ANNÉE 2016

UNIVERSITÉ DE RENNES 1

UNIVERSITE BRETAGNE LOIRE

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Bretagne Loire

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

École doctorale Matisse

présentée par

Sana BEN NASR

préparée à l'unité de recherche Inria
Inria Rennes Bretagne Atlantique
ISTIC

Mining and Modeling Variability from Natural Language Documents: Two Case Studies

Thèse soutenue à Rennes
le 5 Avril 2016

devant le jury composé de :

Marianne HUCHARD
Professeur, Université de Montpellier / *Présidente*

Camille SALINESI
Professeur, Université Paris 1 Panthéon Sorbonne / *Rapporteur*

Mehrdad SABETZADEH
Senior Research Scientist, Université du Luxembourg / *Rapporteur*

Pascale SÉBILLOT
Professeur, INSA de Rennes / *Examinatrice*

Benoit BAUDRY
Chargé de recherche, INRIA Rennes / *Directeur de thèse*

Mathieu ACHER
Maître de conférence, Université de Rennes 1 / *Co-directeur de thèse*

Acknowledgements

This thesis would not have been completed without the help of others. I would like to take this opportunity to express my gratitude towards them and acknowledge them.

First of all, I would like to offer my deepest gratitude to Benoit Baudry for supervising this research work and for guiding my steps over the last three years. But above all I would like to thank him for his very professional manner of conducting research which he also taught me, and for his very insightful criticism, which have lead to highly improve the quality of my work. Patient, caring, supportive, understanding, and positive are just a few words to describe him.

I am extremely grateful to Mathieu Acher, my thesis co-supervisor. Our regular meetings and discussions have always been very constructive and fruitful. His guidance and helpful advices as well as his patience and faith in me have constantly helped me feel positive and confident. He always knew how to motivate me and make me focus on the interesting topics to work on, helping me to overcome the difficult times in my thesis. I would like to mention that it was a privilege and honor for me to have been his first Phd student. I could never have imagined any better supervisors.

I am grateful to the jury members Prof. Camille Salinesi, Dr. Mehrdad Sabetzadeh, Prof. Pascale Sébillot and Prof. Marianne Huchard for having accepted to serve on my examination board and for the time they have invested in reading and evaluating this thesis. You all gave me a lot of interesting and inspiring feedback that I will keep in mind in the future.

I greatly appreciate and wish to thank all the (former and current) members of the DiverSE research team for providing a great and friendly work atmosphere, and for all the different fruitful discussions. A special thanks to Guillaume Bécan and Nicolas Sannier for all the support they have offered me. I enjoyed a great deal working with you!

My eternal gratitude goes to my parents Zeineb and Youssef for their support, comprehension, confidence and, especially for instilling into me that education is the basis of progress. You have always been behind me and believed in me. You encourage me to succeed in everything I do. Thanks to my sister Sonia and my brother Selim for their unconditional love, and support. I love you so much!

I am deeply in debt with my beloved husband Faker. You are my closest friend, and my confident. Thank you very much for your support, for your energy, for your love and patience. You have been right by my side all along the way. You are my daily motivation and inspiration and for sure, I would not be who I am today without you.

Résumé en Français

Contexte

Une Ligne de Produits (PL) est définie comme une collection de systèmes partageant un ensemble de propriétés communes et satisfaisant des besoins spécifiques pour un domaine particulier. L'analyse du domaine vise à identifier et organiser les caractéristiques communes et variables dans un domaine. La notion de *Feature* est définie comme étant tout concept, caractéristique ou propriété proéminente du système qui est visible pour tous les acteurs. L'analyse du domaine est généralement effectuée par des experts à partir de documents informels (fiches techniques, cahier des charges, résumés d'entretiens, descriptions) qui sont écrits en langue naturelle, sans formalisme particulier. Dans la pratique, le coût initial et le niveau d'effort manuel associés à cette analyse constituent un obstacle important pour son adoption par de nombreuses organisations qui ne peuvent en bénéficier.

Plusieurs travaux se sont intéressés à l'identification et la spécification de la variabilité des systèmes [ACP+12a, YZZ+12, DDH+13a, LHLG+14, BABN15]. Cependant, peu d'entre eux sont attachés à proposer des techniques automatisées pour la construction de modèles de variabilité à partir des documents non structurés et ambigus. Les techniques de traitement automatique du langage naturel et d'exploration de données ont été utilisées par les chercheurs pour la recherche d'information, l'extraction terminologique, le partitionnement de données, l'apprentissage des règles d'association, etc., [T+99, RS10, KU96, PY13, AIS93].

Le but de cette thèse est d'adopter et exploiter ces techniques pour automatiquement extraire et modéliser les connaissances relatives à la variabilité à partir de documents informels dans différents contextes. En particulier, l'objectif est d'identifier les features, les points communs, les différences et les dépendances entre features d'une ligne de produits. En effet, les techniques de traitement automatique du langage naturel employées lors de l'extraction de la variabilité dépendent de la nature du texte et du formalisme considéré. L'enjeu est de réduire le coût opérationnel de l'analyse du domaine, en particulier des opérations manuelles effectuées par les experts de ce domaine en y apportant un support automatisé pour faciliter l'identification et la spécification de la variabilité des systèmes.

Nous étudions l'applicabilité de notre idée à travers deux études de cas pris dans deux contextes différents: (1) la rétro-ingénierie des *Modèles de Features (FMs)* à partir des exigences réglementaires de sûreté dans le domaine de l'industrie nucléaire civil et (2)

l'extraction de *Matrices de Comparaison de Produits (PCMs)* à partir de descriptions informelles de produits. FMs et PCMs sont des formalismes fondamentaux pour la spécification des caractéristiques communes et variables des produits de la même famille.

La première étude de cas traite des exigences réglementaires de sûreté pour la certification des systèmes de contrôle-commande importants pour la sûreté de fonctionnement d'une centrale nucléaire de production d'électricité et a eu lieu dans le cadre du projet industriel CONNEXION. Les exigences réglementaires sont édictées par les autorités nationales et complétées par des recommandations pratiques et des textes normatifs.

Ces exigences, bien que rédigées en langue naturelle non contraintes, présentent un haut niveau de structure et de rigueur d'écriture. Cependant, elles n'expriment aucune propriété fonctionnelle ou non fonctionnelle sur les systèmes mais précisent des objectifs à atteindre ou des moyens à mettre en œuvre. Or, dans la pratique, ces exigences de sûreté sont de haut niveau et sont ambigus. De plus, les textes supportant ces exigences (réglementations nationales, et internationales, normes internationales) varient d'un pays à un autre. Des pays, tels que la France et les autres pays d'Europe de l'ouest suivent le courant CEI/AIEA de la réglementation. D'autre part, un courant ISO/IEEE est suivi aux Etats-Unis ou en Asie. Ces deux référentiels évoluent indépendamment l'un de l'autre. Il y a donc un enjeu industriel important pour la prise en compte de la variabilité dans les exigences réglementaires et l'adoption d'une approche de ligne de produits pour le développement et la certification de tels systèmes.

La deuxième étude de cas concerne les descriptions informelles de produits publiées sur des sites Web à vocation commerciale. Les sites commerciaux ou d'information sur les produits proposent des descriptions de leur produits, en décrivent les avantages et les caractéristiques techniques. Si la gamme de produits décrits est étonnamment large, la description de ces produits manque d'une structure cohérente et systématique, ainsi que de contraintes de rédaction en langage naturel qui permettrait une description à la fois précise et homogène sur tout un ensemble de produits d'une même famille. Par conséquent, la description des produits peut comprendre des omissions ou des ambiguités sur les features, et l'enjeu est de pouvoir réconcilier toutes les informations issues de produits d'une même famille en un modèle cohérent, plus propice à l'analyse par un expert.

Problématique

L'une des étapes les plus critiques dans l'analyse du domaine est l'identification des éléments variables et communs d'une ligne de produits. La construction du modèle de variabilité à partir de documents informels est une activité difficile et complexe qui dépend principalement de l'expérience et l'expertise des ingénieurs du domaine. Le but de cette thèse consiste à:

- proposer une formalisation de la variabilité afin d'avoir une vue globale, homogène et complète de cette connaissance;
- adopter des techniques de traitement automatique du langage naturel permettant d'extraire et modéliser la variabilité à partir des documents informels, ambigus et

hétérogènes;
- assurer une traçabilité de la variabilité pour une meilleure compréhension et maintenance.

Dans la première étude de cas, l'hétérogénéité dans les exigences de sûreté est la cause directe des difficultés rencontrées par EDF et Areva dans la certification du projet EPR (Evolutionary Pressurized Reactor) dans les différents pays où ce dernier a été proposé (construction en Finlande, France et Chine, certification en cours aux Etats-Unis et en Grande-Bretagne). Ainsi, depuis 2008, sur les cinq projets d'EPR les plus avancés EDF et Areva en sont désormais à quatre architectures différentes pour le contrôle-commande et à cinq processus de certification propres à chaque pays [SB12a, dlVPW13]. Proposer un système de contrôle commande dans diffèrent pays pose un grand problème de variabilité qui concerne non seulement les règlementations mais aussi l'architecture elle-même. Dans cette étude de cas, l'objectif consiste à (1) extraire et formaliser la variabilité dans les exigences réglementaires, (2) automatiser la construction de modèles de features à partir des exigences réglementaires, (3) tracer la variabilité à différents niveaux d'abstraction afin d'étudier la conformité de l'architecture par rapport aux exigences.

Dans la deuxième étude de cas, les descriptions de produits contiennent une grande quantité d'information informelles à rassembler, analyser, comparer, et structurer. Cependant, une revue cas par cas de chaque description de produit demande un travail intense, beaucoup de temps et il devient impossible quand le nombre de produits à comparer augmente du fait de l'explosion combinatoire que cette augmentation engendre. Le plus grand défi est lié au nombre de produits et au nombre de features à rassembler et organiser. Plus il y a d'actifs et de produits, plus l'analyse sera difficile.

Etant donnée un ensemble de descriptions textuelles de produits, le but est de synthétiser automatiquement une matrice de comparaison de produits (PCM). Le principal défi est de pouvoir automatiser l'extraction de la variabilité à partir du texte informel et non structuré. Dans ce contexte, on s'intéresse à (1) automatiser l'extraction de PCM à partir des descriptions informelles de produits, (2) étudier la complémentarité entre les descriptions de produits et des spécifications techniques, et (3) assurer la traçabilité des PCMs avec les descriptions originales et les spécifications techniques pour plus de raffinement et maintenance par l'utilisateur.

Contributions

La contribution générale de cette thèse consiste à **appliquer des techniques automatiques pour extraire des connaissances relatives à la variabilité à partir du texte**. Pour se faire, il est nécessaire d'identifier les features, les points communs, les différences et les dépendances entre features. Nous étudions l'applicabilité de cette idée par l'instancier dans deux contextes différents qui présentent différentes caractéristiques à la fois en termes du degré de formalisme du langage et d'homogénéité de contenus. La section suivante introduit les principales contributions dans chaque contexte.

Etude de Cas 1: la rétro-ingénierie des modèles de features à partir des exigences réglementaires dans le domaine nucléaire.

Nous proposons une approche pour extraire la variabilité à partir des exigences réglementaires et maintenir la traçabilité de la variabilité à différents niveaux d'abstraction afin de dériver une architecture conforme aux exigences.

Dans cette étude de cas, la contribution principale est **une approche (semi) automatique pour la rétro-ingénierie des modèles de features à partir des exigences réglementaires**. Pour cette raison, nous exploitons des techniques du traitement automatique du langage naturel et d'exploration de données pour (1) extraire les features en se basant sur l'analyse sémantique et le regroupement (clustering) des exigences, (2) identifier les dépendances entre features en utilisant les règles d'association. Ces dépendances incluent les dépendances structurelles pour construire la hiérarchie (les relations parent-enfant, les relations obligatoires et optionnelles) et les dépendances transversales (les relations d'implication et d'exclusion). Cette méthode vise à assister les experts du domaine lors de la construction des modèles de features à partir des exigences. L'évaluation de cette approche montre que 69% de clusters sont corrects sans aucune intervention de l'utilisateur. Les dépendances structurelles montrent une capacité prédictive élevée: 95% des relations obligatoires et 60% des relations optionnelles sont identifiées. Egalement, la totalité des relations d'implication et d'exclusion sont extraites.

Pour résoudre le problème de variabilité dans l'industrie nucléaire, nous proposons d'abord **une formalisation de la variabilité dans les réglementations**. Nous choisissons pour cela d'utiliser le langage CVL (Common Variability Language) [Com], car il s'agit d'un langage indépendant du domaine. CVL ne demande aucun changement des artéfacts de développement, il n'introduit pas de complexité supplémentaire au niveau des artéfacts originaux, et peut être utilisé en conjonction avec différents artéfacts. En effet, CVL spécifie la variabilité dans des modèles séparés (modèles de features) qui sont liés à des artefacts de développement.

Les domaines et sujets couverts par les exigences de sûreté sont nombreux et larges. Pour réduire l'espace de recherche, nous utilisons la notion de topic (ou thème). L'idée est de modéliser la variabilité dans les règlementations par topic, dans différents corpus (dans différents pays), et considérant un même niveau d'abstraction (standard, document règlementaires, guides, pratique, etc.). D'autre part, notre approche fournit un moyen pratique pour maintenir **la traçabilité de la variabilité entre l'espace des problèmes** (exigences appartenant aux thèmes inhérents à la sûreté de fonctionnement tels que les défaillances de causes communes, la séparation et l'isolation des systèmes, la communication entre systèmes classés) **et l'espace solution** (l'architecture du système de contrôle-commande) pour étudier la robustesse de l'architecture dérivée par rapport à la variabilité dans les exigences.

