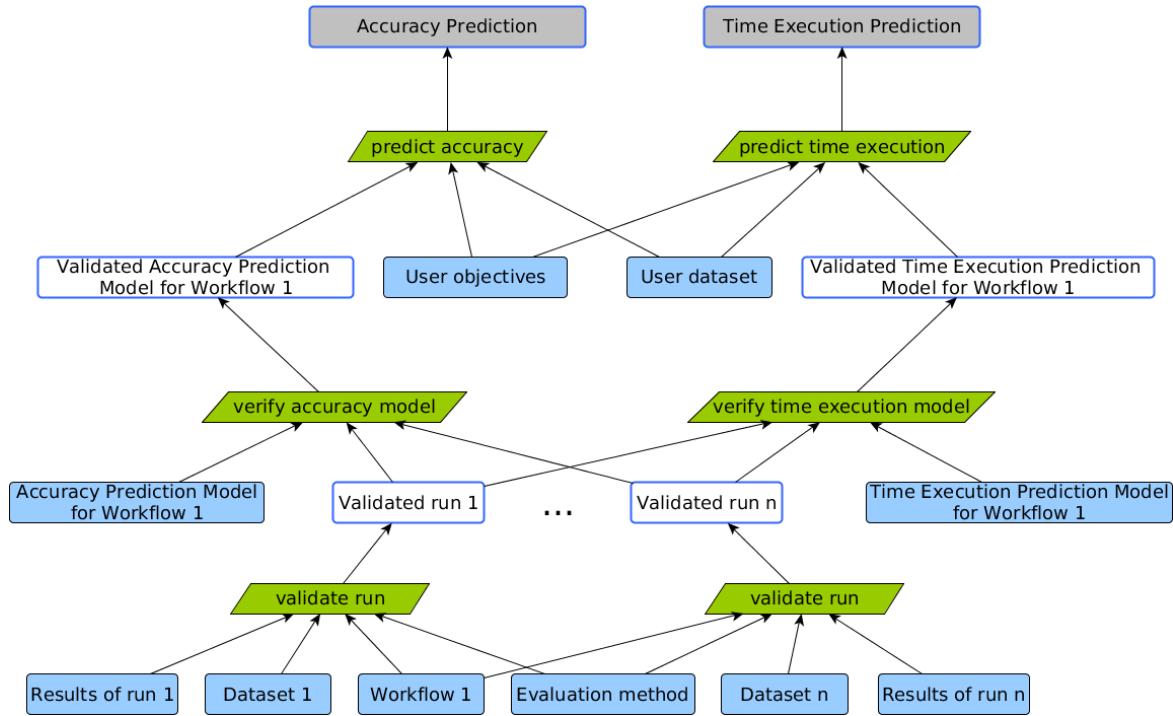


FIGURE 9.3 – Exemple de DJ sur ROCKFlows

Ainsi en appliquant nos travaux sur les systèmes expérimentaux en récupérant l’approche EF, nous proposons l’architecture présentée en Figure 9.2. Comme dans l’étude pour AXONIC, où la production des justifications se faisait sans que les outils n’aient connaissance de la justification globale ni même comment en construire une partie, ici aussi nous proposons de brancher au *justification bus* les *logiciels de raisonnement*. Il est de la responsabilité de ces logiciels de savoir produire des assertions. Un exemple de DJ envisagé sur RockFlows est donné en Figure 9.3. Dans cette figure, nous retrouvons finalement les artefacts produits par les *logiciels d’expérimentation* en évidences du DJ et par l’utilisation des *logiciels d’analyse* ces évidences sont pas à pas assemblées dans des justifications permettant d’attester de la prédition, de la précision et du temps d’exécution d’un algorithme sur une base.

Dans ce domaine, très empirique, il reste un écueil que nous n’avons pas levé. En effet, comme dit en fin de Chapitre 7, plusieurs types de SJ doivent être possibles. Ici, cet exemple illustre bien ce besoin. En effet, dans les systèmes expérimentaux, un résultat attendu est obtenu par les expérimentations, mais la conduite pour y arriver peut varier en fonction des aléas d’étude (e.g., cas à rejeter dû aux conditions expérimentales, biais découverts tardivement). Dès lors, les exigences de justifications sont flexibles, ce que ne permet pas un DPJ. Il n’existe donc pas formellement de DPJ, mais seulement la notion d’objectif à atteindre pour la justification. Le besoin de disposer d’un SJ centré sur une base de patrons non structurée est donc renforcé.

9.2 Leçons apprises et limites

9.2.1 Les DPJs élicitent les exigences de justification

Les DPJs synthétisent l’information requise pour justifier d’une propriété. En fait, ils sont une extension de la formalisation des exigences et des méthodes.

DPJ fait le lien entre Verification & Validation (V&V) et les exigences de justifications. La relation entre les exigences de justification et les DPJs repose actuellement sur l'expertise des responsables qualité pour l'aspect normatif puis est déporté sur les responsables *métiers* pour la transcription dans de réelles pratiques. Dans l'application avionique comme pour AXONIC, établir cette relation était la partie essentielle du travail. Par nos travaux, nous avons formalisé le raffinement des exigences de justifications pour permettre de supporter l'élicitation des exigences de justifications. Ceci s'est matérialisé chez nous par la caractérisation de la conformité entre justifications. À travers nos études, nous avons montré l'utilité de cette formalisation.

Concevoir un DPJ requiert un processus itératif. La construction des DPJs est fortement dépendante de plusieurs facteurs. De notre expérience, le niveau de maturité des équipes autour du produit à développer, l'objectif du projet (*e.g.*, prospectif, production) et le cycle de développement choisi sont les points clés. Dans nos expérimentations, les équipes ont dû raffiner les DPJs de l'étape du projet. Pour des raisons de coût et de management, toutes les justifications ne peuvent pas être produites au début d'un projet. Certaines nécessitent d'être ajoutées ou raffinées durant le cycle de vie du projet, et parfois, des activités de justification doivent être ajoutées durant le développement quand la production d'un artefact de justification n'avait pas été anticipée. Donc, la construction des DPJs est un processus itératif dans lequel non seulement les experts qualité doivent être intégrés, mais également toutes les parties prenantes du projet. À travers les cas d'études présentés, nous avons montré que les opérations que nous avons définies sur les justifications ont permis de guider les experts dans la conception des DPJs.

Nos études se focalisent sur des projets long terme dans des domaines différents. À travers les études et exemples présentés dans nos travaux, nous avons étudié différents contextes où des normes étaient impliquées, mais aussi d'autres sans contexte normatif. Cependant, ils correspondent tous à des projets long terme. Par conséquent, les équipes ne peuvent pas encore réutiliser et appliquer des DPJs sur d'autres projets. Cependant, la nature itérative des études ou des projets ciblés et l'utilisation des opérations pour faire évoluer les DPJs montrent l'intérêt de réutiliser les DPJs entre les grandes phases d'un même projet.

9.2.2 Les DJs aident à gérer les justifications

Comme nous l'avons vu, les éléments des DJs sont connectés à des artefacts de justification, par exemple, un rapport ou un procès-verbal de réunion. Ainsi, un DJ peut être considéré comme un moyen d'organiser les documents de justification.

Les DJs fournissent une justification globale basée sur des artefacts de justification. Dans l'application pour AXONIC, les DJs se réfèrent aux artefacts de justification stockés dans les outils de l'entreprise (*e.g.*, Redmine, Jenkins). Notre approche, par la structuration de ces artefacts sous forme de DJ, nous a permis de générer automatiquement le document textuel central de justification : le *Master File*. Dans ce projet, lors d'un audit annuel les DJs ont été présentés à l'auditeur qui y a trouvé un moyen plus visuel et structuré de représenter le contenu du SMQ. Cependant, pour les activités hautement régulées, il n'apparaît pas possible actuellement de travailler seulement avec les DJs. Néanmoins, les retours de l'équipe de développement et de l'auditeur tendent à montrer le bénéfice significatif de référencement automatique par les DJs et la possibilité de projeter ce format de justifications sur des canevas plus standard. Bien que nous ayons initié une étude sur l'utilisabilité et la compréhension des DJs, une évaluation utilisateur systématique doit être menée dans ce cadre pour évaluer plus précisément la représentation graphique des DJs.

Les DJ s aident la production des justifications en toute confiance. Parce que les DPJs peuvent être vus comme des guides structurés, une liste détaillée des artefacts de justifications nécessaires, ils soulagent les équipes de développement de l'oubli de justifications nécessaires pour la conformité du produit. En effet, la validation, par les responsables qualité, de la bonne capture de toutes les exigences de justifications par le DPJ, permet l'équipe de se focaliser sur le développement et non les normes tout en ayant le fil conducteur à suivre pour la production des justifications associées. De plus, les DJ s permettent de supporter la production des justifications. En effet, dans le cadre d'AXONIC, nous avons automatisé la production de nombreuses justifications basées sur les artefacts produits par ou dans les outils de l'entreprise avec l'intégration continue et le SMQ par Redmine. Cette automatisation est un des points centraux dans l'atteinte de la conformité en toute confiance. Une fois les outils configurés, ils produisent toujours des artefacts cohérents contrairement à un humain qui peut faire des erreurs.

Le contexte expérimental peut être une limite de la validation. Dans chaque cas d'étude, la construction d'un DJ a été faite dans de petites équipes. Nous n'avons pas testé notre approche sur de grosse équipe où il existe des experts qualité pour des aspects spécifiques (e.g., prototypage, études cliniques, marquage CE) Nous ne savons pas si nos résultats sont transposables sur ce type d'équipe. De plus, tandis qu'AXONIC construit et fait évoluer le DPJ durant le développement d'un projet courant, dans le cas avionique les DPJs sont basées sur des projets historiques. Comme évoqué précédemment, il est donc difficile d'établir une méthodologie standardisée pour la construction des DPJs. Cependant, avec les cas d'études très différents que nous avons présentés, nous pouvons affirmer que, peu importe le niveau de maturité de l'entreprise ou le modèle de développement, pour de petites équipes l'utilisation de notre approche a été utile.

9.3 Perspectives

9.3.1 Recherche

Relation entre DJ et processus de justification

Lors de l'étude sur le cas d'AXONIC nous avons mis en évidence que les DJ s peuvent devenir très gros et difficilement utilisables par la notation actuelle. Un moyen de résoudre ce problème d'utilisabilité est de mieux identifier le scope d'un DJ en le découplant en plusieurs DJ répondant à des exigences de justifications intermédiaires (e.g., un DPJ par norme métier dont certaines assertions servent à justifier la norme de plus haut niveau). Pour guider dans ce découpage, il est pour nous intéressant de creuser la piste de la relation des DJ s avec les processus métier. Le processus de justification permet de définir le flot traitant de production des justifications des points métiers spécifiques jusqu'à leur insertion dans la justification globale dans le SMQ. L'idée, comme montrée en Figure 9.4, est d'attacher des DJ s à chaque activité d'un processus de justification représenté en étant orienté données NIGAM et CASWELL [2003]. Ainsi nous pouvons expliciter à un grain plus fin l'apport en justification de chaque activité de ce processus en utilisant ces données comme des artefacts de justifications. D'activité en activité sont ainsi construits de plus petits DJ s et dont seulement les conclusions essentielles pour les DJ s sont réutilisées d'étape en étape. Pour creuser cette idée, une piste vient de la facilité d'interagir au niveau métamodèle entre notre approche et d'autres formalismes comme montré avec i* et SACM. Il s'agit donc de trouver un métamodèle pour les processus orientés données pertinent et d'amener la collaboration avec notre approche par celui-ci.

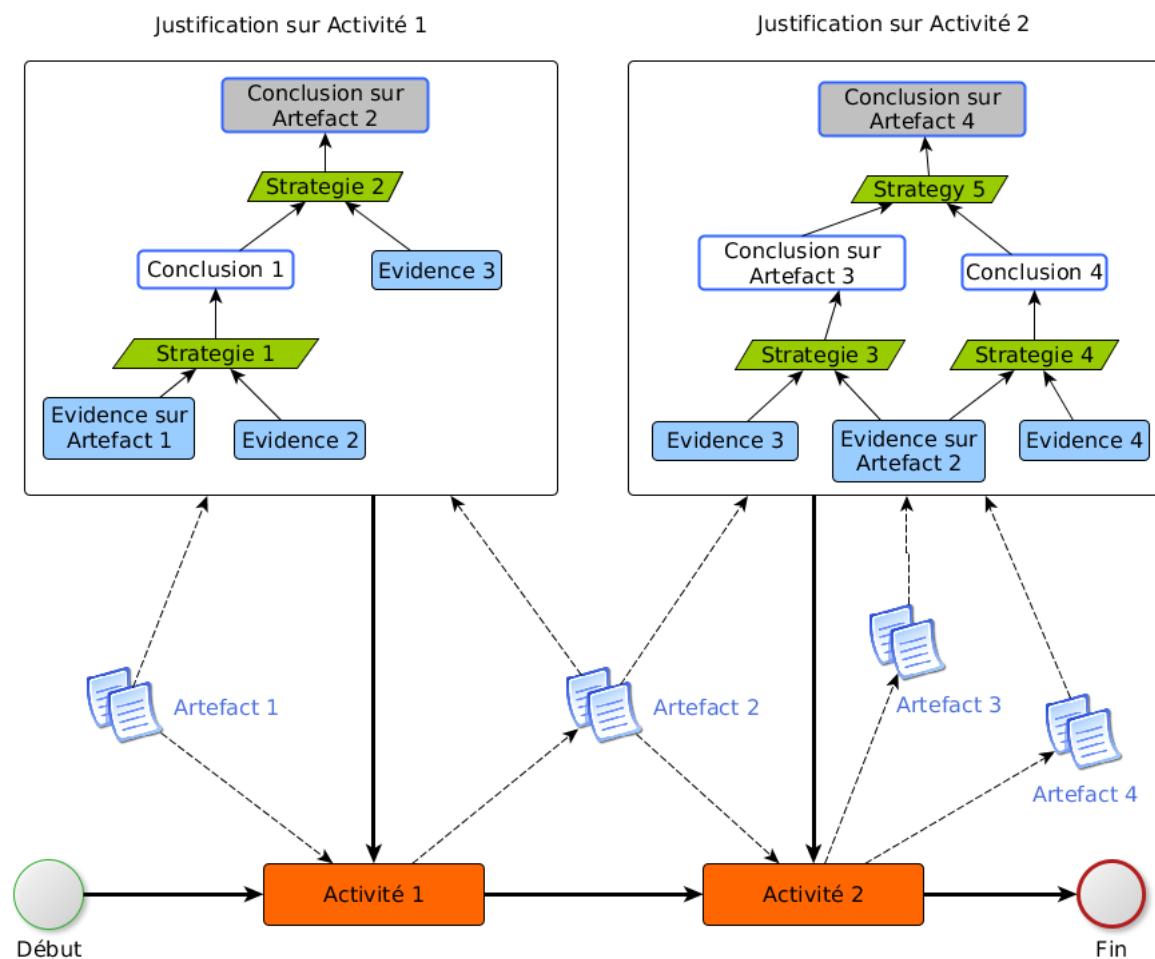


FIGURE 9.4 – Relation entre *BPMN* et *DJ*

Des modélisations pour les justifications adaptées au métier pour favoriser l'interaction entre domaines

Comme nous l'avons vu dans nos études, la conception des DPJ a toujours été accompagnée par un expert des modélisations de justification. Toute méthodologie ou outil permettant d'aider dans la conception de ces DPJs serait donc une avancée pour aider dans la prise en main de notre approche par un public plus large. La communauté ingénierie des exigences, par exemple, pourrait être utilisatrice de notre approche mais il est nécessaire de s'adapter à ses modélisations, notamment celles présentées en Section 3.2. En effet, comme le montre Wanderley et al. WANDERLEY et collab. [2014], il est important d'aligner la modélisation d'exigences sur des représentations usuelles pour l'utilisateur cible. Dans leur exemple, ils utilisent des modèles usuels pour représenter et structurer des besoins, les *Mind Maps*, pour les transformer en modèles formels dans KAOS puis SysML. Ces transformations successives de modèle permettent d'obtenir des représentations adaptées aux besoins et utilisateurs. Une perspective de notre travail reste donc de formaliser plus finement l'apport des justifications dans des formalisations comme pour l'ingénierie des exigences afin de les étendre. In fine, cela permettrait, i) à partir d'une représentation exprimant des exigences pour un système, d'extraire directement les éléments de justifications pour construire automatiquement le DPJ, ii) de donner une autre visualisation des exigences de justifications favorisant l'interaction entre la communauté ingénierie des exigences et justification et iii) d'avoir une vision d'ensemble en intégrant les exigences justification dans la définition des exigences globales d'un système.

Concrètement, si on prend l'exemple de la communauté autour de i*, elle tente de standardiser son approche pour capturer des exigences à partir de *goal*. Cette communauté en est ainsi venue à définir un noyau DALPIAZ et collab. [2016] sur lequel il est possible de mettre des extensions pour prendre en compte spécifiquement d'autres besoins comme la sécurité ou la performance GONÇALVES et collab. [2018]. Nous observons également pour KAOS une forte structuration formelle avec plusieurs méta-modèles proposés par la communauté qui permettrait également un rapprochement NWOKEJI et collab. [2013]. Ces pistes peuvent être un point de départ à explorer pour intégrer une extension capturant les éléments nécessaires pour les DJs au sein des modélisations pour l'ingénierie des exigences.

Formaliser d'autres Système de Justification (SJ) pour étendre la justification à d'autres applications

Comme introduit en conclusion du Chapitre 7 et complété lors de l'étude des systèmes expérimentaux en Section 9.1.1, il faut dériver le concept de SJ en fonction des usages de la justification. Ce travail de fond est autant d'étudier, sur d'autres cas, les cadres où les justifications sont utiles (e.g., domaine critique, systèmes à bases de connaissances) et autant de formaliser quels en sont les objectifs et usages dans ces cadres. Ce travail permettra de définir des SJ répondant plus spécifiquement aux besoins des cas bien différents de la justification.

Étudier la réutilisabilité des DJs de projet en projet

Comme introduit en 9.2, sur les trois études que nous avons menées, il n'a pas été possible d'étudier l'aspect réutilisabilité de notre approche. L'enjeu ici est d'étudier si le raffinement d'exigences de justifications que nous avons proposé permet de capturer des granularités suffisamment différentes pour permettre la réutilisabilité des DJ les plus abstraits. Dans nos travaux, nous avons montré que nous supportions un raffinement des exigences des normes jusqu'à la production des justifications associées à un projet pour

une entreprise en passant des étapes intermédiaires comme l'élicitation de pratiques internes. Les normes et guides de projet en projet peuvent naturellement être réutilisés et donc les **DJs** associés également. Pour certains il sera nécessaire de les dériver pour ajouter ou enlever des exigences de justifications qui ne sont pas pertinentes pour le contexte du nouveau projet. Les autres éléments raffinés (*e.g.*, pratiques internes, production des justifications) sont quant à eux très liés à une entreprise. Il faut donc mener des études pour mesurer cette réutilisabilité dans différents contextes.

Impact de l'incrémentalité des artefacts de justifications dans les justifications

Lors de l'étude sur le cas d'AXONIC, nous avons été confrontés à une forte incrémentalité au sein des artefacts de justification (*e.g.*, codes intégrés plusieurs fois par jour, document révisé plusieurs fois par sprint). Cette évolution au sein des artefacts n'est pas prise en compte dans les **DJs**. Plusieurs questions se posent alors : Est-ce que la notion d'artefact doit être intégrée dans les **DJs** et ainsi capturer directement cette incrémentalité ? Cela amènerait à définir formellement le concept d'artefact et d'étudier les relations entre les justifications basées sur les versions de ces artefacts. Est-ce que les **DJs** doivent être réactifs à cette incrémentalité ? Cela consisterait à construire autant de justifications que de versions d'artefacts et donc il y aurait plusieurs **DJ** dans un **SJ** pour capturer cette incrémentalité. Le choix reste ouvert et des études sur plusieurs cas concrets permettraient d'identifier les propriétés souhaitées et mécanismes en jeu.

9.3.2 Industrie

Hors des études conjointes que l'on a pu mener avec les partenaires industriels, l'adoption de notre approche passera par la facilité à migrer ou adapter de leur processus de justifications existant vers cette approche outillée. Cela passe par plus d'extensions d'outils comme nous avons pu le faire pour Jenkins et Redmine (*e.g.*, Atlassian pour la gestion de tickets, SonarQube pour l'analyse statique du code), mais aussi par des outils spécifiques notamment pour les domaines critiques. En effet, certains outils sont revendicables comme une bonne pratique pour les normes (*e.g.*, CodeSonar pour l'analyse statique du code qui propose des profils d'analyse pour le médical avec IEC 62304 ou encore avionique avec DO 178C). Intégrer facilement de la justification dans ses outils reconnus permet de réduire le fossé entre notre approche très générique et les pratiques spécifiques de ces entreprises et ainsi gagner en facilité d'adoption.

Par ailleurs, que ce soit dans un but interne pour gagner en confiance lors du développement ou pour certifier un produit, la justification dans le domaine de l'intelligence artificielle (IA) en est à ses débuts. Par exemple, Dvijtoham propose une approche pour, à partir d'une donnée, établir le voisinage autour de cette donnée (intervalle multi dimensionnel de variation dans la donnée) où la prédiction de l'IA restera robuste **DVIJOTHAM et collab. [2018]**. Cette approche de portée générale ne supporte néanmoins que des études locales à partir d'une donnée et le passage à l'échelle reste à démontrer. Par notre étude sur les systèmes expérimentaux notamment autour de ROCKFlows et les discussions que nous avons pu avoir avec les partenaires industriels des domaines critiques, nous avons pu identifier le besoin d'expliciter comment l'entraînement d'une IA a été mené, mais également les décisions prises par celle-ci en production. En effet, nous voyons avec notamment la voiture autonome tous les problèmes d'éthique (*e.g.*, sauve-t-on le conducteur ou le piéton?) **FLEETWOOD [2017]** et de gestion a posteriori des incidents (*e.g.*, en conduite autonome de niveau 3, autonomie conditionnée, en cas d'accident, qui est responsable IA ou conducteur? en conduite autonome de niveau 5, autonomie totale, en cas d'accident, qui est responsable IA ou piéton qui traversait?) **GREENBLATT [2016]**

que cela amène, qui sont difficiles à établir tant décortiquer l'impact de l'apprentissage, de la configuration et des prises de décisions en temps réel sur des cas litigieux est complexe. Certains travaux dans le domaine de l'IA abordent ces problématiques, mais ils restent encore très spécifiques [PEDERSEN et BENESTY \[2018\]](#); [RIBEIRO et collab. \[2018\]](#); [VANNI et collab. \[2018\]](#) et ne s'intègrent pour l'instant pas dans une démarche globale de justification. Ainsi, l'utilisation d'une approche comme la nôtre permettant de construire automatiquement des justifications et de les utiliser par la suite est un des leviers possibles pour rassembler les différentes techniques actuelles (et futures) dans un canevas de justification globale.

9.4 Synthèse

Pour finir, à travers cette thèse à vocation industrielle (CIFRE), nous avons montré comment la définition d'une sémantique formelle permet de capturer les concepts des justifications, le cycle de vie des activités de justifications et l'automatisation de ces activités permettent de définir un noyau pour ce domaine. À partir de celle-ci, nous avons proposé une architecture permettant d'adresser différents outils et autres formalismes pour rendre utilisable notre sémantique. Cette architecture a été implémentée afin de proposer une suite d'outils clé en main pour notre approche. Cette suite d'outils a été testée et validée dans le contexte des cas d'études de l'entreprise porteuse de cette thèse. Ces études n'avaient pas seulement pour but de mettre en oeuvre nos contributions de recherche, mais bien d'industrialiser notre approche pour AXONIC. Cet objectif est atteint étant donné qu'hier, aujourd'hui comme demain des justifications seront produites pour AXONIC par la *Justification Factory*. De nombreuses perspectives de recherche comme industrielle restent à explorer pour permettre d'étendre la sémantique, l'architecture ou encore les cas d'application. Pour conclure ce manuscrit, nous montrons dans le Tableau [9.1](#) un résumé des défis et contributions y répondant.