Etude de Cas 2: L'extraction des matrices de comparaison de produits à partir des descriptions informelles.

Dans cette étude de cas, notre principale contribution est **une approche automatique pour synthétiser des matrices de comparaison de produits (PCMs) à partir de descriptions textuelles non structurées**. Nous exploitons des techniques automatiques pour extraire des PCMs en dépit de l'informalité et de l'absence de structure dans les descriptions de produits. Au lieu d'une revue au cas par cas de chaque description de produit, notre but est de fournir à l'utilisateur une vue compacte, synthétique et structurée d'une ligne de produits à travers une matrice (produit x feature).

Notre approche repose sur (1) la technologie d'analyse contrastive pour identifier les termes spécifiques au domaine à partir du texte, (2) l'extraction des informations pour chaque produit, (3) le regroupement des termes et le regroupement des informations. Notre étude empirique montre que les PCMs obtenus contiennent de nombreuses informations quantitatives qui permettent leur comparaison : 12.5% de features quantifiées et 15.6% des features descriptives avec seulement 13% de cellules vides. L'expérience utilisateur montre des résultats prometteurs et que notre méthode automatique est capable d'identifier 43% de features correctes et 68% de valeurs correctes dans des descriptions totalement informelles et ce, sans aucune intervention de l'utilisateur.

D'autre part, nous étudions l'aspect complémentaire entre les descriptions des produits et leur spécification technique. Le but est d'analyser les relations qui peuvent exister entre ces deux artéfacts. En effet, nous générons automatiquement des PCMs à partir des descriptions de produits (à l'aide de notre outil) puis calculons des PCMs à partir des spécifications techniques afin de trouver le chevauchement entre ces deux types de PCM. Notre étude utilisateur montre que concernant une grande partie des features (56%) et des valeurs (71%), nous avons autant ou plus d'informations dans la première catégorie des PCMs générées automatiquement avec notre outil. Nous montrons qu'il existe un potentiel pour compléter ou même raffiner les caractéristiques techniques des produits.

Nous avons implémenté notre approche dans un outil, *MatrixMiner*, qui est un environnement Web avec un support interactif non seulement pour synthétiser automatiquement des PCMs à partir des descriptions textuelles des produits, mais il est aussi dédié à la visualisation et l'édition des PCMs. Les résultats de l'évaluation suggèrent en effet que l'automatisation présente un grand potentiel, mais aussi certaines limites. L'intervention humaine est bénéfique et reste nécessaire pour (1) raffiner/corriger certaines valeurs (2) réorganiser la matrice pour améliorer la lisibilité du PCM. Pour cette raison, *MatrixMiner* offre également la possibilité de tracer les produits, les features et les valeurs d'un PCM avec les descriptions de produits originaux et les spécifications techniques. Les utilisateurs peuvent ainsi comprendre, contrôler et raffiner les informations dans les PCMs synthétisés en se référant aux descriptions et spécifications de produits.

La principale leçon à tirer de ces deux études de cas, est que l'extraction et l'exploitation de la connaissance relative à la variabilité dépendent du contexte, de la nature de la

variabilité et de la nature du texte. En particulier, le formalisme pour exprimer la variabilité dépend du contexte. Les modèles de features, pour extraire la variabilité à partir d'exigences réglementaires, facilitent la traçabilité de la variabilité à différents niveaux d'abstraction (les exigences et l'architecture). Pour la comparaison de produits divers, le PCM offre une vue claire et plus facile des produits d'une même famille. Il permet à l'utilisateur d'identifier immédiatement les features récurrentes et comprendre les différences entre les produits.

De même, les techniques du traitement automatique du langage naturel et d'exploration de données employées lors de l'extraction de la variabilité dépendent de la nature du texte et du formalisme qui a été considéré. En effet, lors de la construction d'un modèle de features, nous devons adopter des techniques capables de capturer les features et leurs dépendances: les dépendances structurelles (les relations parent-enfant, les relations obligatoires et optionnelles) pour construire la hiérarchie et les dépendances transversales (les relations d'implication et d'exclusion). Cependant, lors de la construction d'un PCM, nous avons besoin d'appliquer des techniques capables d'identifier les features pertinentes et leurs valeurs (booléennes, numériques ou descriptives) à partir du texte.

Contents

Contents **ix**

Introduction **1**

I Background and State of the Art **7**

1 Background **9**
1.1 Product Line Engineering 9
1.2 Variability Management 11
1.2.1 Variability 12
1.2.2 Variability Modeling 12
1.3 Feature Models 13
1.4 Product Comparison Matrices 16
1.5 Requirements Engineering and Regulations 19
1.5.1 Requirements Engineering 19
1.5.2 Requirements Engineering and Compliance with Regulations . . 22
1.6 Conclusion 24

2 State of the Art **25**
2.1 Statistical Techniques to Construct Variability Models 25
2.1.1 Text Mining 25
2.1.2 Clustering 27
2.1.3 Association Rules 28
2.2 Mining Knowledge using Terminology and Information Extraction 29
2.2.1 Terminology Extraction 29
2.2.2 Information Extraction 30
2.3 Approaches for Mining Features and Constraints 33
2.4 Approaches for Feature Models Synthesis 34
2.4.1 Synthesis of FMs from configurations/dependencies 34
2.4.2 Synthesis of FMs from product descriptions 35
2.4.3 Synthesis of FMs from requirements 36
2.5 Discussion and Synthesis 39
2.6 Conclusion 40

II Contributions **41**

3 Moving Toward Product Line Engineering in a Nuclear Industry Consortium **43**
3.1 Context . . . 44
3.2 Motivation and Challenges . . . 49
3.2.1 On the Regulation Heterogeneity and Variability . . . 49
3.2.2 Lack of Product Line Culture . . . 50
3.3 Common Variability Language (CVL) . . . 51
3.4 Information Retrieval and Data Mining Techniques . . . 51
3.5 Overview of the General Approach . . . 54
3.6 Handling Variability in Regulatory Requirements . . . 55
3.7 Automating the Construction of Feature Model from Regulations . . . 60
3.7.1 Overview of Feature Model Synthesis . . . 60
3.7.2 Requirements Relationship Graph . . . 61
3.7.3 Requirements Clustering . . . 62
3.7.4 Build the Product Model Hierarchy . . . 63
3.7.5 Variability Modeling . . . 63
3.8 Traceability between Requirements and Architecture . . . 65
3.8.1 Modeling Variability in Design Rules . . . 65
3.8.2 Mapping Between the Standards FM and the Design Rules FM . 67
3.8.3 Mapping Between the Design Rules FM and the I&C Architecture 67
3.9 Implementation . . . 68
3.10 Case Study and Evaluation Settings . . . 71
3.11 Evaluation . . . 72
3.12 Lessons Learned and Discussion . . . 75
3.13 Threats to Validity . . . 77
3.14 Conclusion . . . 78

4 Automated Extraction of Product Comparison Matrices From Informal Product Descriptions **79**
4.1 Context . . . 79
4.1.1 Toward Automatic Extraction of PCMs . . . 82
4.1.2 The Complementarity Aspect of Product Overviews and Technical Specifications . . . 83
4.2 Overview of the Automatic Extraction . . . 84
4.3 Terms & Information Extraction . . . 86
4.3.1 Terms Mining . . . 86
4.3.2 Contrastive Analysis . . . 89
4.3.3 Information Extraction . . . 90
4.4 Building the PCM . . . 91
4.4.1 Terms and Information Similarity . . . 92
4.4.2 Terms and Information Clustering . . . 92
4.4.3 Extracting features and Cell Values . . . 92

4.5 Tool Support . . . 93
4.5.1 Implementation and Used Technologies . . . 93
4.5.2 Importing, Visualizing, and Editing . . . 95
4.6 Case Study and Evaluation Settings . . . 96
4.6.1 Data . . . 96
4.6.2 Threshold Settings . . . 96
4.6.3 Research Questions . . . 97
4.7 Empirical Study . . . 98
4.7.1 Dataset . . . 98
4.7.2 RQ1.1. Properties of the resulting PCMs extracted from textual overviews . . . 99
4.7.3 RQ1.2. Impact of Selected Products on the Synthesized PCMs . 102
4.7.4 RQ1.3. Complementarity of Overview PCM and Specification PCM 104
4.8 User Study . . . 106
4.8.1 Experiment Settings . . . 106
4.8.2 RQ2.1. How effective are the techniques to fully extract a PCM from a user point of view? . . . 110
4.8.3 RQ2.2. How effective is the overlap between overview PCMs and specification PCMs? . . . 110
4.9 Threats to Validity . . . 112
4.10 Conclusion . . . 113

5 Two Case Studies: Comparison, Lessons Learned and Discussion 115
5.1 Input Documentation . . . 115
5.2 Variability Models . . . 116
5.3 NLP and Data Mining Techniques for Mining Variability . . . 117
5.4 Traceability . . . 119
5.5 Conclusion . . . 120

III Conclusion and Perspectives 121

6 Conclusion and Perspectives 123
6.1 Conclusion . . . 123
6.2 Perspectives . . . 125

Bibliography 131

List of Figures 161

List of Tables 163

Introduction

Context

A Product Line (PL) is a group of closely related products that together address a particular market segment or fulfil a particular mission. In product line engineering, domain analysis aims to identify and organize features that are common or vary within a domain [PBvdL05a]. A feature can be roughly defined as a prominent and distinctive user visible characteristic of a product. Domain analysis is generally carried out by experts on the basis of existing informal documentation. Yet, the construction process of variability models may prove very arduous for stakeholders, especially when they take unstructured artifacts as inputs. Indeed, they have to deal with a huge amount of scattered and informal data to collect, review, compare and formalize. This can be an arduous task when performed manually, and can be error-prone in the presence of a change in requirements.

Numerous approaches have been proposed to mine variability and support domain analysis [ACP+12a, YZZ+12, DDH+13a, LHLG+14, BABN15]. However, few of them pay attention to adopt automated techniques for the construction of variability models from unstructured and ambiguous documents. Natural Language Processing (NLP) and data mining techniques have been used by researchers to support a number of activities such as information retrieval, terminology extraction, clustering, association rule learning, etc., [T+99, RS10, KU96, PY13, AIS93]. In this thesis, our main challenge is to adopt and exploit these techniques to address mining and modeling variability from informal documentation in different contexts. In particular, we aim to identify features, commonalities, differences and features dependencies among related products. Indeed, the NLP techniques employed when mining variability depend on the nature of text and the formalism which has been considered.

We investigate the applicability of this idea by instantiating it in two different contexts: (1) reverse engineering *Feature Models (FMs)* from regulatory requirements in nuclear domain and (2) synthesizing *Product Comparison Matrices (PCMs)* from informal product descriptions. FMs and PCMs are fundamental formalisms for specifying and reasoning about *commonality* (i.e., the common characteristics of products) and *variability* (i.e., the differences between products) of a set of related products.

The first case study handles regulatory requirements for safety systems certification in nuclear domain. The regulatory requirements are provided in large and heterogeneous documents: regulatory documents, guides, standards and even tacit knowledge acquired

from anterior projects in the past. These regulations are most often disconnected from the technical system requirements, which capture the expected system behavior. In many cases, regulatory documents provide very high level and ambiguous requirements that leave a large margin for interpretation. Worse, regulation changes over time and from one country to another. In Europe, nuclear actors mainly follow the IEC/IAEA corpus whereas in the US, IEEE/ISO standards are applied. These two corpora have been written independently from each other.

The second case study deals with publicly available product descriptions found in online product repositories and marketing websites. Informal product descriptions describe features including technical characteristics and benefits of products. Product descriptions lack of consistent and systematic structure to describe products, and constraints in writing these descriptions expressed in natural language.

Motivation and Challenges

One of the most critical steps in domain analysis is the identification of variable and common elements in the products that are to be supported. Deriving an accurate variability model from textual documents remains a hard and complex activity and still mostly relies on the experience and expertise of domain engineers. Our global challenges consist in:

- formalizing variability to keep a homogeneous, complete and global view of this knowledge;
- adopting effective automated NLP techniques capable of mining and modeling variability from informal, ambiguous and heterogeneous documentation;
- tracing variability to improve the understanding of system variability, as well as support its maintenance and evolution.

In the specific context of regulatory requirements, one applicant has to deal with very heterogeneous regulations and practices, varying from one country to another. This heterogeneity has a huge impact in the certification process as the regulators safety expectations, evidences and justification to provide can vary [SB12a, dlVPW13]. At this level, the main concern comes from the difference between national practices and the set of documents (regulatory texts and standards) to comply with. The nuclear industry has an unstable and growing set of safety standards. Worse, the set of safety standards is increasing within two main standards areas.

Performing the same safety function in different countries then leads to a huge problem of variability that concerns, not only the set of requirements to comply with and the certification process, but also the system's architecture itself. The major challenge is the conformance of safety systems to multiple different regulations. In this case study, we aim to (1) extract and formalize the variability in regulatory requirements, (2) automate the construction of feature models from regulatory requirements, (3) address tracing variability between artifacts across problem and solution space to investigate the robustness of the derived architecture against regulations variability.

In the second case study, product descriptions contain a huge amount of scattered and informal data to collect, review, compare, and structure. Numerous organizations or individuals rely on these textual descriptions for analyzing a domain and a set of related products. Analyzing manually a group of related products is notoriously hard [HCHM+13a, DDH+13a]. A case-by-case review of each product description is labor-intensive, time-consuming, and quickly becomes impractical as the number of considered products grows. The biggest challenge is related to the number of products and the number of features an analyst has to gather and organize. The more assets and products, the harder the analysis.

Our goal is to automate the manual task of analyzing each product with respect to its textual description and clustering information over several products, and provides a reader with an accurate and synthetic PCM – i.e., tabular data that describe products along different features [BSA+14]. In this case study, our goal is to (1) automate the extraction of PCMs from informal descriptions of products, (2) investigate the complementarity between products descriptions and technical specifications, and (3) maintain traceability of PCMs with the original descriptions and the technical specifications for further refinement or maintenance by users.