TABLEAU 9.1 – Synthèse des contributions

Défi ⁴	Refs dans les contributions (# de chapitre)	Synthèse de la contribution	Synthèse de la validation
$C_{1.1}$	5, 7	Définition d'une sémantique formelle pour la notation existante DJ, Définition d'un méta-modèle et explicitation des transformations vers SACM	Représentation graphique originelle préservée dans les études
$C_{1.2}$	5	Structuration à l'aide des DPJs et formalisation des relations entre exigences de justifications	Application aux normes IEC 62304 et ISO 13485 dans le cadre de pratiques Agiles
$C_{1.3}$	5	Définition de la relation de conformité \mathcal{R} sur les éléments de justification	Raffinement des exigences de la norme IEC 62304 vers les spécificités d'AXONIC
$C_{1.4}$	6	Définition d'opérations et leurs propriétés permettant de faire évoluer les justifications	Application sur l'évolution des exigences de justification pour IEC 62304 entre phases d'un projet
$C_{2.1}$	7	Définition d'une architecture permettant l'automatisation de la production de justifications	Application sur une plateforme d'intégration continue pour IEC 62304, Aujourd'hui 72% des 82 justifications produites, liées aux tests, le sont automatiquement
$C_{2.2}$	7	<i>Justification Factory</i> basé sur notre sémantique pour extraire, utiliser et valider des justifications, Architecture orientée service	Application au dossier de conception d'AXONIC pour ISO 13485 dans le but de générer le <i>Master file</i> , Intégration par les tests (Jenkins) et le SMQ (Redmine)
$C_{3.1}$	7	Définition d'une architecture supportant l'incrémentalité de la production des justifications	Application sur une plateforme d'intégration continue pour IEC 62304 intégrant plusieurs fois par jour les codes d'AXONIC
$C_{3.2}$	6, 7	Définition d'opérations contrôlant la conformité au cours du cycle de vie des justifications	Application au dossier de conception d'AXONIC dans le but d'assurer la conformité avec ISO 13485 permettant la vérification et validation automatique de l'ensemble de ces justifications, Automatisation approuvée par la responsable qualité utilisatrice

$C_{1.1}$: Préserver la sémantique et les représentations connues adaptées à la justification, $C_{1.2}$: Structurer des normes et pratiques internes, $C_{1.3}$: Supporter le raffinement des exigences de justifications, $C_{1.4}$: Supporter l'évolution des exigences de justifications, $C_{2.1}$: Supporter le passage à l'échelle dans la production des justifications, $C_{2.2}$: Proposer un environnement intégré pour l'utilisation des justifications, $C_{3.1}$: Supporter une production des justifications en continu, $C_{3.2}$: Faciliter la validation de justifications

9.5 Liste des publications

Les contributions présentées dans cette thèse ont été publiées dans des conférences internationales à comité de lecture ainsi que dans des conférences et revues nationales :

1. DUFFAU, C., C. CAMILLIERI M. BLAY-FORNARINO. 2017a, Improving confidence in experimental systems through automated construction of argumentation diagrams, *ICEIS 2017*, 1, 495–500
2. DUFFAU, C., B. GRABIEC M. BLAY-FORNARINO. 2017c, Towards embedded system agile development challenging verification, validation and accreditation: Application in a healthcare company, *ISSRE 2017-IEEE International Symposium on Software Reliability Engineering*, 1–4
3. DUFFAU, C., T. POLACSEK M. BLAY-FORNARINO. 2018a, Support of justification elicitation: Two industrial reports, *Advanced Information Systems Engineering - 30th International Conference, CAiSE 2018, Tallinn, Estonia, June 11-15, 2018. Proceedings*, Lecture Notes in Computer Science, Springer
4. DUFFAU, C. M. BLAY-FORNARINO. 2016b, Du recueil d'expertises à la production de logiciels fiables application aux systèmes de neurostimulation médicale, *INformatique des ORganisations et Systèmes d'Information et de Décision 2016 (INFORSID)*, 1–4 qui a été sélectionné pour être étendu en FAVRE, C., C. ARTAUD, C. DUFFAU, O. FRAISIER R. KOBBO KOMBI. 2017, *Forum jeunes chercheurs of Inforsid 2016*, 22, Ingénierie des Systèmes d'Information, Lavoisier
5. DUFFAU, C. M. BLAY-FORNARINO. 2016a, Aide à la décision pour le recueil incrémental d'expertises médicales, *INformatique des ORganisations et Systèmes d'Information et de Décision 2016 (INFORSID) - Ateliers*, 1–5
6. DUFFAU, C., C. CAMILLIERI M. BLAY-FORNARINO. 2017b, Vers l'argumentation automatique d'expérimentations: application à un portfolio de workflows, *6ème Conférence en Ingénierie du Logiciel (CIEL)*
7. DUFFAU, C., T. POLACSEK M. BLAY-FORNARINO. 2018b, Une sémantique pour les patrons de justification, *INformatique des ORganisations et Systèmes d'Information et de Décision 2018 (INFORSID)*, 121–147

Bibliographie

- AAMI, A. T. 2012, «Guidance on the use of agile practices in the development of medical device software», *Association for the Advancement of Medical Instrumentation*. [21](#), [22](#), [78](#)
- AHMAD, M., J.-M. BRUEL, R. LALEAU et C. GNAHO. 2012, «Using relax, sysml and kaos for ambient systems requirements modeling», *Procedia Computer Science*, vol. 10, p. 474–481.
- AKHIGBE, O., D. AMYOT et G. RICHARDS. 2015, «Information technology artifacts in the regulatory compliance of business processes : a meta-analysis», dans *International Conference on E-Technologies*, Springer, p. 89–104. [20](#)
- AKHIGBE, O., D. AMYOT et G. RICHARDS. 2018, «A systematic literature mapping of goal and non-goal modelling methods for legal and regulatory compliance», *Requirements Engineering*, p. 1–23. [24](#)
- ASSOCIATION FOR THE ADVANCEMENT OF ARTIFICIAL INTELLIGENCE, éd.. 1992, *Proceedings of the AAAI Spring Symposium on producing cooperative explanations*, Association for the Advancement of Artificial Intelligence. [10](#)
- ASSOCIATION FOR THE ADVANCEMENT OF ARTIFICIAL INTELLIGENCE, éd.. 1993, *Proceedings of the IJCAI'93 Workshop on Explanation and Problem Solving*, International Joint Conferences on Artificial Intelligence. [10](#)
- BALCI, O. 1998, «Verification, validation, and accreditation», dans *Proceedings of the 30th conference on Winter simulation*, IEEE Computer Society Press, p. 41–4. [2](#)
- BASILI, V. R., G. CALDIERA et H. D. ROMBACH. 1994, «Experience factory», *Encyclopedia of software engineering*. [97](#)
- BECK, K. 2003, *Test-driven development : by example*, Addison-Wesley Professional. [86](#)
- BOOCH, G. 1991, «Object oriented design with applications. redwood city», . [27](#)
- BOOCH, G. 2005, *The unified modeling language user guide*, Pearson Education India. [25](#)
- BRINKKEMPER, S. 1996, «Method engineering : engineering of information systems development methods and tools», *Information and software technology*, vol. 38, n° 4, p. 275–280. [59](#)
- CAMILLIERI, C., L. PARISI, M. BLAY-FORNARINO, F. PRECISO, M. RIVEILL et J. CANCELA VAZ. 2016, «Towards a Software Product Line for Machine Learning Workflows : Focus on Supporting Evolution.», dans *Proc. 10th Work. Model. Evol. co-located with ACM/IEEE 19th Int. Conf. Model Driven Eng. Lang. Syst. (MODELS)*, p. 65–70. [96](#)
- CHANDRASEKARAN, B., M. C. TANNER et J. R. JOSEPHSON. 1989, «Explaining control strategies in problem solving», *IEEE Intelligent Systems*, vol. 4, p. 9–15, 19–24, ISSN 1541-1672. [10](#)
- CHUNG, L., B. A. NIXON, E. YU et J. MYLOPOULOS. 2012, *Non-functional requirements in software engineering*, vol. 5, Springer Science & Business Media. [22](#)
- DALPIAZ, F., X. FRANCH et J. HORKOFF. 2016, «istar 2.0 language guide», *CoRR*, vol. abs/1605.07767. [23](#), [103](#)

- DUFFAU, C. et M. BLAY-FORNARINO. 2016a, «Aide à la décision pour le recueil incrémental d'expertises médicales», dans *INformatique des ORganisations et Systèmes d'Information et de Décision 2016 (INFORSID) - Ateliers*, p. 1–5.
- DUFFAU, C. et M. BLAY-FORNARINO. 2016b, «Du recueil d'expertises à la production de logiciels fiables application aux systèmes de neurostimulation médicale», dans *INformatique des ORganisations et Systèmes d'Information et de Décision 2016 (INFORSID)*, p. 1–4.
- DUFFAU, C., C. CAMILLIERI et M. BLAY-FORNARINO. 2017a, «Improving confidence in experimental systems through automated construction of argumentation diagrams», dans *ICEIS 2017*, vol. 1, p. 495–500.
- DUFFAU, C., C. CAMILLIERI et M. BLAY-FORNARINO. 2017b, «Vers l'argumentation automatique d'expérimentations : application à un portfolio de workflows», dans *6ème Conférence en Ingénierie du Logiciel (CIEL)*.
- DUFFAU, C., B. GRABIEC et M. BLAY-FORNARINO. 2017c, «Towards embedded system agile development challenging verification, validation and accreditation : Application in a healthcare company», dans *ISSRE 2017-IEEE International Symposium on Software Reliability Engineering*, p. 1–4. [81, 86](#)
- DUFFAU, C., T. POLACSEK et M. BLAY-FORNARINO. 2018a, «Support of justification elicitation : Two industrial reports», dans *Advanced Information Systems Engineering - 30th International Conference, CAiSE 2018, Tallinn, Estonia, June 11-15, 2018. Proceedings*, Lecture Notes in Computer Science, Springer.
- DUFFAU, C., T. POLACSEK et M. BLAY-FORNARINO. 2018b, «Une sémantique pour les patrons de justification», dans *INformatique des ORganisations et Systèmes d'Information et de Décision 2018 (INFORSID)*, p. 121–147.
- DVIJOTHAM, K., R. STANFORTH, S. GOWAL, T. MANN et P. KOHLI. 2018, «A dual approach to scalable verification of deep networks», *arXiv preprint arXiv:1803.06567*. [104](#)
- EMMET, L. et G. CLELAND. 2002, «Graphical notations, narratives and persuasion : a pliant systems approach to hypertext tool design», dans *HYPertext 2002, Proceedings of the 13th ACM Conference on Hypertext and Hypermedia*, édité par J. Blustein, R. B. Allen, K. M. Anderson et S. Moulthrop, ACM, p. 55–64. [13](#)
- FAVRE, C., C. ARTAUD, C. DUFFAU, O. FRAISIER et R. KOBBO KOMBI. 2017, *Forum jeunes chercheurs of Inforsid 2016*, vol. 22, Ingénierie des Systèmes d'Information, Lavoisier.
- FERNÁNDEZ-DELGADO, M., E. CERNADAS, S. BARRO et D. AMORIM. 2014, «Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?», *Journal of Machine Learning Research*, vol. 15, p. 3133–3181. URL <http://jmlr.org/papers/v15/delgado14a.html>. [95, 96](#)
- FLEETWOOD, J. 2017, «Public health, ethics, and autonomous vehicles», *American journal of public health*, vol. 107, n° 4, p. 532–537. [104](#)
- FOWLER, M. et M. FOEMMEL. 2006, «Continuous integration», *Thought-Works*, vol. 122, p. 14. [27](#)
- GONÇALVES, E., J. ARAÚJO et J. CASTRO. 2018, «Towards extension mechanisms in istar 2.0.», dans *iSTAR@ CAiSE*. [103](#)