Contributions

In this thesis, **our general contribution is to address mining and modeling variability from informal documentation using NLP and data mining techniques**. To do so, it is necessary to identify features, commonalities, differences and features dependencies among the related products. We investigate the applicability of this idea by instantiating it in the two different case studies. In this section, we summarize our main contributions in each context.

Case Study 1: Reverse Engineering Feature Models from Regulatory Requirements in Nuclear Domain

We propose an approach to extract variability from safety requirements as well as mapping variable requirements and variable architecture elements to derive a complying architecture. This complex task requires a comprehensive and in-depth analysis of regulations and the architecture for safety systems in nuclear power plants.

In this case study, our core contribution is **a (semi)automated approach to reverse engineering feature models from regulatory requirements**. We adopt NLP and data mining techniques to (1) extract features based on semantic analysis and requirements clustering and (2) identify features dependencies using association rules. These dependencies include structural dependencies to build the hierarchy (parent-child relationships, mandatory and optional relationships) and transversal dependencies (requires and exclude relationships). This automated method assists experts when constructing feature models from these regulations. The evaluation shows that our approach is able to retrieve 69% of correct clusters without any user intervention. We notice that

structural dependencies show a high predictive capacity: 95% of the mandatory relationships and 60% of optional relationships are found. We also observe that the totality of requires and exclude relationships are extracted.

To tackle the variability issue in unaware nuclear industry, we propose before **a formalization of variability in regulations**. We choose to rely on the Common Variability Language (CVL) [Com] since it is a domain independent language for specifying and resolving variability. CVL does not require changing the complexity of the development artifacts and can be used in conjunction with different development artifacts. Indeed, CVL promotes specifying variability in separate models (feature models) which are linked to the development artifacts. To narrow the problem space, the idea is to analyze variability in regulatory documents by topic on different corpora (*i.e.* in different countries) and on the same abstraction level. On the other hand, our approach provides **tracing variability across problem and solution space** to investigate the robustness of the derived architecture against regulations variability.

Case Study 2: Synthesizing Product Comparison Matrices from Informal Product Descriptions

In this case study, our main contribution is **an approach to automate the extraction of product comparison matrices from informal descriptions of products**. We investigate the use of automated techniques for synthesizing a PCM despite the informality and absence of structure in the textual descriptions. Instead of reading and confronting the information of products case-by-case, our purpose is to deliver a compact, synthetic, and structured view of a product line - a PCM.

Our proposed approach relies on contrastive analysis technology to mine *domain specific terms* from text, information extraction, terms clustering and information clustering. Overall, our empirical study shows that the resulting PCMs exhibit numerous quantitative and comparable information: 12.5% of quantified features, 15.6% of descriptive features and only 13% of empty cells. The user study shows that our automatic approach retrieves 43% of correct features and 68% of correct values in one step and without any user intervention.

On the other hand, we investigate the complementarity aspect between products descriptions and technical specifications. The purpose here is to analyze the nature of relationship that may exist between these two artifacts. Indeed, we need to synthesize PCMs from product descriptions and compute PCMs from technical specifications in order to calculate the overlap between these two kinds of PCMs. Our user study shows that regarding a significant portion of features (56%) and values (71%), we have as much or more information in the generated PCMs than in the specifications. We show that there is a potential to complement or even refine technical information of products.

The evaluation insights drive the design of the *MatrixMiner* which is a web environment with an interactive support not only for automatically synthesizing PCMs from textual descriptions of products, but also is dedicated to the visualization and edition of PCMs. The results indeed suggest that automation has a great potential but also some limitations. Human intervention is beneficial to (1) refine/correct some values

(2) reorganize the matrix for improving readability of the PCM. For this reason, *MatrixMiner* also provides the ability to tracing products, features and values of a PCM to the original product descriptions and technical specifications. Likewise users can understand, control and refine the information of the synthesized PCMs within the context of product descriptions and specifications.

The main lesson learnt from the two case studies is that the exploitability and the extraction of variability knowledge depend on the context, the nature of variability and the nature of text. In particular, the formalism to express variability depends on the context: feature models to capture variability in regulatory requirements, it is easier to address variability-aware bridging of the two levels of abstraction (requirements and architecture); Meanwhile, when comparing products on the web, PCMs offer a clear *product line view* to practitioners. It is then immediate to identify recurrent features and understand the differences between products.

Similarly, the NLP and data mining techniques employed when mining variability depend on the nature of text and the formalism which has been considered. Indeed, when building a feature model, we need to adopt techniques capable of extracting features and their dependencies: structural dependencies (parent-child relationships, mandatory and optional relationships) to build the hierarchy and transversal dependencies (requires and exclude relationships). But when constructing a PCM, we need to apply techniques able to mine relevant features and their values (boolean, numerical or descriptive) from the text.

Plan

The remainder of this thesis is organized as follows.

Chapter 1 gives a background about product line engineering, variability modeling and requirements engineering. The variability models are presented briefly through a classification based on the main variability concepts.

Chapter 2 presents the state of the art regarding our approach. This chapter provides a survey of the most used statistical techniques to perform the construction of variability models, and NLP techniques for terminology and information extraction. We also explain and compare methods to extract features and synthesize feature models from different artifacts.

Chapter 3 instantiates our global contribution in the first case study to reverse engineering feature models from regulatory requirements in the nuclear domain. In this chapter, we formalize the variability in safety requirements, propose an approach to automatically synthesize feature models from these regulations and establish tracing variability with the architecture.

Chapter 4 instantiates our general contribution in the second case study to synthesize product comparison matrices from informal product descriptions. In this chapter we propose an approach to automate the extraction of PCMs from unstructured descriptions written in natural language, investigate the complementarity aspect between products descriptions and technical specifications and implement our approach in a tool,

MatrixMiner.

Chapter 5 provides a comparison, lessons learned and discussion regarding these two case studies. We characterize in each context the nature of input text, the variability model including the used formalism and its exploitation, the adopted techniques and finally how variability tracing could be applied in practice.

Chapter 6 draws conclusions and identifies future work and perspectives for variability management.

Publications

- Sana Ben Nasr, Guillaume Bécan, Mathieu Acher, João Bosco Ferreira Filho, Benoit Baudry, Nicolas Sannier, and Jean-Marc Davril. MatrixMiner: A Red Pill to Architect Informal Product Descriptions in the Matrix. In *ESEC/FSE'15*, Bergamo, Italy, August 2015.

- Sana Ben Nasr, Nicolas Sannier, Mathieu Acher, and Benoit Baudry. Moving Toward Product Line Engineering in a Nuclear Industry Consortium. In *18th International Software Product Line Conference (SPLC'2014)*, Florence, Italy, September 2014.

- Guillaume Bécan, Mathieu Acher, Benoit Baudry, and Sana Ben Nasr. Breathing ontological knowledge into feature model synthesis: an empirical study. In *Empirical Software Engineering (ESE)* published by Springer, 2015.

- Guillaume Bécan, Sana Ben Nasr, Mathieu Acher, and Benoit Baudry. WebFML :Synthesizing Feature Models Everywhere. In *SPLC'2014*, Florence, Italy, September 2014.

- Nicolas Sannier, Guillaume Bécan, Mathieu Acher, Sana Ben Nasr, and Benoit Baudry. Comparing or Configuring Products : Are We Getting the Right Ones ? In *8th International Workshop on Variability Modelling of Software-intensive Systems*, Nice, France, January 2014. ACM.

- Nicolas Sannier, Guillaume Bécan, Sana Ben Nasr, and Benoit Baudry. On Product Comparison Matrices and Variability Models from a Product Comparison/-Configuration Perspective. In *Journée lignes de produits - 2013*, Paris, France, November 2013.

Under Review

Sana Ben Nasr, Guillaume Bécan, Mathieu Acher, Nicolas Sannier, João Bosco Ferreira Filho, Benoit Baudry and Jean-Marc Davril. Automated Extraction of Product Comparison Matrices From Informal Product Descriptions. *Journal of Systems and Software*.

Part I

Background and State of the Art

Chapter 1

Background

In this chapter, we discuss different domains and concepts applied in our proposal, including Product Line Engineering, Variability Modeling and Requirements Engineering. The objective of this chapter is to give a brief introduction to these concerns, used throughout the thesis. This introduction aims at providing a better understanding of the background and context in which our work takes place, as well as the terminology and concepts presented in the next chapters.

The chapter is structured as follows. In Section 1.1, we present the main concepts of product line engineering. Section 1.2 describes briefly some approaches dealing with variability modeling. Section 1.3 describes the essential principles and semantic foundation of feature models. Section 1.4 introduces product comparison matrices. Section 1.5 explains the basics of requirements engineering and deals with two aspects of particular interest: regulatory requirements and compliance with these latters.

1.1 Product Line Engineering

Product line engineering is a viable and important reuse based development paradigm that allows companies to realize improvements in time to market, cost, productivity, quality, and flexibility [CN02]. According to Clements & Northrop [CN02] product line engineering is different from single-system development with reuse in two aspects. First, developing a family of products requires "choices and options that are optimized from the beginning and not just one that evolves over time". Second, product lines imply a preplanned reuse strategy that applies across the entire set of products rather than ad-hoc or opportunistic reuse. The product line strategy has been successfully used in many different industry sectors, and in particular, in software development companies [PBvdL05b] [KCH+90] [W+99].

Software Product Lines (SPL) engineering is a rapidly emerging software engineering paradigm to develop software applications (software-intensive systems and software products) using platforms and mass customization [PBvdL05b].
The traditional focus of software engineering is to develop single software, i.e., one software system at a time. A typical development process begins with the analysis of

customers' requirements and then several development steps are performed (specification, design, implementation, testing). The result obtained is a single software product. In contrast, SPL engineering focuses on the development of multiple similar software systems from common core assets [CN02] [PBvdL05b].

Software product line engineering relies on the concept of *mass customization*, which is a large-scale production of goods tailored to individual customer's need [Dav97]. SPL engineering aims at developing related variants in a systematic way and providing appropriate solutions for different customers [CN02]. Instead of individually developing each variant from scratch, commonalities are considered only once.

Definition 1.1 (Software Product Line) *"A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way" [CN02].*

Software product line engineering thus focuses on the production and maintenance of multiple similar software products by reusing common software artifacts, or assets in the context of software product lines.

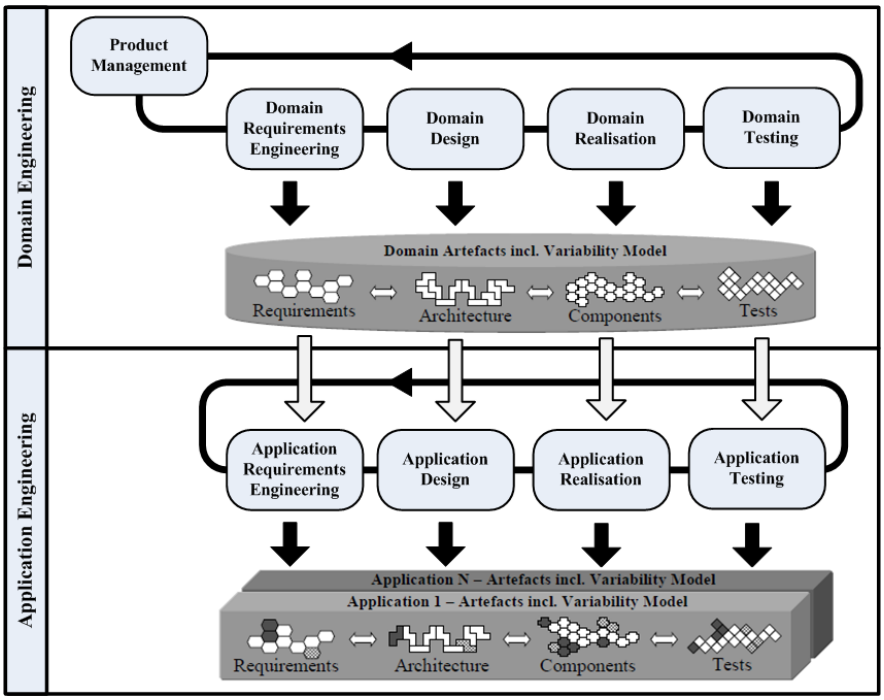


Figure 1.1: The product line engineering framework [PBvdL05b]

Product line engineering is separated in two complementary phases: domain engineering and application engineering. Domain engineering is concerned with development for reuse while application engineering is the development with reuse [W+99] [PBvdL05b].

In other words, the domain engineering process is responsible for creating reusable assets, while application engineering is the process of reusing those assets to build individual but similar products. Both the domain engineering as well as the application engineering are complementary processes and do not follow a specific order. For instance, it is possible to create assets from already developed products, in which case, assets are built from the artifacts that constitute the products. Otherwise, artifacts are built from scratch in order to be reused in several products. The idea behind this approach to product line engineering is that the investments required to develop the reusable artifacts during domain engineering, are outweighed by the benefits of deriving the individual products during application engineering [DSB04] [DSB05].

Domain engineering. The process to develop a set of related products instead of a single product is called domain engineering. It is the process to identify what differs between products as well as reusable artifacts, to plan their development. It thus defines the *scope* of the product line. In particular, the domain analysis phase is responsible for identifying and describing the common artifacts and those that are specific for particular products. This is the development *for reuse* process, made easier by traceability links between those artifacts [PBvdL05b]. In the domain realization phase, each artifact is modeled, planned, implemented and tested as reusable components.

Application engineering. Application engineering is the development process *with reuse*. It is the process of combining common and reusable assets obtained during the domain engineering process. Applications are thus built by reusing those artifacts and exploiting the product line. During the application requirements phase, a product configuration is defined, that fits those requirements. Then, the final product is built during a product derivation process, which is part of the application realization phase.