- GREENBLATT, N. A. 2016, «Self-driving cars and the law», *IEEE spectrum*, vol. 53, n° 2, p. 46–51. [104](#)
- HEAVEN, W. et A. FINKELSTEIN. 2004, «Uml profile to support requirements engineering with kaos», *IEE Proceedings-Software*, vol. 151, n° 1, p. 10–27. [22](#)
- HOLLOWAY, C. M. 2013, «Making the implicit explicit : Towards an assurance case for do-178c», . [21](#), [22](#), [64](#)
- JOUAULT, F., F. ALLILAIRE, J. BÉZIVIN et I. KURTEV. 2008, «Atl : A model transformation tool», *Science of computer programming*, vol. 72, n° 1-2, p. 31–39. [70](#)
- KELLY, T. et R. WEAVER. 2004a, «The goal structuring notation / - a safety argument notation», dans *Proc. of Dependable Systems and Networks 2004 Workshop on Assurance Cases*. [11](#), [13](#)
- KELLY, T. et R. WEAVER. 2004b, «The goal structuring notation—a safety argument notation», dans *Proc. of the dependable systems and networks 2004 workshop on assurance cases*, Citeseer. [64](#)
- KIRSCHNER, P. A., S. BUCKINGHAM-SHUM et C. S. CARR. 2003, *Visualizing Argumentation : Software Tools for Collaborative and Educational Sense-Making*, Computer Supported Cooperative Work, Springer. [15](#)
- KNAUSS, E., G. LIEBEL, K. SCHNEIDER, J. HORKOFF et R. KASAULI. 2017, «Quality requirements in agile as a knowledge management problem : More than just-in-time», dans *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, p. 427–430. [2](#)
- KUNZ, W. et H. RITTEL. 1970, «Issues as elements of information systems», Working Paper 131, Institute of Urban and Regional Development, University of California, Berkeley, California. [15](#)
- VAN LAMSWEERDE, A. 2009, *Requirements Engineering - From System Goals to UML Models to Software Specifications.*, Wiley, ISBN 978-0-470-01270-3. [22](#)
- LARRUCEA, X., I. SANTAMARÍA et R. COLOMO-PALACIOS. 2016, «Assessing iso/iec29110 by means of itmark : results from an experience factory», *Journal of Software : Evolution and Process*, vol. 28, n° 11, p. 969–980. [97](#)
- LEE, C., L. GUADAGNO et X. JIA. 2003, «An agile approach to capturing requirements and traceability», dans *Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE 2003)*, vol. 20. [26](#)
- LEYTON-BROWN, K., E. NUDELMAN, G. ANDREW, J. MCFADDEN et Y. SHOHAM. 2003, «A portfolio approach to algorithm select», dans *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, p. 1542–1543. URL <http://dl.acm.org/citation.cfm?id=1630659.1630927>. [96](#)
- MACLEAN, A., R. M. YOUNG, V. M. E. BELLOTTI et T. P. MORAN. 1991, «Questions, options, and criteria : Elements of design space analysis», *Hum.-Comput. Interact.*, vol. 6, n° 3, p. 201–250, ISSN 0737-0024. [15](#)
- MASSEY, A. K., E. HOLTGREFE et S. GHANAVATI. 2017, «Modeling regulatory ambiguities for requirements analysis», dans *International Conference on Conceptual Modeling*, Springer. [24](#)

- MATOUSSI, A., F. GERVAIS et R. LALEAU. 2008, «A first attempt to express kaos refinement patterns with event b», dans *International Conference on Abstract State Machines, B and Z*, Springer, p. 338–338. [22](#)
- MATSUNO, Y. 2014, «A design and implementation of an assurance case language», dans *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*, IEEE, p. 630–641. [15](#)
- MATULEVICIUS, R. et P. HEYMANS. 2005, «Analysis of kaos meta-model», *Namur University : Computer Science Department : Belgiun*. [23](#)
- MEAD, N. R. et T. STEHNEY. 2005, *Security quality requirements engineering (SQUARE) methodology*, vol. 30, ACM. [23](#)
- NGUYEN, C. M., R. SEBASTIANI, P. GIORGINI et J. MYLOPOULOS. 2016, «Requirements evolution and evolution requirements with constrained goal models», dans *International Conference on Conceptual Modeling*, Springer, p. 544–552. [23](#)
- NIGAM, A. et N. S. CASWELL. 2003, «Business artifacts : An approach to operational specification», *IBM Systems Journal*, vol. 42, n° 3, p. 428–445. [101](#)
- NWOKEJI, J. C., T. CLARK et B. S. BARN. 2013, «Towards a comprehensive meta-model for kaos», dans *Model-Driven Requirements Engineering (MoDRE), 2013 International Workshop on*, IEEE, p. 30–39. [23, 103](#)
- (OMG), O. M. G. 2011, «Business process modelling notation (bpmn)», . [91](#)
- (OMG), O. M. G. 2013, «Structured assurance case meta-model (sacm)», . [14](#)
- PDG, S. G.-V. 2010, «Generic methodology for verification and validation (gm-vv) to support acceptance of models, simulations, and data, handbook», cahier de recherche, SISO-GUIDE-00X. 2-2010 DRAFT v1. 5.1. [22](#)
- PEDERSEN, T. L. et M. BENESTY. 2018, «lime : Local interpretable model-agnostic explanations»,. [105](#)
- PERELMAN, C. et L. OLBRECHTS-TYTECA. 1958, *Traité de l'argumentation : La nouvelle rhétorique*, Presses universitaires de France. [11](#)
- POHL, K. 2010, *Requirements engineering : fundamentals, principles, and techniques*, Springer Publishing Company, Incorporated. [22](#)
- POLACSEK, T. 2016, «Validation, accreditation or certification : a new kind of diagram to provide confidence», dans *10th IEEE International Conference on Research Challenges in Information Science, RCIS*, p. 59–466. [11](#)
- POLACSEK, T., S. ROUSSEL, F. BOUSSIERRÉ, C. CUILLER, P. DEREUX et S. KERSUZAN. 2017, «Towards thinking manufacturing and design together : An aeronautical case study», dans *Conceptual Modeling - 36th International Conference, ER 2017*, vol. 10650, édité par H. C. Mayr, G. Guizzardi, H. Ma et O. Pastor, Springer, p. 340–353. [94](#)
- POLACSEK, T., S. SHARMA, C. CUILLER et V. TULOUP. «The need of diagrams based on toulmin schema application : an aeronautical case study», *EURO Journal on Decision Processes*, p. 1–26. [64](#)
- POUR LA QUALITÉ, M. F. 2005, «Lexicoguide pour bien démarrer sa démarche qualité», cahier de recherche. [3](#)

- REMPEL, P., P. MÄDER, T. KUSCHKE et J. CLELAND-HUANG. 2014, «Mind the gap : Assessing the conformance of software traceability to relevant guidelines», dans *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, ACM, ISBN 978-1-4503-2756-5, p. 943–954. [25](#)
- RHEINBERGER, H.-J. 1997, «Toward a history of epistemic things : Synthesizing proteins in the test tube», . [97](#)
- RIBEIRO, M. T., S. SINGH et C. GUESTRIN. 2018, «Anchors : High-precision model-agnostic explanations», dans *AAAI Conference on Artificial Intelligence*. [105](#)
- SOUTHWICK, R. W. 1991, «Explaining reasoning : an overview of explanation in knowledge-based systems», *Knowledge Eng. Review*, vol. 6, n° 1, p. 1–19. [10](#)
- STEINBERG, D., F. BUDINSKY, E. MERKS et M. PATERNOSTRO. 2008, *EMF: eclipse modeling framework*, Pearson Education. [70](#)
- TANG, A., Y. JIN et J. HAN. 2007, «A rationale-based architecture model for design traceability and reasoning», *Journal of Systems and Software*, vol. 80, n° 6, p. 918–934. [25](#)
- TOULMIN, S. E. 2003, *The uses of argument*, Cambridge University Press. [10](#)
- VAN ZEE, M., D. MAROSIN, F. BEX et S. GHANAVATI. 2016, «Rationalgrl : A framework for rationalizing goal models using argument diagrams», dans *Conceptual Modeling : 35th International Conference, ER 2016*, Springer, p. 553–560. [24](#)
- VANNI, L., M. DUCOFFE, D. MAYAFFRE, F. PRECIOSO, D. LONGRÉE, V. ELANGO, N. SANTOS, J. GONZALEZ, L. GALDO et C. AGUILAR. 2018, «Text deconvolution saliency (tds) : a deep tool box for linguistic analysis», dans *56th Annual Meeting of the Association for Computational Linguistics*. [105](#)
- DE LA VARA, J. L. et R. K. PANESAR-WALAWEGE. 2013, «Safetymet : A metamodel for safety standards», dans *International Conference on Model Driven Engineering Languages and Systems*, Springer, p. 69–86. [15](#)
- VERNAY, J.-P. et O. KETELS. 2007, «Savoir faire la chasse à la surqualité», cahier de recherche. [2](#)
- WANDERLEY, F., N. BELLOIR, J.-M. BRUEL, N. HAMEURLAIN et J. ARAÚJO. 2014, «Des buts à la modélisation système : une approche de modélisation des exigences centrée utilisateur», dans *Inforcid 2014*, p. 113–128.
- WARMER, J. B. et A. G. KLEPPE. 1998, «The object constraint language : Precise modeling with uml (addison-wesley object technology series)», .
- WEINSTOCK, C. B. et J. GOODENOUGH. 2009, «Towards an assurance case practice for medical devices», cahier de recherche, Software Engineering Institute. [13](#)
- WOHLIN, C., P. RUNESON, M. HÖST, M. C. OHLSSON, B. REGNELL et A. WESSLÉN. 2012, *Experimentation in software engineering*, Springer Science & Business Media. [97](#)
- WOLPERT, D. H. 1996, «The lack of a priori distinctions between learning algorithms», *Neural computation*. [95](#)
- YE, L. R. et P. E. JOHNSON. 1995, «The impact of explanation facilities in user acceptance of expert system advice», *MIS Quarterly*, vol. 19, n° 2, p. 157–172. [10](#)

YU, E. S. 1997, «Towards modelling and reasoning support for early-phase requirements engineering», dans *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on*, IEEE, p. 226–235. [23](#)

Annexe A

Annexes

Pour des questions de propriétés intellectuelles de l'entreprise AXONIC, certaines annexes seront enlevées dans la version publiée

A.1 Méta-modèle détaillé pour les Diagrammes de Justification (DJs)

La Figure A.1 montre le méta-modèle détaillé pour les DJs nous y retrouvons les concepts de la sémantique et des opérations présentés en Chapitre 5 et 6 dans les Figure 5.2 et 6.6. Y est ajouté le concept de SJ du Chapitre 7 par JustificationSystem. Nous y retrouvons le tuple de DPJ, DJ et B_{unused} ainsi que la notion de SJ complet par l'attribut *isComplete*.

Le code de *Justification Factory* est basé sur ce méta-modèle qui est étendu pour permettre des confort de programmation (e.g., spécialisation des Assertion, définition de Strategie, JustificationSystem pour un cas d'application) et des extensions (e.g., plusieurs types de SJ, systèmes de contraintes permettant de spécialiser des fonctionnalités comme \mathcal{F}_2 (Que peut-on justifier ?). Un extrait du modèle est montré en Figure A.2.