Product configuration: this process refers to the selection or deselection of a set of reusable artifacts identified in the domain engineering process. This selection is usually done relying on a *variability model*, which describes the commonalities and differences between potential products at an higher abstraction level.

Product derivation: once a configuration is defined through the variability model, the related artifacts are given as input to the product derivation process, which in return yields the final product. This process can be manual or automated, and differs among product lines.

1.2 Variability Management

Central and unique to product line engineering is the management of *variability*, i.e., the process of factoring out common and variable artifacts of the product line. Managing variability is the key, cross-cutting concern in product line engineering [CN02, PBvdL05b, CBK13, MP14]. It is also considered as one of the key feature that distinguishes SPL engineering from other software development approaches or traditional software reuse approaches [BFG+02]. Product line variability describes the variation among the products of a product line in terms of properties, such as features. Many

definitions of feature have been proposed in the product line literature.

Definition 1.2 (Feature) *"a prominent or distinctive user-visible aspect, quality or characteristic of a software system or systems" [KCH+90], "a product characteristic from user or customer views, which essentially consists of a cohesive set of individual requirements" [CZZM05] or "end-user visible functionality of the system" [CE00]*

1.2.1 Variability

Several definitions of variability have been given in the literature.

Variability in Time vs. Variability in Space. Existing work on software variation management can be generally split into two categories. The variability in time and the variability in space are usually considered as fundamentally distinct dimensions in SPL engineering. Pohl *et al.* define the variability in time as *"the existence of different versions of an artifact that are valid at different times"* and the variability in space as *"the existence of an artifact in different shapes at the same time"* [PBvdL05b]. Variability in time is primarily concerned with managing program variation over time and includes revision control system and the larger field of software configuration management. The goal of SPL engineering is mainly to deal with variability in space [Erw10, EW11].

Commonality and Variability. Weiss and Lai define variability in SPL as *"an assumption about how members of a family may differ from each other"* [W+99]. Hence variability specifies the particularities of a system corresponding to the specific expectations of a customer while commonality specifies assumptions that are true for each member of the SPL. [SVGB05] adopt a software perspective and define variability as the *"the ability of a software system or artifact to be efficiently extended, changed, customized or configured for use in a particular context"*. At present, these two definitions are sufficient to capture the notion of variability: the former definition is more related to the notions of domain and commonality while the later focuses more on the idea of customization. Nevertheless, there is no one unique perception or definition of variability: [BB01] propose different categories of variabilities, [SVGB05] have defined five levels of variability while some authors distinguish essential and technical variability [HP03], external and internal variability [PBvdL05b], product line and software variability [MPH+07].

1.2.2 Variability Modeling

As managing variability is a key factor, it must be expressed using a dedicated support. Product line variability is thus documented in so-called variability models. Chen *et al.* [CABA09] provide an overview of various approaches dealing with variability modeling.

Feature modeling is by far the most widespread notation in software product line engineering, offering a simple and effective way to represent variabilities and commonalities in a product family. A feature is defined as a *"prominent or distinctive user-visible aspect, quality, or characteristic of a software system or system"* [KCH+90]. The modeling

approach enables the representation of variability and commonality early in the product life cycle, as a support for the domain analysis process.

Using feature models for variability modeling was first introduced back in 1990 by Kang *et al.*, as part of the Feature Oriented Domain Analysis (FODA) [KCH+90]. Many extensions and dialects of feature models have been proposed in literature (e.g., FORM [KKL+98], FeatureRSEB [GFA98], [Rie03]; [BPSP04], [CHE05]; [SHTB07], [AMS06, AMS07]). Thus, feature models are nowadays considered as the *de-facto* standard for representing variability. Djebbi and Salinesi [DS06] provided a comparative survey on four feature diagram languages for requirements variability modeling. The languages are compared according to a list of criteria that includes readability, simplicity and expressiveness, type distinction, documentation, dependencies, evolution, adaptability, scalability, support, unification, and standardizeability.

Decision modeling is one mean for variability modeling. A decision model is defined as *"a set of decisions that are adequate to distinguish among the members of an application engineering product family and to guide adaptation of application engineering work products"* [SRG11]. Decision-oriented approaches treat *decisions* as first-class citizens for modelling variability. DOPLER (Decision–Oriented Product Line Engineering for effective Reuse), introduced by Dhungana *et al.* [DGR11], is one of the most representative decision-oriented approaches. Schmid and John [SJ04], Forster *et al.* [FMP08], Dhungana *et al.* [DRGN07], amongst others, use decision models as variability modeling language.

Variability can be specified either as an integral part of the development artifacts or in a separate orthogonal variability model [PBvdL05b]. The former way commonly yield annotation-based approaches, in which the development artifacts are marked (annotated) introducing variability-related aspects. Examples of such methods are presented in [Gom06, ZJ06]. Another way of variability modeling is by mean of orthogonal variability models (OVM) [PBvdL05b]. In those models, the main concept is the one of variation points, which are an abstraction of software artifacts that represent variability. In the OVM only the variability of the product line is documented (independent of its realization in the various product line artifacts). The variability elements in an OVM are, in addition, related to the elements in the traditional conceptual models which "realize" the variability defined by the OVM. Another approach proposed to make variability models orthogonal to the product line models is the Common Variability Language (CVL).
As learned from Chen's survey, most of existing approaches in variability management can be classified (and classify themselves) as feature modeling ones [SRG11].

1.3 Feature Models

Feature Models (FMs) aim at characterizing the *valid combinations of features* (a.k.a. configurations) of a system under study. A feature *hierarchy*, typically a tree, is used to facilitate the organization and understanding of a potentially large number of concepts (features). Figure 1.2 gives a first visual representation of a feature model. Features

are graphically represented as rectangles while some graphical elements (e.g., unfilled circle) are used to describe the variability (e.g., a feature may be optional). Figure 1.2 depicts a simplified feature model inspired by the mobile phone industry. The model illustrates how features are used to specify and build software for mobile phones. The software loaded in the phone is determined by the features that it supports.

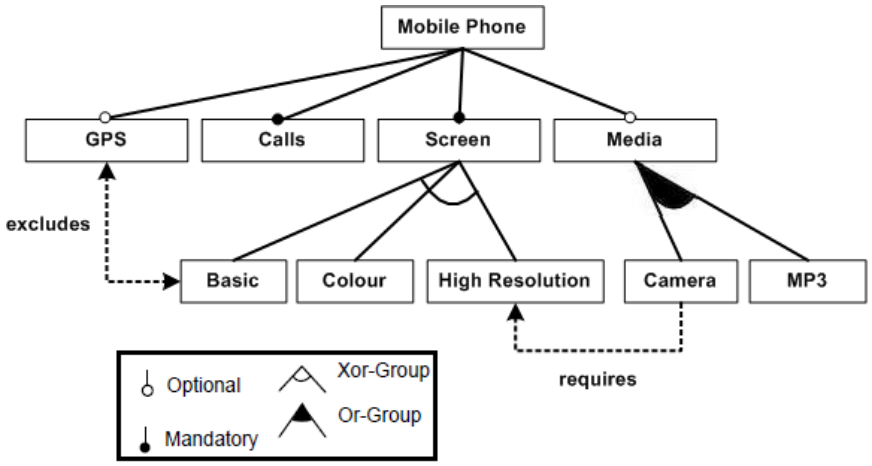


Figure 1.2: A family of mobile phones described with a feature model [BSRC10]

Syntax of Feature Models. Different syntactic constructions are offered to attach variability information to features organized in the hierarchy (see Definition 1.3). When decomposing a feature into subfeatures, the subfeatures may be *optional* or *mandatory*. According to the model, all phones must include support for calls. The feature Calls is mandatory. However, the software for mobile phones may optionally include support for GPS and multimedia devices. The features GPS and Media are optional. Note that a feature is mandatory or optional in regards to its parent feature (e.g., a feature may be modeled as a mandatory feature and not be necessary included in a configuration in the case its parent is not included in the configuration).

Features may also form *Or*–, or *Xor*–groups. Camera and MP3 form Or–group. In Figure 1.2, whenever Media is selected, Camera, MP3 or both can be selected. Features Basic, Colour and High resolution form an Xor–group, they are mutually exclusive. In the example, mobile phones may include support for a basic, colour or high resolution screen but only one of them.

Cross-tree *constraints* over features can be specified to restrict their valid combinations. Any kinds of constraints expressed in Boolean logic, including predefined forms of Boolean constraints (equals, requires, excludes), can be used. Mobile phones including a camera must include support for a high resolution screen: Camera requires High resolution. GPS and basic screen are incompatible features: GPS excludes Basic. We consider that a feature model is composed of a feature diagram plus a set of constraints expressed in propositional logic (see Definition 1.3).

Definition 1.3 (Feature Model) *A feature diagram is a 8-tuple* $\langle G, E_M, G_{MTX}, G_{XOR}, G_{OR}, EQ, RE, EX\rangle$*:* $G = (\mathcal{F}, E)$ *is a rooted tree where* $\mathcal{F}$ *is a finite set of features,* $E \subseteq \mathcal{F} \times \mathcal{F}$ *is a set of directed child–parent edges ;* $E_M \subseteq E$ *is a set of edges that define mandatory features with their parents ;* $G_{MTX}, G_{XOR}, G_{OR} \subseteq 2^{\mathcal{F}}$ *are non-overlapping sets of edges participating in feature groups. EQ (resp. RE, EX) is a set of equals (resp. requires, excludes) constraints whose form is* $A \Leftrightarrow B$ *(resp.* $A \Rightarrow B$*,* $A \Rightarrow \neg B$*) with* $A \in \mathcal{F}$ *and* $B \in \mathcal{F}$*. The following well-formedness rule holds: a feature can have only one parent and can belong to only one feature group. A feature model is a pair* $\langle FD, \psi\rangle$ *where FD is a feature diagram, and* ψ *is a Boolean formula over* $\mathcal{F}$*.*

Semantics of Feature Model. The essence of an FM is its *configuration semantics* (see Definition 1.4). The syntactical constructs are used to restrict the combinations of features authorized by an FM. For example, at most one feature can be selected in a Mutex-group. As such Mutex-groups semantically differ from optional relations. Mutex-groups also semantically differ from Xor-groups. These latters require that at least one feature of the group is selected when the parent feature is selected. Formally, the cardinality of a feature group is a pair (i, j) (with $i \leq j$) and denotes that at least i and at most j of its k arguments are true. G_{MTX} (resp. G_{XOR}, G_{OR}) are sets of *Mutex-groups* (resp. *Xor-groups*, *Or-groups*) whose cardinality is $(0, 1)$ (resp. $(1, 1)$, $(1, m)$: m being the number of features in the Or-group). The configuration semantics can be specified via translation to Boolean logic [CW07a]. Table 1.1 shows the valid product configurations defined by the FM in Figure 1.2. In particular, the configuration semantics states that a feature cannot be selected without its parent, i.e., all features, except the root, logically imply their parent. As a consequence, the *feature hierarchy also contributes to the definition of the configuration semantics.*

Definition 1.4 (Configuration Semantics) *A configuration of a feature model* g *is defined as a set of selected features.* $[\![g]\!]$ *denotes the set of valid configurations of* g*.*

Another crucial and dual aspect of an FM is its *ontological semantics* (see Definition 1.5). Intuitively the ontological semantics of an FM defines the way features are conceptually related. Obviously, the feature hierarchy is part of the ontological definition. The parent–child relationships are typically used to *decompose* a concept into sub-concepts or to *specialize* a concept. There are also other kinds of implicit semantics of the parent-child relationships, *e.g.*, to denote that a feature is *"implemented by"* another feature [KLD02]. Looking at Figure 1.2, the concept of Mobile Phone is composed of different properties like Calls, Screens, or Media; Media can be either specialized as a Camera or an MP3, etc. Feature groups are part of the ontological semantics (see Definition 1.5) since there exists FMs with the same configuration semantics, the same hierarchy but having different groups [SLB+11a, ABH+13a].

Definition 1.5 (Ontological Semantics) *The hierarchy* $G = (\mathcal{F}, E)$ *and feature groups* $(G_{MTX}, G_{XOR}, G_{OR})$ *of a feature model define the semantics of features' relationships including their structural relationships and conceptual proximity.*

Table 1.1: Valid product configurations of mobile phone SPL

Products	Mobile Phone	Calls	Screen	Media	GPS	Basic	Colour	High Resolution	Camera	MP3
P1	✓	✓	✓			✓				
P2	✓	✓	✓				✓			
P3	✓	✓	✓					✓		
P4	✓	✓	✓		✓		✓			
P5	✓	✓	✓		✓			✓		
P6	✓	✓	✓	✓		✓				
P7	✓	✓	✓	✓		✓				✓
P8	✓	✓	✓	✓			✓			
P9	✓	✓	✓	✓			✓			✓
P10	✓	✓	✓	✓				✓		
P11	✓	✓	✓	✓				✓	✓	
P12	✓	✓	✓	✓				✓		✓
P13	✓	✓	✓	✓	✓		✓			
P14	✓	✓	✓	✓	✓		✓			✓
P15	✓	✓	✓	✓	✓			✓		
P16	✓	✓	✓	✓	✓			✓	✓	
P17	✓	✓	✓	✓	✓			✓		✓
P18	✓	✓	✓	✓				✓	✓	✓
P19	✓	✓	✓	✓	✓			✓	✓	✓

Some extensions of feature models have been proposed *e.g.*, feature attributes [BTRC05], cardinality–based feature models [CHE05, CK05]. Kang *et al.* use an example of attribute in [KCH+90] while Czarnecki *et al.* coin the term "feature attribute" in [CBUE02]. However, as reported in [BSRC10], the vast majority of research in feature modeling has focused on "basic" [CHKK06, CW07b], propositional feature models.

1.4 Product Comparison Matrices

Considerable research effort has been devoted to the study of spreadsheets [HG94, Pan08]. All studies have the same observation: errors in spreadsheet are common but non trivial [AE07, CVAS11, HPD12, HPvD12]. Automated techniques have been developed for locating errors; guidelines on how to create well-structured and maintainable spreadsheets have been established, etc. Herman *et al.* reported that the current state of spreadsheet use still leads to numerous problems [HPVD11]. Product Comparison Matrices (PCMs) can be seen as a special form of spreadsheets with specific characteristics and objectives (see Figure 1.3 for an example). A shared goal of this line of research is to improve the quality of spreadsheets (i.e., PCMs). Some works aim at tackling programming errors or code smells in spreadsheets [CFRS12]. General rules exposed in [CFRS12] can be implemented. Specific rules that apply to specific concepts of PCMs can also be considered. In both cases, the formalization of PCMs eases the realization.

As spreadsheets are subject to errors and ambiguity, some works propose to synthesize high-level abstractions or to infer some information [AE06, CE09, CES10]. For instance, Chambers and Erwig [CE09] describe a mechanism to infer dimensions (i.e., units of measures). These works typically operate over formulas of spreadsheets - a

Name	Navigation	Speaker	Screen size (in)	Screen type	Resolution (px)
Archos 105[1]	D-pad, 7 buttons	No	1.8	OLED	160 × 128
Archos 405[2][3]	D-pad, 6 two-way buttons	No	3.5	TFT LCD	320 × 240
Cowon Q5W[4]	Touchscreen	Yes	5	TFT LCD	800 × 480
Gigabeat T-Series[5]	D-pad, 4 buttons	No	2.4	TFT LCD	320 × 240
GP2X F-200[6]	8-way d-pad/touchscreen, 9 buttons	Yes	3.5	TFT LCD	320 × 240
iPod classic[7]	Click wheel, 1 center, 4 embedded buttons	Clicker only	2.5	TFT LCD	320 × 240
iPod nano 6G[8]	Multi-touch screen	No	1.54	TFT LCD	240 × 240
iPod touch[9]	Multi-touch screen, 3 buttons	Yes (ex. 1st gen.)	3.5	TFT LCD	480 × 320 (4th gen. 960 × 640)

Figure 1.3: PCM of Wikipedia about Portable Media Players

concept not apparent in PCM - or target general problems that are not necessarily relevant for PCMs. Some of the techniques could be reused or adapted. Another research direction is to elicitate the domain information stored in spreadsheets. For instance, Hermans *et al.* proposed to synthesize class diagrams based on the analysis of a spreadsheet [HPvD10].

Constructive approaches for ensuring that spreadsheets are correct by construction have been developed in order to prevent typical errors associated with spreadsheets. ClassSheet [EE05] introduces a formal object-oriented model which can be used to automatically synthesize spreadsheets. MDSheet is based on ClassSheet and relies on a bi-directional transformation framework in order to maintain spreadsheet models and their instances synchronized [CFMS12]. Francis *et al.* [FKMP13] develop tools to consider spreadsheets as first-class models and thus enable the reuse of state of the art model management support (e.g., for querying a model).

PCMs form a rich source of data for comparing a set of related and competing products over numerous features. A PCM can also be considered as a declarative representation of a feature model. Despite their apparent simplicity, PCMs contain heterogeneous, ambiguous, uncontrolled and partial information that hinders their efficient exploitations. Bécan *et al.* [BSA+14] proposed a metamodel that offers a more formal canvas for PCM edition and analysis. Figure 1.4 presents the PCM metamodel defined as an unifying canvas.

This metamodel describes both the structure and the semantic of the PCM domains. In this metamodel, PCMs are not individual matrices but a set of different matrices that contain cells. This happens when comparing a large set of products or features. In order to preserve readability, PCM writers can split the PCM content into several matrices. Cells can be of 3 types: *Header*, *ValuedCell*, and *Extra*. Header cells identify products or features.

In the metamodel, the structure of the PCM is not led by rows or columns but with

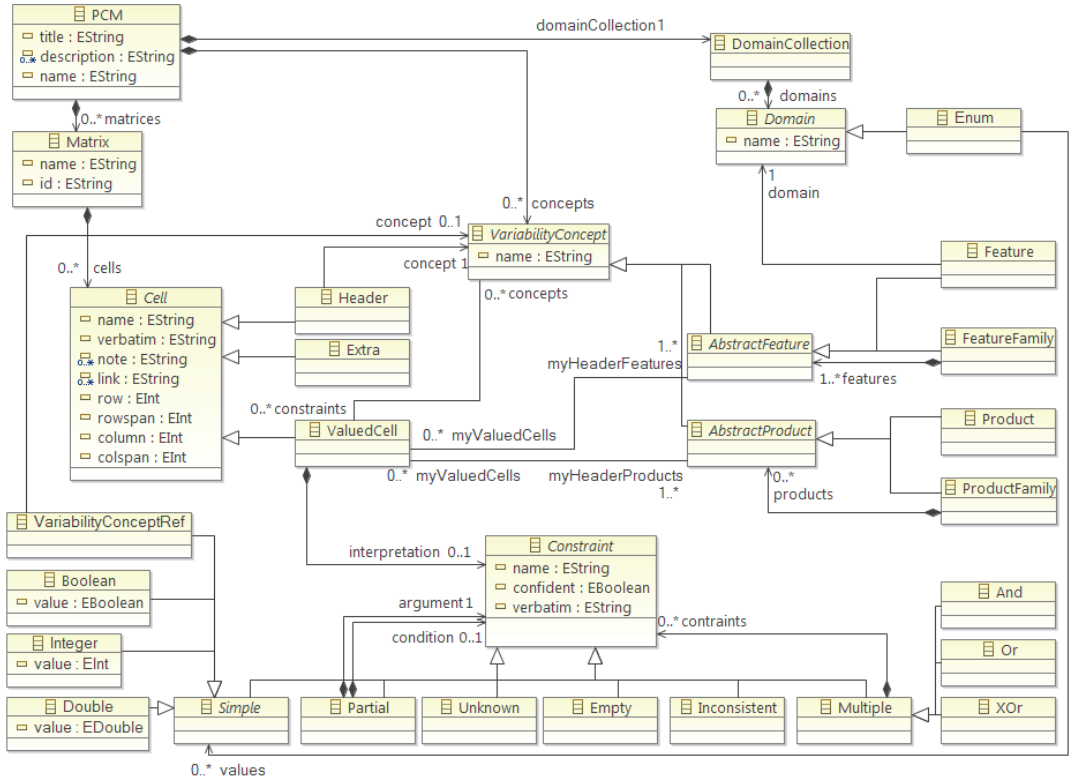


Figure 1.4: The PCM Metamodel [BSA+14]

explicit concepts of products and features. These products (resp. features) can have a composite structure that is used when describing several level of granularity for these products (resp. features) and which are usually represented as product (resp. features) row or column spans. In Excel or Wikipedia, cell values are associated to products and features because of their relative and fortunate positions. Bécan *et al.* [BSA+14] have explicit associations between a cell and its related product and feature. In addition, they keep the syntactic layout with the row and column attributes in the *Cell* class.

On the semantic side, a PCM expresses commonalities and differences between products. As a consequence, formalizing such domains necessarily requires to introduce some concepts from the variability and product line engineering community but also to introduce new ones. Two main concepts were introduced: the *Constraint* class that represents the interpretation of the information contained in a valued cell and the *Domain* class that define the possible values for a feature. The interpretation of a valued cell is given according to different patterns and information types defined as sub-concepts of *Constraint* in the metamodel:

- Boolean: states that the feature is present or not,
- Integer: integer number
- VariabilityConceptRef: references a product or a feature,

- Partial: states that the feature is partially or conditionally present,
- Multiple (And, Or, Xor): composition of values constrained by a cardinality,
- Unknown: states that the presence or absence of the feature is uncertain,
- Empty: the cell is empty,
- Inconsistent: the cell is inconsistent with the other cells bound to the same feature

The domain of a feature is represented as a set of Simple elements (*Boolean*, *Integer*, *Double* or *VariabilityConceptRef*) which defines the valid values for the cells that are related to this feature. The concept of domain allows to detect invalid values and reason on discrete values such as features but also use the properties of boolean, integers and real values for ranking or sorting operations.

A first advantage of this metamodel over spreadsheet applications (e.g. Excel), database or websites is that it contains explicit notions of products and features. With this metamodel, a comparator can directly reason in terms of these variability concepts. It does not have to care about the structural information (rows and columns) and the representation of the cell content. This eases the development and quality of comparators.

A second advantage is that the clear semantics of the cells enables the development of advanced reasoning facilities. The constraints defined by the cells can be easily encoded into state-of-the-art reasoners input format (e.g. CSP or SMT solvers). Such reasoners expect formatted and consistent data that cannot be provided without formalization. Based on these two previous advantages, comparators can build advanced filtering capabilities working on multiple criteria. The absence of structural constraints in the metamodel allows to reorganize products and features in order to visualize only the most important ones according to a user. This can reduce the cognitive effort required by a user to analyze a PCM. The reasoning facilities also allows to filter the products based on user-defined constraints or empirical data (e.g. best-selling product).

1.5 Requirements Engineering and Regulations

1.5.1 Requirements Engineering

Requirements engineering (RE), is an early stage in the software development life cycle, and plays an important role in successful information systems development. RE consists in a set of activities used by systems analysts to identify needs of a customer and assesses the functionality required in a proposed system [Poh94, BR02].

Definition 1.6 (Requirement) *"a condition or capability needed by a user to solve a problem or achieve an objective." Alternatively, it is defined as "a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents." [IEE90]*

Definition 1.7 (Requirements Engineering) *"The systematic process of developing requirements through an iterative co-operative process of analyzing the problem, doc-*

umenting the resulting observations in a variety of representations formats and checking the accuracy of the understanding gained" [LK95]

"Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints of software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families." [ZJ97]

Despite the evident and simple nature of these definitions, RE is a critical process where all the possible failures of a system should be identified to prevent them in the future and formalise what the future system must be. For that a wide variety of methods have been developed and in some software developments those ones must get involved at all stages of the life cycle.

Requirements Classification.

Software requirements are classified into two categories: user requirements and system requirements. User requirements, which are the high level abstract requirements, describe, either in plain language or graphically, the services and constraints of the system. Whereas, system requirements, which are the detailed description of the system, precisely describe the functions, services and operational constraints of the system in details, and acts as an agreement between users and developers. Following is the brief classification of the different software system requirements:

- *Functional requirements.* A functional requirement is a software requirement that specifies the function of a system or of on of its component. The primary objective of functional requirements is to define the behavior of the system, i.e., the fundamental processes or transformations that software and hardware components of the system perform on input to produce output.

- *Non-functional requirements.* A non-functional requirement is a software requirement that specifies the criterion to judge the behavior of a system, i.e., it describes how the software should perform rather than what it performs.

Requirements Engineering Process.

Different authors include heterogeneous sub-processes as part of requirements engineering but the common primary activities during different requirements engineering processes are elicitation, analysis and negotiation, verification and validation, change management, and requirements tracing.

Typically, a requirement is first *elicited*. In a second step the various stakeholders negotiate about the requirement, agree on it or change it accordingly. The requirement is then in the *specification/documentation* task integrated with the existing documentations and finally in the *validation/verification* task checked if it corresponds to the original *user/customer* needs (adapted to the limitations opposed on the requirements process by constraints) or conflicts with other documented requirements. Even when

the software is installed, new requirements may emerge. Thus, requirements management and tracing must be achieved. In following we describe the main goals of these tasks and their relations.

Requirements Elicitation. Every RE process somehow starts with the elicitation of the requirements, the needs, and the constraints about the system to be developed. The common methods used to gather requirements from stakeholders are interviews, questionnaires, observations, workshops, brainstorming, use cases, prototyping, ethnography, etc [GW89, BFJZ14, RP92, DJ85, SRS+93]. Different methods and tools, including rules of writing quality requirements are available in literature [Lam05, Hoo93, Fir03, Wie99].

Requirements Analysis & Negotiation. Requirements analysis is a process of categorization and organization of requirements into related subsets, exploration of relationships among requirements, examination of requirements for consistency, omissions and ambiguity, and ranking requirements based on the needs of customers [Pre05]. The structured analysis of the requirements can be achieved by analysis techniques, such as requirements animation, automated reasoning, knowledge-based critical analysis, consistency checking, analogical and case-based reasoning. It is common during the requirements analysis phase that different customers propose conflicting requirements, which from their point of view are essential for the system. The goal of the negotiation task is to establish an agreement on the requirements of the system among the various stakeholders involved in the process.

Requirements Specification & Documentation. Requirements specification is the activity of translating the information gathered during the analysis activity into a document that defines a set of requirements. Software Requirements Specifications is defined by the IEEE Computer Society [CB98, AB04] as "a process result of which are unambiguous and complete specification documents". The three most common classes of languages for requirement specifications are informal, semi-formal [Che76, YC79, Boo67] and formal languages [AAH05, SA92, Jon86]. The desirable characteristics for requirement specifications [CB98] are: complete, correct, ranked, unambiguous, consistent, modifiable, traceable verifiable, valid and testable.

Requirements Verification & Validation. The main goal is to analyze and ensure that the derived specification corresponds to the original stakeholder needs and conforms to the internal and/or external constraints set by the enterprise and its environments. V&V activities examine the specification to ensure that all system requirements have been stated unambiguously, consistently, completely, and correctly. Its task is to show that requirements model in some easily comprehensible form to customers. IEEE proposes a comprehensive verification and validation plan for the software development life-cycle.