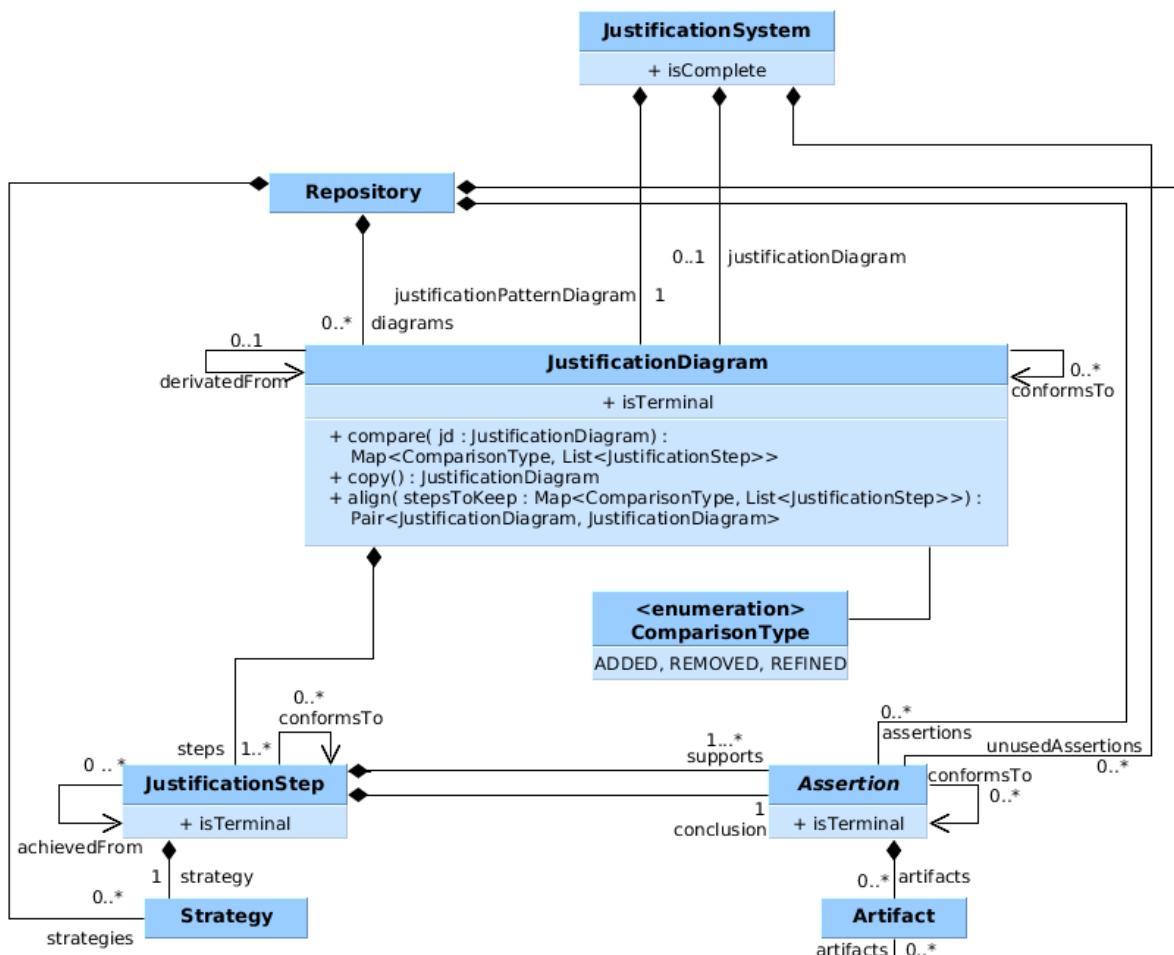


FIGURE A.1 – Méta-modèle détaillé DJ

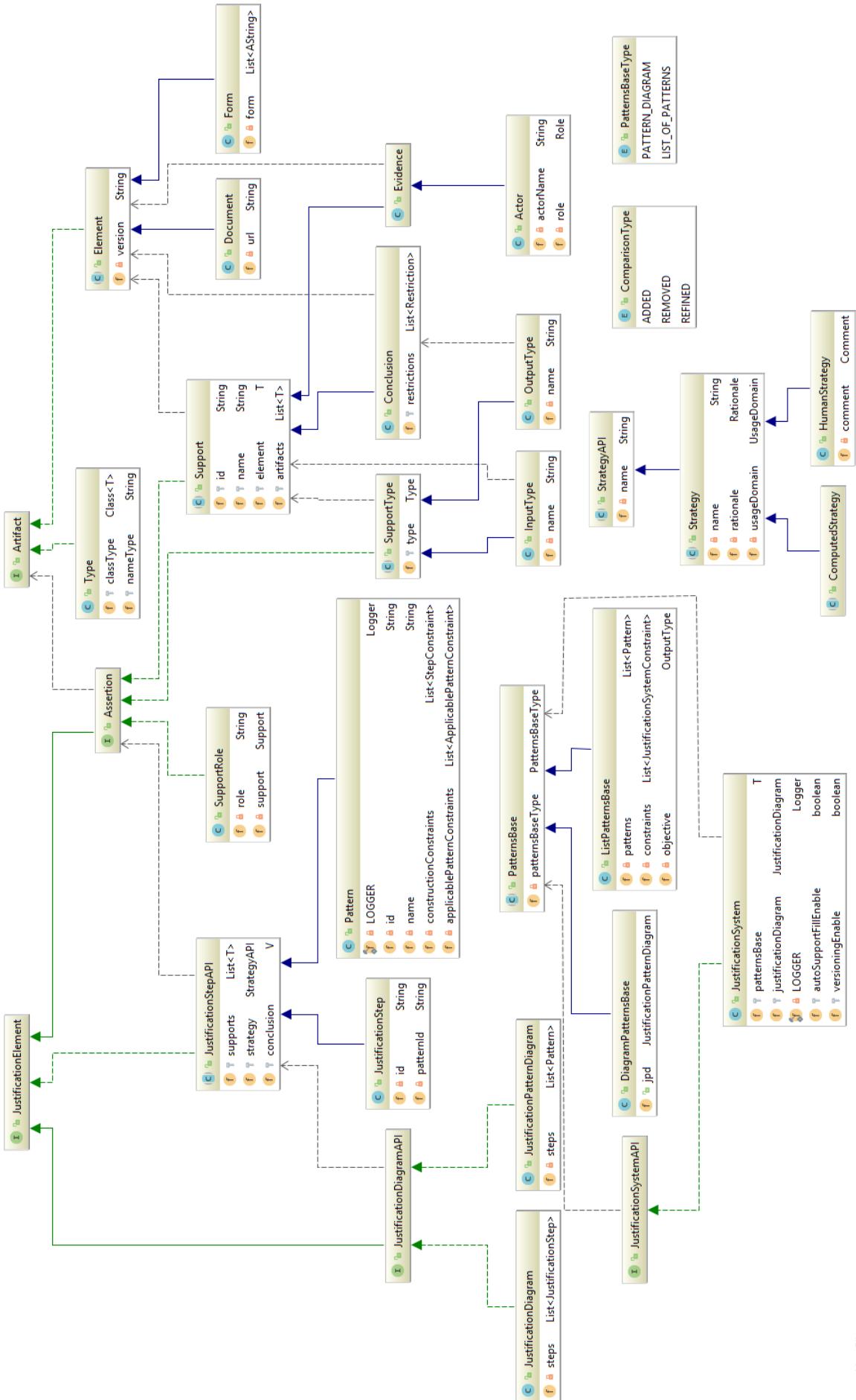


FIGURE A.2 – Méta-modèle des DJ au niveau du code

A.2 Interfaces des services exposés par *Justification Factory*

<code>GET</code>	<code>/justification/operation/achievement/{justification_diagram_name}</code>
<code>GET</code>	<code>/justification/operation/alignment/{justification_diagram_1_name}/{justification_diagram_2_na me}</code>
<code>GET</code>	<code>/justification/operation/comparison/{justification_diagram_1_name}/{justification_diagram_2_n ame}</code>
<code>GET</code>	<code>/justification/operation/derivation/{justification_diagram_1_name}/{justification_diagram_2_n ame}</code>
<code>POST</code>	<code>/justification/system</code>
<code>POST</code>	<code>/justification/system/{justification_system_name}</code>
<code>GET</code>	<code>/justification/system/{justification_system_name}/completed</code>
<code>GET</code>	<code>/justification/systems</code>
<code>GET</code>	<code>/justification/type</code>
<code>GET</code>	<code>/justification/types/{justification_artifact}</code>
<code>GET</code>	<code>/justification/{justification_system_id}</code>
<code>DELETE</code>	<code>/justification/{justification_system_id}</code>
<code>DELETE</code>	<code>/justification/{justification_system_id}/justificationDiagram</code>
<code>POST</code>	<code>/justification/{justification_system_id}/pattern</code>
<code>GET</code>	<code>/justification/{justification_system_id}/patterns</code>
<code>GET</code>	<code>/justification/{justification_system_id}/patterns/{pattern_id}</code>
<code>GET</code>	<code>/justification/{justification_system_id}/types</code>
<code>POST</code>	<code>/justification/{justification_system_id}/{pattern_id}/step</code>

FIGURE A.3 – API des services

A.3 Diagramme de séquence pour l'orchestration du *Justification bus*

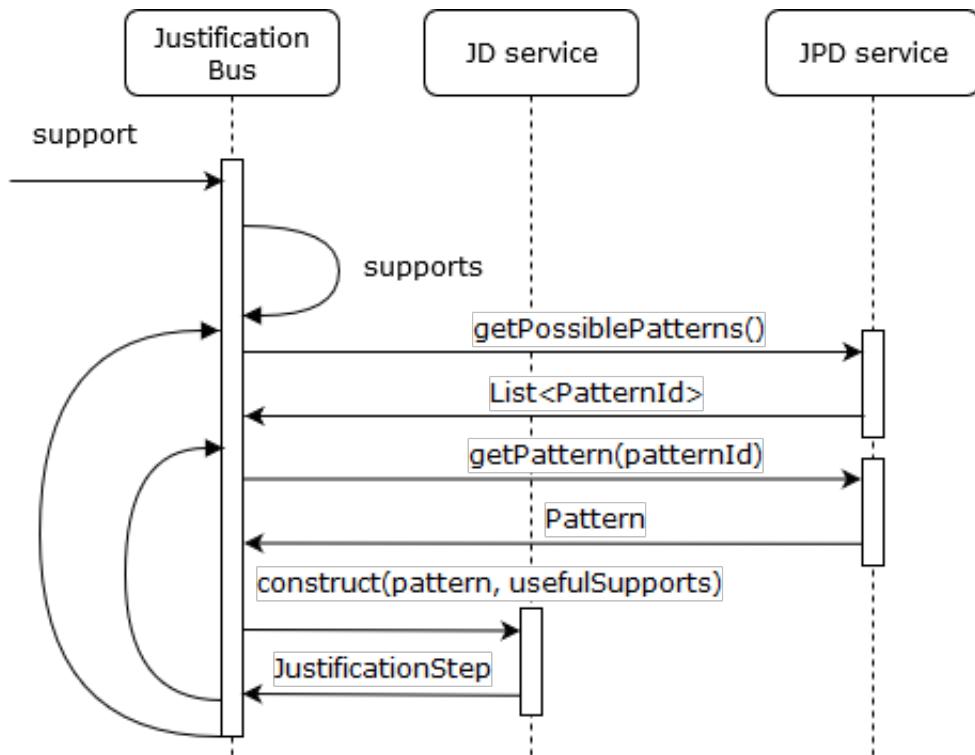


FIGURE A.4 – API des services

A.4 Autre exemple de I* et Diagramme de Justification (DJ)

La figure A.5 présente la collaboration entre un diagramme i* et un DJ. Le diagramme i* présente les exigences à atteindre : l'objectif *stimulator sold* et prend en compte également des qualités comme *ISO 13485 compliance*. L'atteinte de cette qualité est justifiée par un DJ. Pour justifier de la qualité *ISO 13485 compliance*, un DJ a été conçu. Nous pouvons vérifier que les 4 éléments i* *IEC 62304 compliance*, *IEC 60601 compliance*, *Quality process* et *Verify and validate system* qui font (make) la qualité *ISO 13485 compliance* sont référencés dans le DJ. De plus, la vérification est transitive, nous pouvons atteindre les tâches *Develop [hardware, mechanic, software]* qui font (make) les qualités intermédiaires pour atteindre *ISO 13485 compliance*. Après avoir atteint les assertions *IEC 60601 compliant*, *IEC 62304 compliant* et *System verified*, le pas de justification utilisant la stratégie *authorities audit* peut être automatiquement appliqué. En effet, le lien entre la ressource *Quality process* et l'assertion *Quality process* aide à automatiquement construire le pas de justification à appliquer.

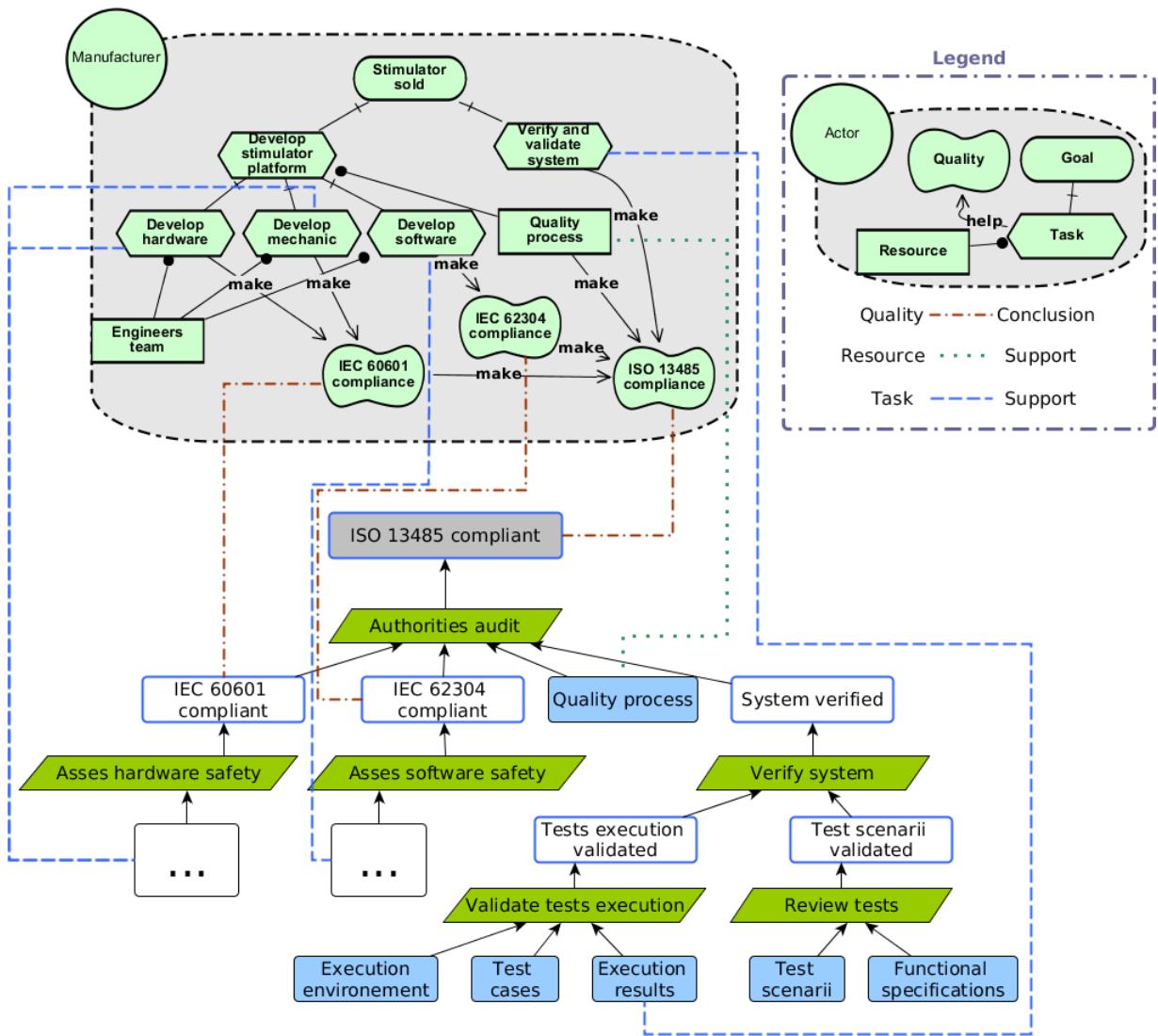


FIGURE A.5 – Exemple de collaboration entre i* et DJ

A.5 **Diagramme-Patron de Justification (DPJ) pour IEC 62304 du prototypage jusqu'au développement**

La Figure A.6 capture les exigences de justifications de la norme IEC 62304 pour les phases d'initialisation, de prototypage, de conception, de développement et de tests. Nous y retrouvons une deuxième norme en jeu, la ISO 14971 qui décrit la gestion des risques pour les **DMIA**. En effet, cette gestion des risques est à prendre en compte dans le développement logiciel pour un **DMIA** et donc ce **DPJ** représente des exigences entrelacées entre IEC 62304 et ISO 14971. Nous y retrouvons ainsi à la base une validation des spécifications générales ainsi que l'identification et prise en compte des risques qui mènent à de spécifications pour la sûreté. Ces deux validations sont ensuite utilisées pour mener des études de faisabilité, concevoir une architecture, proposer une implémentation et mettre en place la V&V. Ces quatre aspects sont les piliers finaux sur lesquels la sûreté d'un logiciel pour IEC 62304 peut être attestée.

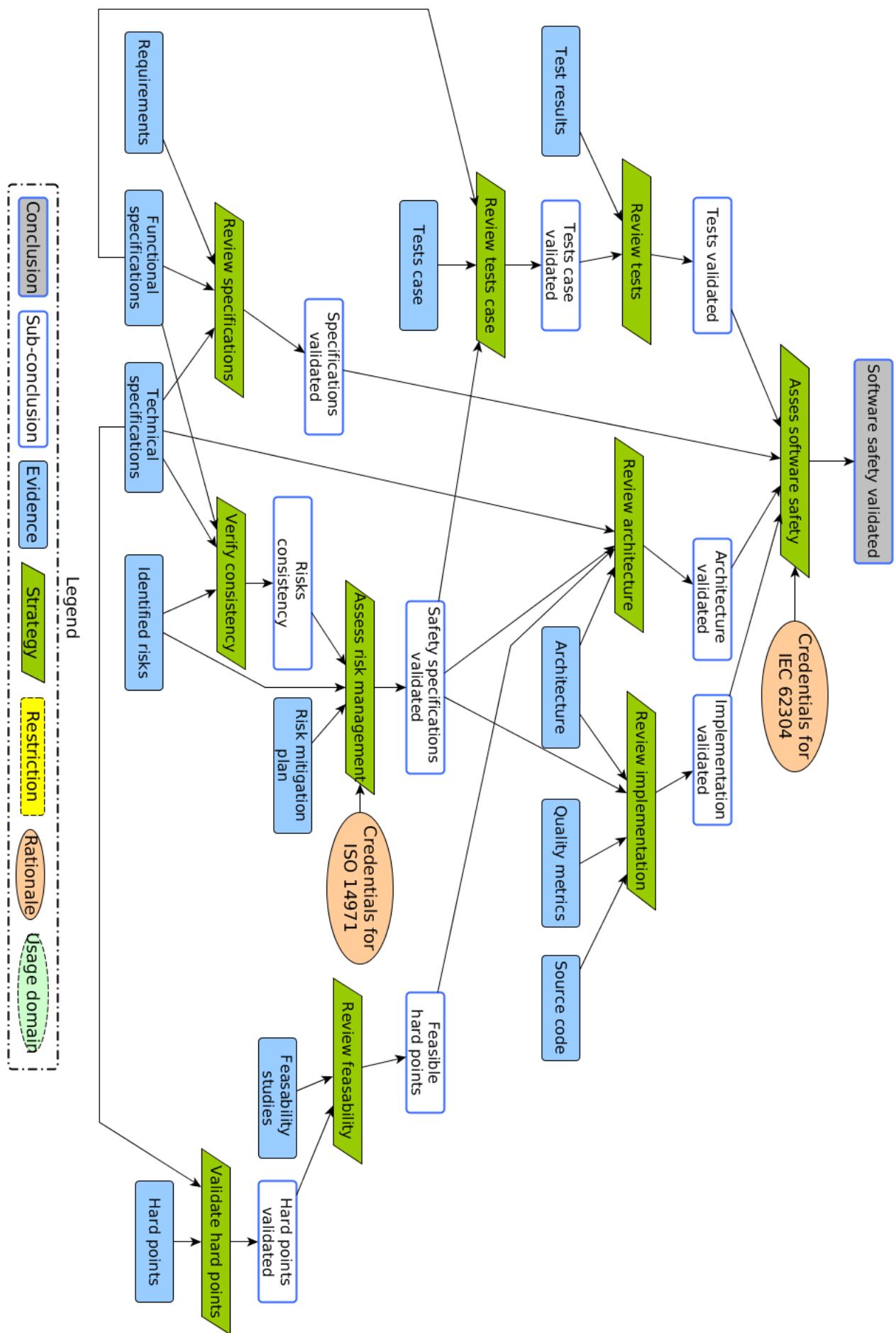


FIGURE A.6 – DPJ représentant les exigences de justification pour IEC 62304 (du prototypage jusqu'au développement)

A.6 Données brutes de l'étude préliminaire sur la compréhension des Diagrammes-Patron de Justification (DPJs)

La Table A.1 présente le questionnaire visant à étudier la compréhensibilité des DPJs et les réponses que l'on a pu récolté.

TABLEAU A.1 – Questionnaire pour l'étude sur la compréhension des Diagrammes-Patron de Justification (DPJs)

Question	Réponse nouvel arrivant
Avez-vous compris le contenu des DPJs?	OUI
Avez-vous compris la différence entre les différences et les liens entre les DPJs?	OUI
Avez-vous fait des parallèles entre les DPJs et la IEC 62304 et les pratiques internes d'AXONIC?	OUI mais certains termes, sémantiquement équivalents, étaient différents entre les documents
Les DPJs vous ont-ils aidés à mieux comprendre les exigences de justifications et la manière dont AXONIC y répond?	OUI
Êtes-vous d'accord avec cette assertion : Les DPJs capturent les liens implicites entre exigences de justifications	OUI
Êtes-vous d'accord avec cette assertion : La succession de DPJs explicite la réponse aux exigences normatives dans les pratiques internes d'AXONIC	OUI
Êtes-vous d'accord avec cette assertion : Les DPJs permettent de hiérarchiser des activités aidant ainsi à concevoir le processus de justifications?	NA (manque de recul sur les domaines critiques)

A.7 Diagramme-Patron de Justification (DPJ) complet pour le dossier de conception d'AXONIC

Nous donnons dans cette section, le DPJ structurant les exigences de justifications pour le dossier de conception complet d'AXONIC pour le projet d'études de nos travaux nommé SWAM. La Figure A.7 montre que le dossier de conception est justifié à l'aide de la *revue d'initialisation* présentée en Figure A.8, de la *revue des données d'entrée* en Figure A.10 et de la *revue de faisabilité* en Figure A.12. Ces exigences normatives sont raffinées au sein de leur diagramme respectif en des choix internes d'AXONIC (e.g., cycle de développement, différentes versions du projet). Par cet exemple, nous montrons que les DJs