Requirements Management. Requirements management is a set of activities during which we identify, control, and track any possible changes to requirements at any time during the life of the project. A track list must be kept for new requirements and also between dependent requirements because if one requirement changes, it may have effect on several other related requirements. Use of traceability policy to define and maintain the relationships among requirements is often advised along with Computer Aided Software Engineering (CASE) tool support for requirements management.

1.5.2 Requirements Engineering and Compliance with Regulations

Nature of Regulatory Requirements.

Software systems designed to perform safety functions must conform to an increasing set of regulatory requirements. In the nuclear energy domain, a licensee must therefore demonstrate that his system meets all regulatory requirements of a regulator. These requirements can be contained in regulatory documents, in guides, standards and even in tacit knowledge [SGN11] acquired from anterior projects in the past. This lays applicants with a huge and increasing amount of documents and information.

Regulatory requirements are complete in the sense that there are no others (even if you should consider them as incomplete). They are ambiguous [Kam05a], not clear and unverifiable. Finally, there is no way (within the scope of qualification) to change and improve them. Thus, these requirements are far from the usual separation between functional/non functional requirements and they are not concerned with requirements quality where the objectives are more to produce complete, verifiable, precise requirements or to try to reach this final state.

For example, the requirements related to the diversity or the independence between the lines of defense are relatively generic requirements, as they apply to systems that need to verify these properties, and have a major impact on the system architecture without having a particular influence from a functional point of view.

Similarly, the processes for quality assurance or validation and verification or documentation are important for safety while they have no impact on the system behavior in terms of function performed, performance, maintainability, and availability. However, they provide a certain level of reliability in the system design and validation process.

Compliance with Regulatory Requirements.

Software developers must ensure that the software they develop complies with relevant laws and regulations. Compliance with regulations, lost reputation, and brand damage resulting from privacy and security breaches are increasingly driving information security and privacy policy decisions [MAS+11]. The costs of noncompliance are significant.

Despite the high cost of noncompliance, developing legally compliant software is challenging. Legal texts contain ambiguities [BA+08, OA+07]. Requirements engineers need to understand domain-specific definitions and vocabulary before they can interpret and extract compliance requirements [OA+07]. Cross-references between different portions of a legal text can be ambiguous and force engineers to analyze the law in a non-sequential manner [Bre09, BA+08], and cross-references to external legal texts increase the number of documents engineers must analyze in order to obtain compliance requirements [OA+07].

Researchers are providing engineers with techniques and tools for specifying and managing software requirements for legally compliant systems [Bre09, CHCGE10, GAP09, MOA+09, MA10, MGL+06, SMPS09, You11]. Massey *et al.* use cross-references, along with other factors, to prioritize compliance requirements, but do not analyze the cross-referenced texts [MOA+09]. Requirements engineering research has focused on internal cross-references [Bre09, MA+09, MGL+06] rather than external cross-references.

The cross-references to external texts are important to analyze, because they may introduce conflicts or refine existing requirements. C. Maxwell *et al.* analyze each external cross-reference within the U.S. Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule to determine whether a cross-reference either: introduces a conflicting requirement, a conflicting definition, and/or refines an existing requirement [MAS+11]. Van Engers and Boekenoogen use scenarios and the Unified Modeling Language (UML) to detect errors in the law and improve legal text quality [vEB03]. Hamdaqa and Hamou-Lhadj present a classification scheme for legal cross-references outline a tool-supported, automated process for extracting cross-references and generating cross-reference graphs [HHL09].

Adedjouma *et al.* developed a framework for automated detection and resolution of cross references in legal texts [ASB14]. They ground their work on Luxembourg's legislative texts, both for studying the natural language patterns in cross reference expressions and for evaluating their solution. The approach is parameterized by a text schema, making it possible to tailor the approach to different legal texts and jurisdictions. Through a study of legislative texts in Luxembourg, they extended existing natural language patterns for cross reference expressions and provided a systematic way to interpret these expressions. Several other approaches are also dealing with automated supports for cross reference detection and resolution [PBM03, DWVE06, KZB+08].

Ghanavati *et al.* use compliance links to trace goals, softgoals, tasks and actors to the law [GAP09]. They use traceability links to connect portions of a Goal Requirements Language (GRL) business model with a GRL model of the law. Berenbach *et al.* use just in time tracing (JIT) to identify: (1) regulatory requirements; (2) system requirements that satisfy said requirements; and (3) sections of the law that require further analysis [BGCH10]. Zhang and Koppaka create legal citation networks based on the citations found in case law [ZK07].

Requirements researchers have examined conflicts in software requirements [BI96, EN95, RF94, TB07, VLDL98]. Robinson and Fickas describe how to detect and resolve requirements conflicts using a tool-supported approach [RF94]. Boehm and In use the WinWin model for negotiating resolutions to conflicts among quality attributes [BI96]. Van Lamsweerde *et al.* use KAOS to identify and resolve conflicts among software goals [VLDL98]. Easterbrook and Nuseibeh use the ViewPoints Framework to handle inconsistencies as a requirements specification evolves [EN95]. Emmerich *et al.* examine standards such as ISO and built a prototype policy checker engine in DOORS [EFM+99]. Thurimella and Bruegge examine conflicts among the requirements of various product lines [TB07].

Panesar-Walawege *et al.* [PWSBC10] developed an extensible conceptual model, based on the IEC 61508 standard, to characterize the chain of safety evidence that underlies safety arguments about software. The conceptual model captures both the information requirements for demonstrating compliance with IEC 61508 and the traceability links necessary to create a seamless chain of evidence. The model can be specialized according to the needs of a particular context and can facilitate software certification.

1.6 Conclusion

In this chapter, we have briefly introduced some principles and basic concepts we will use throughout the thesis including Product Line Engineering, Variability Modeling and Requirements Engineering. We have described in particular existing approaches for variability modeling and compliance with regulations. In the next chapter, we review existing NLP and data mining techniques, present some approaches to deal with synthesizing feature models and discuss their advantages and drawbacks.

Chapter 2

State of the Art

In this chapter, we study existing natural language processing and data mining techniques, as well as existing approaches for synthesizing feature models from different artifacts. Section 2.1 provides a survey of the most used statistical techniques to perform the construction of variability models. In Section 2.2, we review existing techniques for terminology and information extraction. Section 2.3 and Section 2.4 study and compare existing methods to respectively extract features and synthesize feature models from different artifacts. Section 2.5 discusses the limitations of the state of the art.

2.1 Statistical Techniques to Construct Variability Models

2.1.1 Text Mining

Text mining is defined by [UMN+04] as an extension of data mining or knowledge discovery, is a burgeoning new technology that refers generally to the process of extracting interesting and non-trivial patterns or knowledge from unstructured text documents. It is a multidisciplinary field, involving information retrieval, text analysis, information extraction, clustering, categorization, visualization, database technology, and machine learning.

Text mining can be visualized as consisting of two phases [T+99]: Text refining that transforms free-form text documents into a chosen intermediate form, and knowledge distillation that deduces patterns or knowledge from the intermediate form. Intermediate form (IF) can be semi-structured such as the conceptual graph representation, or structured such as the relational data representation. Intermediate form can be document-based wherein each entity represents a document, or concept-based wherein each entity represents an object or concept of interests in a specific domain.

Mining a document-based IF deduces patterns and relationship across documents. Document clustering/visualization and categorization are examples of mining from a document-based IF. Mining a concept-based IF derives pattern and relationship across objects or concepts. Data mining operations, such as predictive modeling and associative discovery, fall into this category. A document-based IF can be transformed into a concept-based IF by realigning or extracting the relevant information according to the

objects of interests in a specific domain. It follows that document-based IF is usually domain-independent and concept-based IF is domain-dependent.

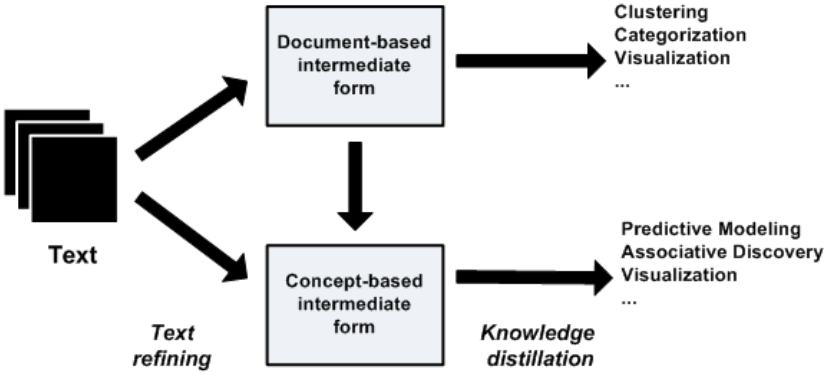


Figure 2.1: Text Mining Framework [T+99]

There are two main categories of text mining products: Document visualization and Text analysis/understanding. The first one consists in organizing documents based on their similarities and graphically represents the groups or clusters of the documents. The second group is mainly based on natural language processing techniques, including text analysis, text categorization, information extraction, and summarization.

Today, text mining plays an important role on document analysis. Subjects as semantic analysis, multilingual text processes and domain knowledge have been explored and studied to derive a sufficiently rich representation and capture the relationship between the objects or concepts described in the documents, produce language-independent intermediate forms and to improve parsing efficiency and derive a more compact intermediate form. There have been some efforts in developing systems that interpret natural language queries and automatically perform the appropriate mining operations.

Text mining techniques have been often implemented successfully in the requirement elicitation field, reinforcing human-intensive task in which analyst proactively identify stakeholders' needs, wants, and desires using a broad array of elicitation tools such as interviews, surveys, brainstorming sessions, joint application design and ethnographic studies. All of this tools are frequently based on human interpretation, natural language and unstructured data. They are "expensive" especially when we talk about acquiring requirements on large scale projects. Castro *et al.* [CHDCHM08] use text mining to elicit needs in their approach, while using TF-IDF (term frequency, inverse document frequency) and remove common (stop) words (e.g. "be" and "have"). TF-IDF weight terms more highly if they occur less frequently and are therefore become more useful in expressing unique concepts in the domain. Niu *et al.* [NE08a] were not only interested in the relevance, but also the quantity of information of a word within a corpus. This measure is defined by it information content as:

$$INFO(w) = -\log_2(P\{w\}) \tag{2.1}$$

where $P\{w\}$ is the observed probability of occurrence of w in a corpus. They also adopted verb direct object correlations to determine lexical affinities between two units of language in a document. Noppen *et al.* [NvdBWR09] use the latent semantic analysis (LSA) on his approach to identify similarity between needs to further clustered them. LSA [SS06b] considers texts to be similar if they share a significant amount of concepts. These concepts are determined according to the terms they include with respect to the terms in the total document space.

As we have notice on requirement engineering applications; once the concept were mined via text mining tools those were then classified or grouped in clusters. Thus clustering is a wide technique used in software engineering as we will see in next section.

2.1.2 Clustering

Clustering is a division of data into groups of similar objects. Each group, called cluster, consists of objects that are similar between themselves and dissimilar to objects of other groups. Representing data by fewer clusters necessarily loses certain fine details (akin to lossy data compression), but achieves simplification. It represents many data objects by few clusters, and hence, it models data by its clusters [RS10].

Data modeling puts clustering in a historical perspective rooted in mathematics, statistics, and numerical analysis. From a machine learning perspective clusters correspond to hidden patterns, the search for clusters is unsupervised learning, and the resulting system represents a data concept. Therefore, clustering is unsupervised learning of a hidden data concept. Data mining deals with large databases that impose on clustering analysis additional severe computational requirements. These challenges led to the emergence of powerful broadly applicable data mining clustering methods. The output clustering (or clusterings) can be hard (a partition of the data into groups) or fuzzy (where each pattern has a variable degree of membership in each of the output clusters) [JMF99]. Hierarchical clustering algorithms produce a nested series of partitions based on a criterion for merging or splitting clusters based on similarity. Partitional clustering algorithms identify the partition that optimizes (usually locally) a clustering criterion. Additional techniques for the grouping operation include probabilistic [Bra91] and graph-theoretic [Zah71] clustering methods.

Clustering techniques have been used by researchers to support a number of activities such as information retrieval performance improvement [Kow98], document browsing [CKPT92], topics discovery [ESK04], organization of search results [ZEMK97], and concept decomposition [DM01]. For document clustering the hierarchical approach is generally preferred because of it natural fit with the hierarchy found in many documents [ZK02]. Hsia *et al.* [HHKH96] and Yaung [Yau92] have used agglomerative hierarchical algorithms for clustering requirements in order to facilitate incremental delivery by constructing proximities based on the references requirements to a set of systems components. Al-Otaiby *et al.* [AOAB05] used traditional hierarchical clustering algorithm to enhance design modularity by computing proximities as a function of concepts shared between pairs of requirements. Chen [CZZM05] computed proximities by manually evaluated requirements to identify resource accesses such as reading or writing

to file, and from this used iterative graph based clustering to automatically construct a feature model. Goldin *et al.* [GB97] implemented an approach based on signal processing to discover abstractions from a large quantity of natural language requirement texts. Laurent *et al.* [LCHD07] apply a multiple orthogonal clustering algorithms to capture the complex and diverse roles played by individual requirements. This knowledge is the used to automatically generate a list of prioritized requirements. Castro *et al.* [CHDCHM08] used the bisecting clustering algorithm to dynamically build the discussion forum by means of requirements. They also used clustering to identify user profile and predictive level of interest of forum participants.

2.1.3 Association Rules

Association rules are an important class of regularities in data. Mining of association rules is a fundamental data mining task. It is perhaps the most important model invented and extensively studied by the database and data mining community. Its objective is to find all co-occurrence relationships, called associations, among data items [Liu07]. Since it was first introduced in 1993 by Agrawal *et al.* [AIS93], it has attracted a great deal of attention. Initial research was largely motivated by the analysis of market basket data, the results of which allowed companies to more fully understand purchasing behavior and, as a result, better target market audiences. One common example is that diapers and beer often are sold together. Such information is valuable for cross-selling, thus increasing the total sales of a company. For instance, a supermarket can place beer next to diapers hinting to parents that they should buy not only necessities for their baby but also luxury for themselves.