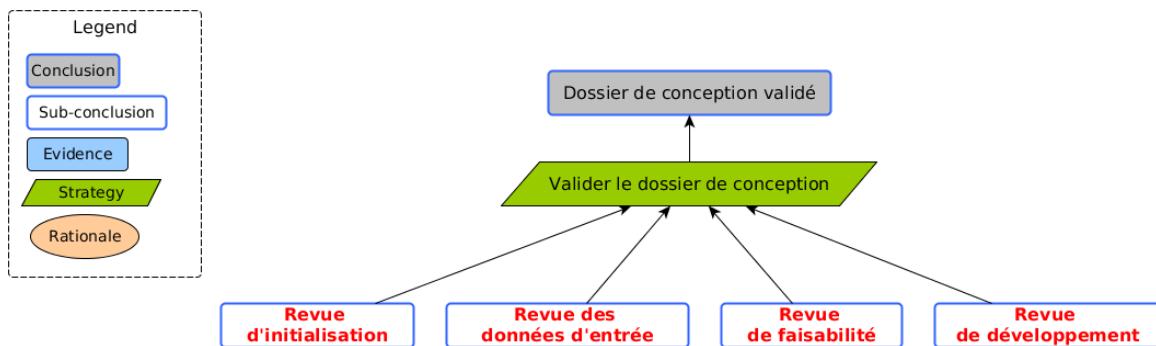


FIGURE A.7 – DPJ du dossier de conception

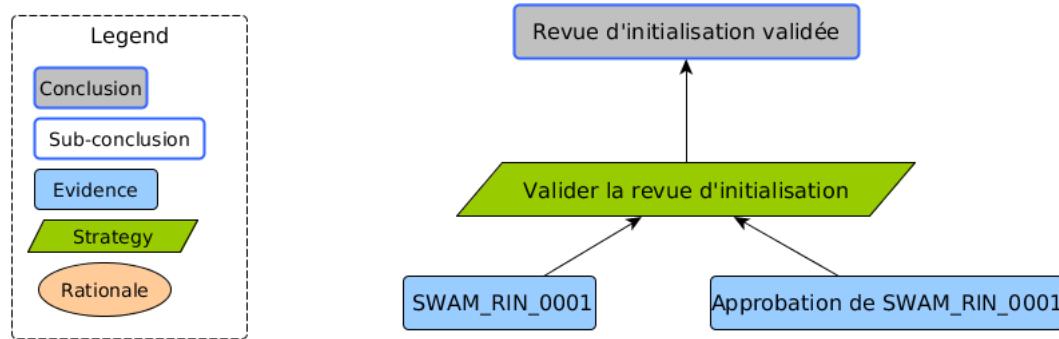


FIGURE A.8 – Sous-DPJ du dossier de conception portant sur la revue d'initialisation

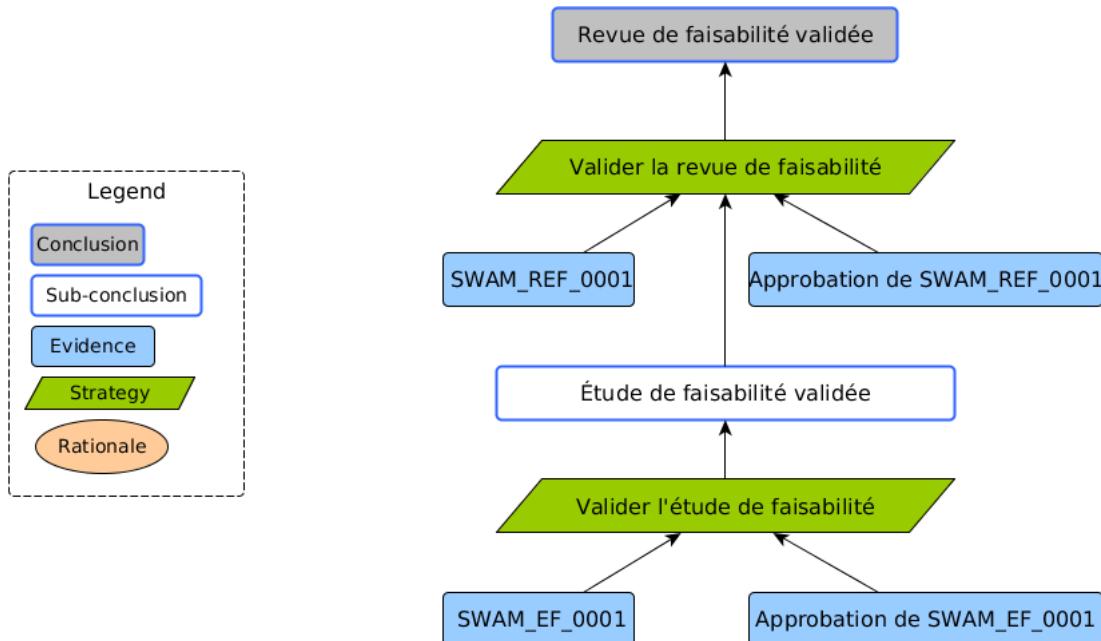


FIGURE A.9 – Sous-DPJ du dossier de conception portant sur la revue de faisabilité

peuvent devenir très conséquents, mais servent tout de même à capturer l'ensemble des exigences de justifications pour un projet toutes normes confondues. Ici, cela représente les 5 normes ISO 13485 Dispositifs médicaux - Systèmes de management de la qualité, ISO 14971 Application de la gestion des risques aux dispositifs médicaux , IEC 62304 Logiciels de dispositifs médicaux - Processus du cycle de vie du logiciel, IEC 60601 Appareils électromédicaux - Exigences générales pour la sécurité de base et les performances essentielles et IEC 62366 Application de l'ingénierie de l'aptitude à l'utilisation aux dispositifs médicaux.

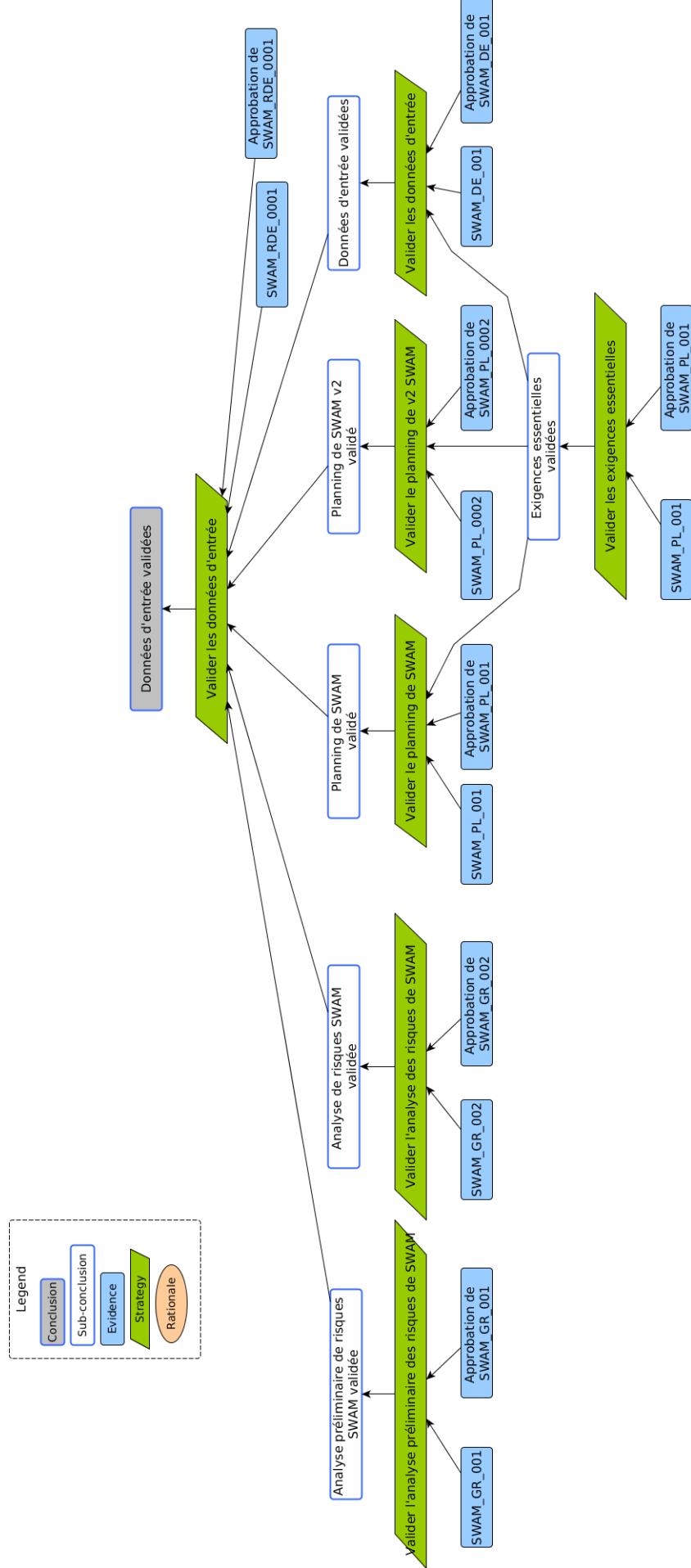


FIGURE A.10 – Sous-DPJ du dossier de conception portant sur la revue des données d'entrée

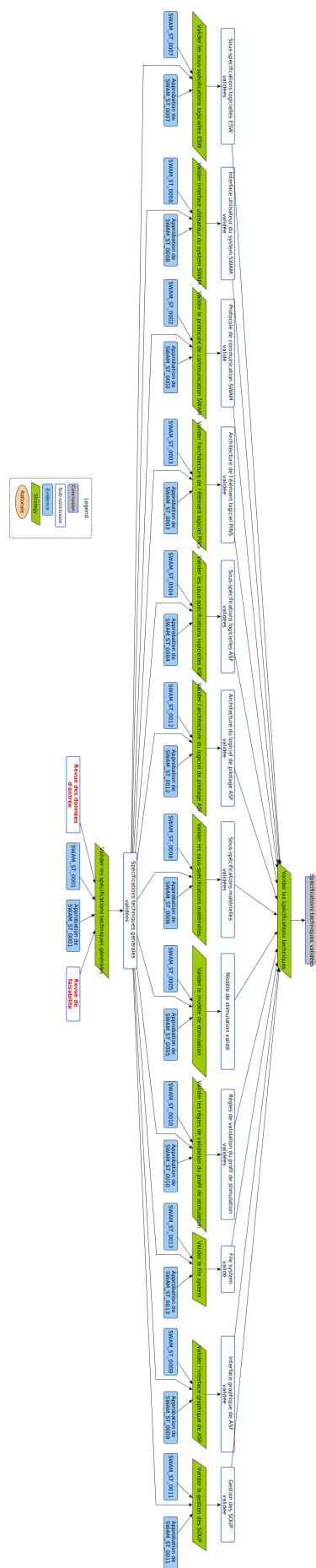


FIGURE A.11 – Sous-DP du dossier de conception portant sur la revue des spécifications techniques

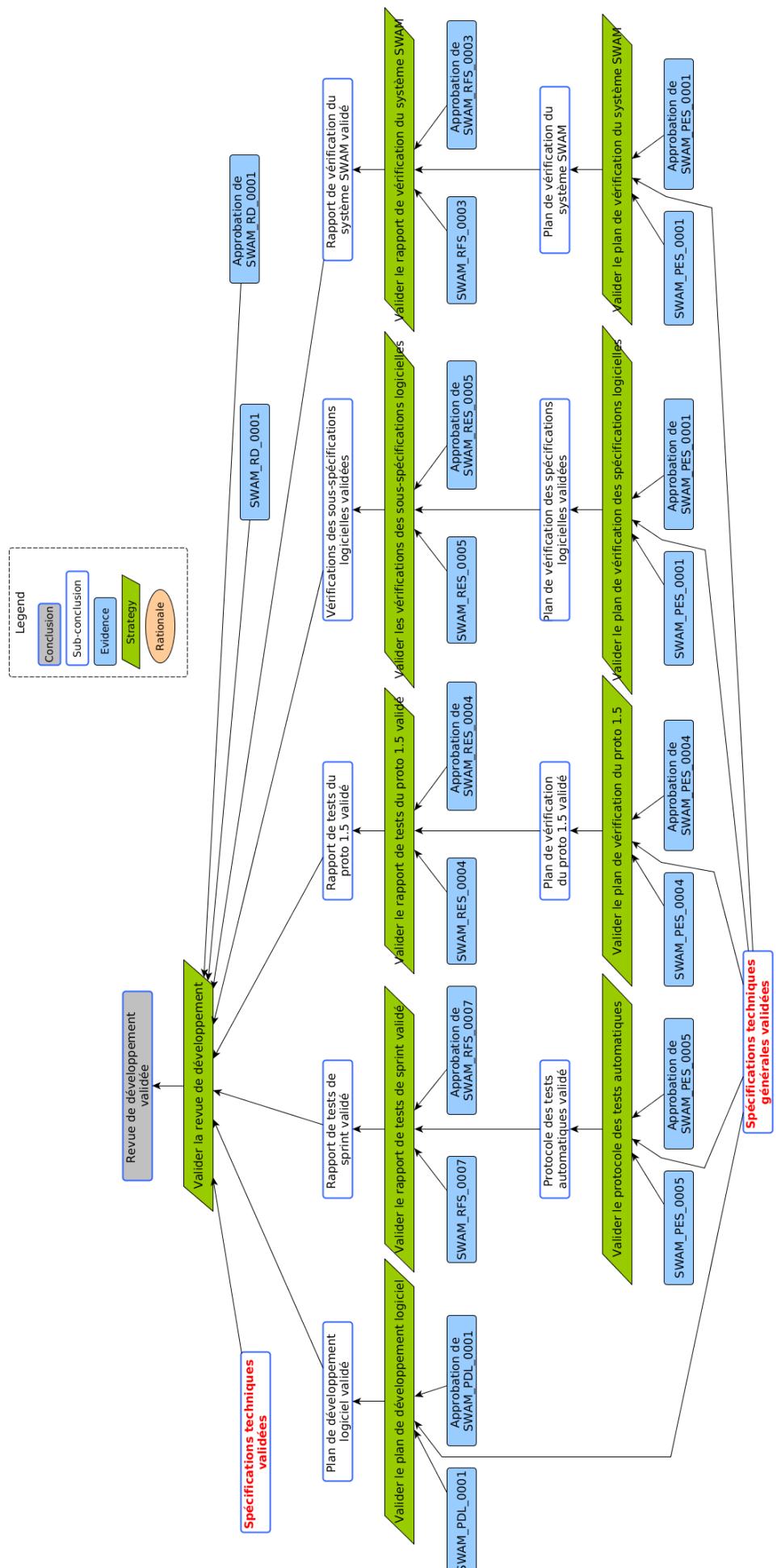


FIGURE A.12 – Sous-DP] du dossier de conception portant sur la revue de développement

A.8 Données brutes de l'étude sur la production automatique des justifications pour IEC 62304

TABLEAU A.2 – Données brutes de l'étude de la production automatique des justifications

Artefacts de justifications	Automatisation avant Justification Factory	Automatisation après Justification Factory
Tests unitaires (~2000)	OUI (mais seulement les artefacts)	OUI
Tests d'intégration (~100)	OUI (mais seulement les artefacts)	OUI
Tests systèmes (~50)	NON	OUI
Tests exploratoires (~10)	NON	NON
Sanity check (13)	NON	NON
Traçabilité livraison (6)	NON	OUI

La Table A.2 présente les artefacts de justifications présents dans le document de justification *Rapport de fin de sprint* d'AXONIC attestant de la validité des livrables logiciels de l'entreprise. Nous retrouvons ainsi des artefacts liés à la V&V de certains modules qui sont évalués individuellement sur leurs *tests unitaires* et *tests d'intégration*. Suite à ces évaluations isolées, ils sont assemblés pour mener des campagnes de *tests du système* entier où il faut d'abord avoir validé que les versions de ces modules sont compatibles (*traçabilité livraison*). Finalement, des vérifications sur l'exécution du binaire finale sont menées (*sanity check*) et des *tests exploratoires* également.

À partir de l'identification de ces artefacts de justifications et du DPJ pour IEC 62304 en Figure A.6, nous avons conçu un DPJ spécifique permettant de représenter les justifications que manipule ce document de *Rapport de fin de sprint*. Ainsi, dans la Figure A.13, nous visualisons un DPJ contenant les justifications à produire pour valider les livrables logiciels d'un DMIA chez AXONIC. Nous avons ensuite identifié ce qui a été produit automatiquement dans ce rapport et ce que nous pourrions produire automatiquement avec nos travaux. De cette analyse découlent les éléments présentés en Table A.2. Nous retrouvons globalement d'aucune justification n'est produite automatiquement par AXONIC avant nos travaux, seulement certains artefacts liés aux tests unitaires et d'intégration. Ce grand nombre d'artefacts (environ 2000 pour les tests unitaires et environ 100 pour les tests d'intégration) déjà produits automatiquement par Jenkins ont été rassemblés dans des rapports permettant une meilleure lisibilité. Finalement chaque jeu de tests unitaires ou d'intégration ne représentait qu'un seul artefact dans le rapport du fait de leur agrégation au préalable. Pour les artefacts restants nous comptons ainsi environ 79 qui devaient être produits à la main sur les 82 à produire au total. Par nos travaux, nous avons d'abord automatisé la production des artefacts pour les tests système. Puis sur la base du DPJ de la Figure A.13, nous avons automatisé la construction des justifications possibles. Il ne reste ainsi que les justifications nécessitant l'intervention humaine pour la production des artefacts de justification ou des justifications elles-mêmes (e.g., tests exploratoires, sanity check).

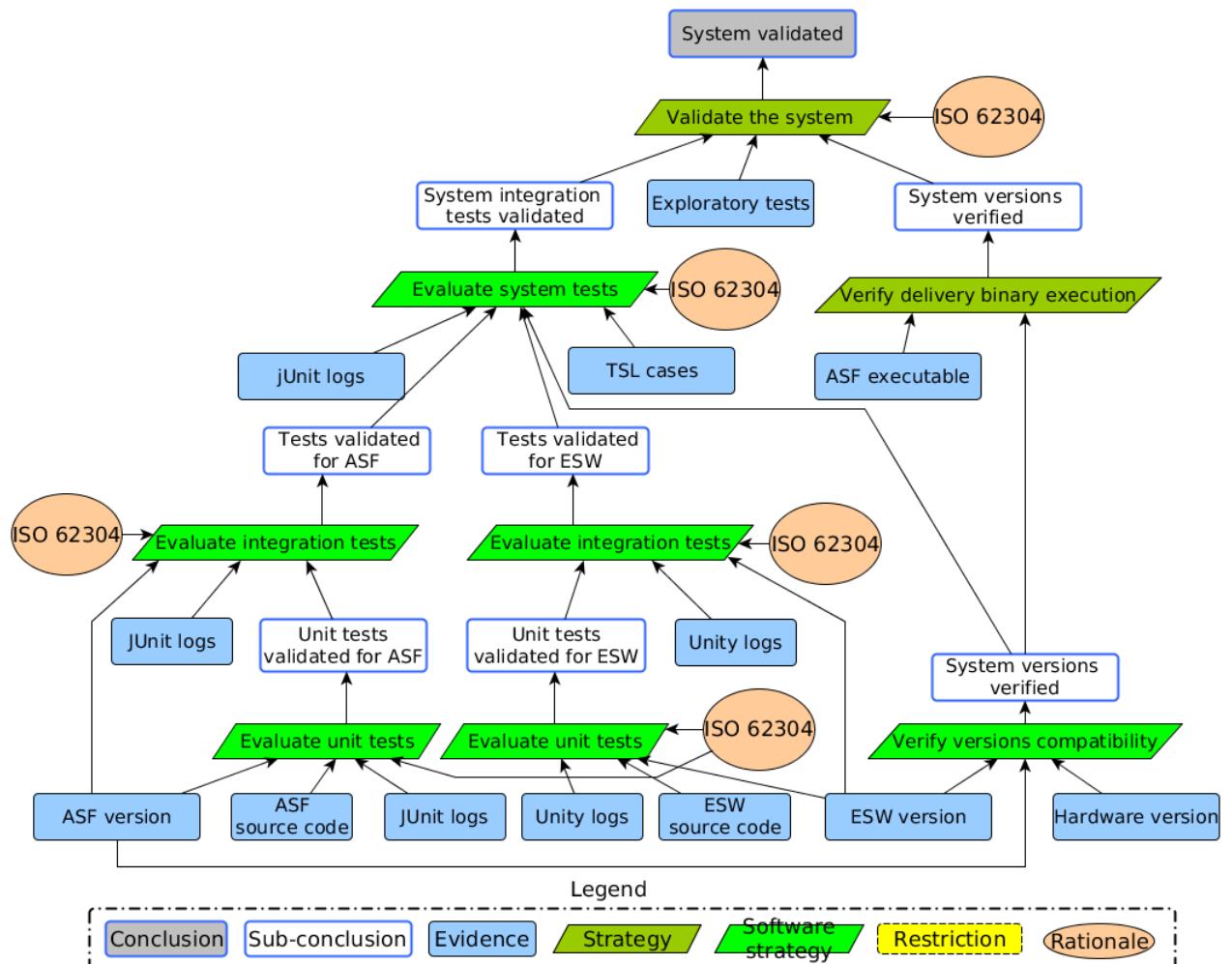


FIGURE A.13 – DPJ pour valider les livrables logiciels d'AXONIC

A.9 Données brutes de l'étude sur l'intégration au SMQ d'AXONIC

TABLEAU A.3 – Données brutes de l'étude du SMQ

		Avant Justification Factory	Après Justification Factory
Nombre total de documents		358	377
Nombre de documents de justifications (présence d'une section "Approbation")		165	184
Nombres de documents de justification en erreur	pas d'auteur	15	8
	pas signé par l'auteur	4	5
	signé par l'auteur, mais non approuvé	11	5
	pas de date de signature	4	2
	approuvé, mais non verrouillé	22	10
Taux d'erreur dans les documents de justifications (en %)		33.9	16.3
Nombre de documents de justifications dans le MasterFile		159	184
Nombre de documents de justifications verrouillés oubliés dans le MasterFile		9	0
Nombre de documents de justifications non verrouillés présents dans le MasterFile		13	0
Taux d'erreur dans le MasterFile (en %)		13.8	0.0

Dans la Table A.3, nous retrouvons les données brutes sur le contenu du SMQ dans Redmine. Avant l'utilisation de notre approche, de nombreuses incohérences étaient présentes dans les documents de justifications (e.g., pas d'auteur, pas signé, pas de date de signature, pas verrouillé), mais également dans le *Master File* où les documents de justifications associées contenaient des erreurs ou n'étaient pas un état correct pour y être insérés.

Après la mise en place de notre approche, le nombre de documents en erreurs à diminuer. Nous montrons dans le tableau, l'état du SMQ au bout de 4 sprints pour en avoir un état des lieux pour l'analyse. Notons que cet état des lieux est pendant le développement du projet, il faut donc évaluer notre solution au regard de cet aspect continu des justifications.

Sur l'absence d'auteurs, cela a diminué mais c'est très dépend du responsable qualité qui doit analyser le document pour trouver la personne concernée. Dès lors, nous n'avons que pu notifier le responsable qualité pour lui pointer ces documents en erreur. Le nombre de documents non signé par l'auteur stagne puisque cela peut correspondre à un document en court d'écriture. Le reste des erreurs sur les documents de justifications diminuent drastiquement, car ces erreurs correspondent aux transitions des documents entre les personnes (auteur vers approuveurs, approuveurs vers responsable qualité). Les erreurs résiduelles restent liées à l'aspect continu de ce flot.

Sur le *Masterfile*, nous avons supprimé l'ensemble des erreurs. Du fait de la génération automatique des justifications contenues dans celui-ci par notre approche ceci semble logique. En effet, nous n'ajoutons dans le DJ associé seulement les justifications reposant sur des documents dans un état final (verrouillé).