Association mining is user-centric as the objective is the elicitation of useful (or interesting) rules from which new knowledge may be derived. The key characteristics of usefulness suggested in the literature are that the rules are novel, externally significant, unexpected, non-trivial and actionable [BJA99, DL98, Fre99, HH99, HH01, RR01, Sah99, ST95]. The association mining system's role in this process is to facilitate the discovery, heuristically filter and enable the presentation of these inferences or rules for subsequent interpretation by the user to determine their usefulness.

Maedche *et al.* [Mae02] used a modification of the generalized association rule learning algorithm for discovering properties between classes. The algorithm generates association rules comparing the relevance of different rules while climbing up and/or down the taxonomy. The apparently most relevant binary rules are proposed to the ontology engineer for modeling relations into the ontology, thus extending. To restrict the high number of suggested relations they defined so-called restriction classes that have to participate in the relations that are extracted. Li *et al.* [LZ05] proposed a method called PR Miner that uses frequent item-set mining to efficiently extract implicit programming rules from large software code written in an industrial programming language such as C, requiring little effort from programmers and no prior knowledge of the software. PR-Miner can extract programming rules in general forms (without being constrained by any fixed rule templates) that can contain multiple program elements of various types such as functions, variables and data types. In addition, they proposed an efficient

algorithm to automatically detect violations to the extracted programming rules, which are strong indications of bugs. Jiao *et al.* [JZ05] in their approach to associate customer needs to functional requirements, used association rules mining technique. By means of this technique they explain the meaning of each functional requirement cluster as well as the mapping of customer needs to each cluster.

2.2 Mining Knowledge using Terminology and Information Extraction

2.2.1 Terminology Extraction

The studies on the definition and implementation of methodologies for extracting terms from texts assumed since the beginning a central role in the organization and harmonization of the knowledge enclosed in domain corpora, through the use of specific dictionaries and glossaries[PPZ05]. Recently, the development of robust computational NLP approaches to terminology extraction, able to support and speed up the extraction process, lead to an increasing interest in using terminology also to build knowledge bases systems by considering information enclosed in textual documents. In fact, both Ontology Learning and Semantic Web technologies often rely on domain knowledge automatically extracted from corpus through the use of tools able to recognize important concepts, and relations among them, in form of terms and terms relations.

Definition 2.1 (Terminology Extraction) *"the task of identifying domain specific terms from technical corpora."[KKM08]*

"the task of automatically extracting terms or keywords from text."[KU96, HB08, CCCB+08]

Starting from the assumption that terms unambiguously refer to domain-specific concepts, a number of different methodologies has been proposed so far to automatically extract domain terminology from texts. Generally speaking, the term extraction process consists of two fundamental steps: 1) identifying term candidates (either single or multi-word terms) from text, and 2) filtering through the candidates to separate terms from non-terms. To perform these two steps, term extraction systems make use of various degrees of linguistic filtering and, then, of statistical measures ranging from raw frequency to Information Retrieval measures such as Term Frequency/Inverse Document Frequency (TF/IDF) [SB88], up to more sophisticated methods such as the C-NC Value method [FA99], or lexical association measures like log likelihood [Dun93] or mutual information. Others make use of extensive semantic resources [MA99], but as underlined in Basili *et al.* [BPZ01], such methods face the hurdle of portability to other domains.

Another interesting line of research is based on the comparison of the distribution of terms across corpora of different domains. Under this approach, identification of relevant term candidates is carried out through inter-domain contrastive analysis [PVG+01, CN04, BMPZ01]. Interestingly enough, this contrastive approach has so far been applied

only to the extraction of single terms, while, multi-word terms' selection is based upon contrastive weights associated to the term syntactic head. This choice is justified by the assumption that multiword terms typically show low frequencies making contrastive estimation difficult [BMPZ01]. On the contrary, Bonin *et al.* [BDVM10] focused their attention on the extraction of multi-word terms, which have been demonstrated to cover the vast majority of domain terminology (85% according to Nakagawa *et al.* [NM03]); for this reason, they have to be considered independently from the head.

2.2.2 Information Extraction

Information extraction (IE) is the task of finding structured information from unstructured text. It is an important task in text mining and has been extensively studied in various research communities including natural language processing, information retrieval and Web mining. It has a wide range of applications in domains such as biomedical literature mining and business intelligence.

Definition 2.2 (Information Extraction) *"to identify a predefined set of concepts in a specific domain, ignoring other irrelevant information, where a domain consists of a corpus of texts together with a clearly specified information need. In other words, IE is about deriving structured factual information from unstructured text." [PY13]*

"to identify instances of a particular prespecified class of entities, relationships and events in natural language texts, and the extraction of the relevant properties (arguments) of the identified entities, relationships or events." [PY13]

Template Filling. Many texts describe recurring stereotypical situations. The task of template filling is to find such situations in documents and fill the template slots with appropriate material. These slot-fillers may consist of text segments extracted directly from the text, or concepts like times, amounts, or ontology entities that have been inferred from text elements through additional processing.

Standard algorithms for template-based information extraction require full knowledge of the templates and labeled corpora, such as in rule-based systems [CLH93, RKJ+92] and modern supervised classifiers [Fre98, CNL03, BM04, PR09]. Classifiers rely on the labeled examples surrounding context for features such as nearby tokens, document position, named entities, semantic classes, syntax, and discourse relations [MC07]. Ji and Grishman [JG08] also supplemented labeled with unlabeled data.

Weakly supervised approaches remove some of the need for fully labeled data. Most still require the templates and their slots. One common approach is to begin with unlabeled, but clustered event-specific documents, and extract common word patterns as extractors [RS98, SSG03, RWP05, PR07]. In particular, Riloff and Schmelzenbach [RS98] have developed a corpus-based algorithm for acquiring conceptual case frames empirically from unannotated text. Sudo *et al.* [SSG03] introduce an extraction pattern representation model based on subtrees of dependency trees, so as to extract entities beyond direct predicate-argument relations. Riloff *et al.* [RWP05] explore the idea of using subjectivity analysis to improve the precision of information extraction systems by automatically filtering extractions that appear in subjective sentences.

Filatova *et al.* [FHM06] integrate named entities into pattern learning to approximate unknown semantic roles. Bootstrapping with seed examples of known slot fillers has been shown to be effective [STA06, YGTH00].

Marx *et al.* proposed a cross-component clustering algorithm for unsupervised information extraction [MDS02]. The algorithm assigns a candidate from a document to a cluster based on the candidate's feature similarity with candidates from other documents only. In other words, the algorithm prefers to separate candidates from the same document into different clusters. Leung *et al.* proposed a generative model to capture the same intuition [LJC+11]. Specifically, they assume a prior distribution over the cluster labels of candidates in the same document where the prior prefers a diversified label assignment. Their experiments show that clustering results are better with this prior than without using the prior.

Shinyama and Sekine [SS06a] describe an approach to template learning without labeled data. They present unrestricted relation discovery as a means of discovering relations in unlabeled documents, and extract their fillers. Central to the algorithm is collecting multiple documents describing the same exact event, and observing repeated word patterns across documents connecting the same proper nouns. Learned patterns represent binary relations, and they show how to construct tables of extracted entities for these relations. The limitations to their approach are that (1) redundant documents about specific events are required, (2) relations are binary, and (3) only slots with named entities are learned. Large-scale learning of scripts and narrative schemas also captures template-like knowledge from unlabeled text [CJ08, KO10]. Scripts are sets of related event words and semantic roles learned by linking syntactic functions with coreferring arguments. While they learn interesting event structure, the structures are limited to frequent topics in a large corpus.

Chambers and Jurafsky presented a complete method that is able to discover multiple templates from a corpus and give meaningful labels to discovered slots [CJ11]. Specifically, their method performs two steps of clustering where the first clustering step groups lexical patterns that are likely to describe the same type of events and the second clustering step groups candidate role fillers into slots for each type of events. A slot can be labeled using the syntactic patterns of the corresponding slot fillers. For example, one of the slots discovered by their method for the bombing template is automatically labeled as "Person/Organization who raids, questions, discovers, investigates, diffuses, arrests." A human can probably infer from the description that this refers to the police slot.

Extraction of Product Attributes. In the field of information extraction from product reviews, most of the work has focused on finding the values for a set of predefined attributes. Recently, there has been growing interest in the automated learning of the attributes themselves, and then finding the associated values. One of the initial papers in the field of "attribute" extraction by Hu *et al.* [HL04] talks about using a frequency based approach to identifying the features in product reviews. They order the noun phrases by frequency and then have different manually defined settings to find the

features (like lower cutoff, upper cutoff, etc). Though they are able to achieve a good workable system with these methods, their assumption that a feature would always be a noun is not always true. There can be multi word features like "optical zoom", "hot shoe flash" where one of the words is an adjective. They take a more holistic approach to the problem and use the opinion (sentiment) words to find infrequent features.

In [GPL+06], Ghani *et al.* have shown success in attribute-value pair extraction using co-EM and Naive Bayes classifiers. However, their work focused on official product description from merchant sites, rather than on reviews. Popescu *et al.*'s OPINE system [PE07] also uses the dataset provided in [HL04]. They explicitly extract noun phrases from the reviews (with a frequency based cutoff) after parts of speech (POS) tagging and then compute Pointwise Mutual Information scores between the phrase and meronymy discriminators associated with the product class. This again assumes that features are always nouns and misses out on features which are not nouns or are combination of different POS tags.

Gupta *et al.* [GKG09] provide a method for finding the key features of products by looking at a number of reviews of the same product. The goal is to use the language structure of a sentence to determine if a word is a feature in the sentence. For this they propose an approach in which we first run a POS tagger on the reviews data, and then generate input vectors using these POS tagged reviews. They formulate the problem of extracting features as a classification problem, where given a word, the goal is to classify it as a feature or not-feature.

Bing *et al.* [BWL12] developped an unsupervised learning framework for extracting popular product attributes from different Web product description pages. Unlike existing systems which do not differentiate the popularity of the attributes, they propose a framework which is able not only to detect concerned popular features of a product from a collection of customer reviews, but also to map these popular features to the related product attributes, and at the same time to extract these attributes from description pages. They developped a discriminative graphical model based on hidden Conditional Random Fields. The goal of extracting popular product attributes from product description Web pages is different from opinion mining or sentiment detection research as exemplified in [DLZ09, KIM+04, LHC05, PE07, TTC09, Tur02, ZLLOS10]. These methods typically discover and extract all product attributes as well as opinions directly appeared in customer reviews. In contrast, the goal here is to discover popular product attributes from description Web pages.

Some information extraction approaches for Web pages rely on wrappers which can be automatically constructed via wrapper induction. For example, Zhu *et al.* developed a model known as Dynamic Hierarchical Markov Random Fields which is derived from Hierarchical CRFs (HCRF) [ZNZW08]. Zheng *et al.* proposed a method for extracting records and identifying the internal semantics at the same time [ZSWG09]. Yang *et al.* developed a model combing HCRF and Semi-CRF that can leverage the Web page structure and handle free texts for information extraction [YCN+10].

Luo *et al.* studied the mutual dependencies between Web page classification and data extraction, and proposed a CRF-based method to tackle the problem [LLX+09]. Some common disadvantages of the above supervised methods are that human effort is needed

to prepare training examples and the attributes to be extracted are pre-defined. Some existing methods have been developed for information extraction of product attributes based on text mining. Probst *et al.* proposed a semi-supervised algorithm to extract attribute value pairs from text description [PGK+07]. Their approach aims at handling free text descriptions by making use of natural language processing techniques.

2.3 Approaches for Mining Features and Constraints

Most of the works focus on the extraction of features from natural language requirements and legacy documentation [Fox95, CZZM05, ASB+08a, NE08a, NE08b, WCR09]. The DARE tool [Fox95] is one of the earliest contribution in this sense. A semi-automated approach is employed to identify features according to lexical analysis based on term frequency (i.e., frequently used terms are considered more relevant for the domain). Chen *et al.* [CZZM05] suggest the usage of the clustering technology to identify features: requirements are grouped together according to their similarity, and each group of requirements represents a feature.

Clustering is also employed in the subsequent works [ASB+08a, NE08a, NE08b, WCR09], but while in [CZZM05] the computation of the similarity among requirements is manual, in the other works automated approaches are employed. In particular, [ASB+08a] use IR-based methods, namely the Vector Similarity Metric (VSM) and Latent Semantic Analysis (LSA). With VSM, requirements are represented as vectors of terms, and compared by computing the cosine among the vectors. With LSA, requirements are similar if they contain semantically similar terms. Two terms are considered semantically similar if they normally occur together in the requirements document.

LSA is also employed by Weston *et al.* [WCR09], aided with syntactic and semantic analysis, to extract the so-called Early Aspects. These are cross-cutting concerns that are useful to derive features. Finally, Niu *et al.* [NE08a, NE08b] use Lexical Affinities (LA) – roughly, term co-occurrences – as the basis to find representative expressions (named Functional Requirements Profiles) in functional requirements.

All the previously cited works use requirements as the main source for feature mining. Other works [Joh06, DGH+11, ACP+12b] present approaches where public product descriptions are employed. While in [Joh06] the feature extraction process is manual, the other papers suggest automated approaches. The feature mining methodology presented in [DGH+11] is based on clustering, and the authors provide also automated approaches for recommending useful features for new products. Instead, the approach presented in [ACP+12b] is based on searching for variability patterns within tables where the description of the products are stored in a semi-structured manner. The approach includes also a relevant part of feature model synthesis. Ferrari *et al.* [FSd13] apply natural language processing techniques to mine commonalities and variabilities from brochures. They conducted a pilot study in the metro systems domain showing the applicability and the benefits in terms of user effort.

Regardless of the technology, the main difference between [DGH+11], [ACP+12b] and [FSd13] is that the former two rely on feature descriptions that are rather struc-

tured. Indeed, in [DGH+11] the features of a product are expressed with short sentences in a bullet-list form, while in [ACP+12b] features are stored in a tabular format. Instead, Ferrari *et al.* [FSd13] deal with brochures with less structured text, where the features have to be discovered within the sentences.

Nadi *et al.* [NBKC14] developed a comprehensive infrastructure to automatically extract configuration constraints from C code. Ryssel *et al.* developed methods based on Formal Concept Analysis and analyzed incidence matrices containing matching relations [RPK11]. Bagheri *et al.* [BEG12] proposed a collaborative process to mine and organize features using a combination of natural language processing techniques and Wordnet.

2.4 Approaches for Feature Models Synthesis

2.4.1 Synthesis of FMs from configurations/dependencies

Techniques for synthesizing an FM from a set of dependencies (e.g., encoded as a propositional formula) or from a set of configurations (e.g., encoded in a product comparison matrix) have been proposed [ABH+13b, ACSW12, CSW08, CW07b, HLHE11, HLHE13, JKW08, LHGB+12, LHLG+15, SLB+11b].

In [ACSW12, CW07b], the authors calculate a diagrammatic representation of all possible FMs, leaving open the selection of the hierarchy and feature groups.
Andersen *et al.* [ACSW12] address the problem of automatic synthesis of feature models from propositional constraints. They propose techniques for synthesis of models from respectively conjunctive normal form (CNF) and disjunctive normal form (DNF) formulas. The authors construct diagrams that contain a hierarchy of groups of binary features enriched by cross-hierarchy inclusion/exclusion constraints. The algorithms assume a constraint system expressed in propositional logics as input. In practice, these constraints can be either specified by engineers, or automatically mined from the source code using static analysis [BSL+10]. Technically, they synthesize not a feature model (FM), but a feature graph (FG), which is a symbolic representation of all possible feature models that could be sound results of the synthesis. Then, any of these models can be efficiently derived from the feature graph.

Janota *et al.* [JKW08] offer an interactive editor, based on logical techniques, to guide users in synthesizing an FM. The algorithms proposed in [HLHE11, HLHE13, LHGB+12] do not control the way the feature hierarchy is synthesized either. In addition no user support is provided to interactively synthesize or refactor the resulting FM. In [ABH+13b], authors present a synthesis procedure that processes user-specified knowledge for organizing the hierarchy of features. The effort may be substantial since users have to review numerous potential parent features.

She *et al.* [SLB+11b] propose an heuristic to rank the correct parent features in order to reduce the task of a user. Though the synthesis procedure is generic, they assume the existence of feature descriptions in the software projects Linux, eCos, and FreeBSD. The authors showed that their attempts to fully synthesize an FM do not lead to a desirable hierarchy – such as the one from reference FMs used in their evaluation

– coming to the conclusion that an additional expert input is needed.

Yi *et al.* [YZZ+12] applied support vector machine and genetic techniques to mine binary constraints (requires and excludes) from Wikipedia. They evaluated their approach on two feature models of SPLOT. Lora *et al.* [LMSM] propose an approach that integrates statistical techniques to identify commonality and variability in a collection of a non predefined number of product models. This approach constructs a product line model from structured data: bill of materials (BOM) and does not pay attention to supporting imperfect information.

An important limitation of prior works is the identification of the feature hierarchy when synthesizing the FM, that is, the user support is either absent or limited. In [BABN15], we defined a generic, ontologic–aware synthesis procedure that computes the likely siblings or parent candidates for a given feature. We developed six heuristics for clustering and weighting the logical, syntactical and semantical relationships between feature names. A *ranking list* of parent candidates for each feature can be extracted from the weighted *Binary Implication Graph* which represents all possible hierarchies of an FM. In addition, we performed hierarchical *clustering* based on the similarity of the features to compute groups of features.

The heuristics rely on general ontologies, e.g. from Wordnet or Wikipedia. We also proposed an hybrid solution combining both ontological and logical techniques. We conducted an empirical evaluation on hundreds of FMs, coming from the SPLOT repository and Wikipedia. We provided evidence that a fully automated synthesis (i.e., without any user intervention) is likely to produce FMs far from the ground truths.

All methods presented above constructs variability models from structured data and not from informal documentation.

2.4.2 Synthesis of FMs from product descriptions

Acher *et al.* [ACP+12a] propose a semi-automated procedure to support the transition from structured product descriptions (expressed in a PCM) to FMs. They provide a dedicated language that can be used by a practitioner to parameterize the extraction process. The language supports scoping activities allowing to ignore some features or some products. It also enables practitioners to specify the interpretation of data in terms of variability and to set a feature hierarchy if needs be.
The second step of their approach is to synthesize an FM characterizing the valid combinations of features (configurations) supported by the set of products. Several FMs, representing the same set of configurations but according to different feature hierarchies, can be derived. They define a specific merging algorithm that first compute the feature hierarchy and then synthesize the variability information (mandatory and optional features, Mutex-, Xor and Or-groups, (bi-)implies and excludes constraints) using propositional logic techniques.
The authors showed that, although many feature groups, implies and excludes constraints are recovered, a large amount of constraints is still needed to correctly represent the valid combinations of features supported by the products. Their initial study was rather informal and conducted on a synthetic and limited data sample. Moreover,

this approach does not handle variability in informal documents since it takes as input product descriptions expressed in a tabular format.

Dumitru *et al.* [DGH+11] developed a recommender system that models and recommends product features for a given domain. Their approach mines product descriptions from publicly available online specifications, utilizes text mining and a novel incremental diffusive clustering algorithm to discover domain-specific features, generates a probabilistic feature model that represents commonalities, variants, and cross-category features, and then uses association rule mining and the k-Nearest-Neighbor machine learning strategy to generate product specific feature recommendations.

Davril *et al.* [DDH+13b] present a fully automated approach, based on prior work [HCHM+13b], for constructing FMs from publicly available product descriptions found in online product repositories and marketing websites such as SoftPedia and CNET. The proposal is evaluated in the anti-virus domain. The task of extracting FMs involves mining feature descriptions from sets of informal product descriptions, naming the features, and then discovering relationships between features in order to organize them hierarchically into a comprehensive model.

Indeed, product specifications are first processed in order to identify a set of features and to generate a product-by-feature matrix. Then, meaningful feature names are assigned and a set of association rules are mined for these features. These association rules are used to generate an implication graph (IG) which captures binary configuration constraints between features. The tree hierarchy and then the feature diagram are generated given the IG and the content of the features. Finally, cross-tree constraints and OR-groups of features are identified.

To evaluate the quality of FMs, the authors first explored the possibility of creating a "golden answer set" and then comparing the mined FM against this standard. The results showed that the generated FMs do not reach the same level of quality achieved in manually constructed FMs. The evaluation involved manually creating one or more FMs for the domain, and then asking users to evaluate the quality of the product lines in a blind study.

2.4.3 Synthesis of FMs from requirements

Chen *et al.* [CZZM05], Alves *et al.* [ASB+08b], Niu *et al.* [NE09], and Weston *et al.* [WCR09] use information retrieval (IR) techniques to abstract requirements from existing specifications, typically expressed in natural language.

Niu *et al.* [NE08a] provide a semi-automated approach for extracting an SPL's functional requirements profiles (FRPs) from natural language documents. They adopt the Orthogonal Variability Model (OVM) [PBvdL05b] to represent the extraction result and therefore manage the variability across, requirements, design, realization and testing artifacts. FRPs, which capture the domain's action themes at a primitive level, are identified on the basis of lexical affinities that bear a verb-DO (direct object) relation. They also analyze the essential semantic cases associated with each FRP in order to model SPL variabilities [NE08a] and uncover early aspects [NE08b].

However, identifying aspects is achieved by organizing FRPs into overlapping clus-

ters [NE08b] without explicitly considering quality requirements. Moreover, we can point out that this approach experiments failures related to lack of semantics, and then an analyst should pass over the heuristics to verify and validate them. We can also notice that there is no heuristics that exclude relationships. For all these reasons, this approach cannot be considered to be totally automated and depends on the domain analyst personal understanding.

Chen *et al.* [CZZM05] propose a requirements clustering based approach to construct feature models from functional requirements of sample applications. For each application, tight–related requirements are clustered into features, and then functional features are organized into an application feature model. All the application feature models are merged into a domain feature model, and the variable features are also labeled. Nevertheless, requirements similarity is performed manually based on the expertise of the domain analyst. They use the concept of resource to define similarity: requirements are similar whenever they share resources. Thus, this approach requires the frequent intervention of an analyst. It will be then difficult to handle in large size projects. Moreover, this approach does not address transversal dependencies between features (requires and exclude constraints). The analyst should manipulate the product line model by including these relationships.

Alves *et al.* [ASB+08b] also propose an approach which, based on a clustering algorithm, generates features models. They instead employ automatic IR techniques, Vector Space Model (VSM) and Latent Semantic Analysis (LSA), to compute requirements similarity. Clusters of requirements are then identified and these are abstracted further into a configuration. The configurations corresponding to all requirement documents are merged into a fully-fledged feature model. However, features closer to the root comprise an increasingly high number of requirements. Therefore, the authors need to identify scalable and systematic naming of features in configurations and propose some heuristics.

Weston *et al.* [WCR09] introduce a tool suite that automatically processes natural–language requirements documents into a candidate feature model, which can be refined by the requirements engineer. The framework also guides the process of identifying variant concerns and their composition with other features. The authors also provide language support for specifying semantic variant feature compositions which are resilient to change.

Niu *et al.* [NE09] investigate both functional and quality requirements via concept analysis [GW12]. The goal is to efficiently capture and evolve a SPL's assets so as to gain insights into requirements modularity. To that end, they set the context by leveraging functional requirements profiles and the SEI's quality attribute scenarios [BCK03]. By analyzing the relation in context, the interplay among requirements are identified and arranged in a so–called concept lattice. The authors then formulate a number of problems that aspect–oriented SPL RE should address, and present their solutions according to the concept lattice. In particular, they locate quality–specific functional units, detect interferences, update the concept hierarchy incrementally, and analyze the change impact.

To deal with large-scale projects Castro *et al.* [CHDCHM08] propose an hybrid

recommender system that recommends forums to stakeholders and infers on knowledge of the user by examining the distribution of topics across the stakeholders' needs. The first step consists on gathering needs using a web enabled elicitation tool. The needs are then processed using unsupervised clustering techniques in order to identify dominant and cross cutting themes around which a set of discussion forums is created. To help keep stakeholders informed of relevant forums and requirements, they use a collaborative recommender system which recommends forums based on the interests of similar stakeholders. These additional recommendations increase the likelihood that critical stakeholders will be placed into relevant forums in a timely manner. Their approach is centered on the requirement elicitation process but can be used to obtain the basis for constructing product line models when we are faced with extremely high number of requirements.

Also Laurent *et al.* [LCHD07] propose a method to not only face that kind of situations but also allowing budgetary and short time to market deadlines restrictions. The goal is then to prioritize requirements and decide, given the available personnel, time and other resources, which one to include in a product release. They therefore propose an approach for automating a significant part of the prioritization process. The proposed method utilizes a probabilistic traceability model combined with a standard hierarchical clustering algorithm to cluster incoming stakeholder requests into hierarchical feature sets. Additional cross-cutting clusters are then generated to represent factors such as architecturally significant requirements or impacted business goals. Prioritization decisions are initially made at the feature level and then more critical requirements are promoted according to their relationships with the identified cross-cutting concerns. The approach is illustrated and evaluated through a case study applied to the requirements of the ice breaker system.

Different prioritization techniques are used [KR97, Mea06, Moi00, BR89, ASC07]. Often stakeholders simply place requirements into distinct categories such as mandatory, desirable, or inessential [Bra90]. They can also quantitatively rank the requirements [Kar95]. Analytical Hierarchy Process AHP [KR97] uses a pair wise comparison matrix to compute the relative value and costs of individual requirements in respect to one another. Theory WW, also known as Win Win [BR89], requires each stakeholder to categorize requirements in order of importance and perceived risk. Stakeholders then work collaboratively to forge an agreement by identifying conflicts and negotiating a solution. The Requirement Prioritization Framework supports collaborative requirements elicitation and prioritization and includes stakeholders profiling as well as both quantitative and qualitative requirements rating [Moi00]. Value oriented prioritization VOP incorporate the concept of perceived value, relative penalty, anticipated cost and technical risks to help select core requirements[ASC07]. These techniques described above were developed to treat only requirement elicitation process. They are centered based on stakeholder participation and negotiation.

Vague, conflicting, imperfect and inaccurate information can severely limit the effectiveness of approaches that derive features product line models from textual requirement specifications. The influence of imperfect information on feature diagrams is well recognized [PRWB04, RP04]. Kamsties [Kam05a] identifies that ambiguity in requirement

specifications needs to be understood before any subsequent design can be undertaken. Noppen *et al.*'s approach [NvdBWR09] defines a first step by proposing the use of fuzzy feature diagrams and design steps to support these models, but it does not provide an automatic method to construct them.

2.5 Discussion and Synthesis

The analysis of the state-of-the-art reveals these limitations:

- **Lack of variability formalization for safety requirements.** Safety requirements are provided in large and heterogeneous documents such as laws, standards or regulatory texts. Regulatory requirements are most often disconnected from the technical system requirements, which capture the expected system behavior. These requirements are ambiguous, not clear and unverifiable, leaving a large margin for interpretation. They express high level objectives and requirements on the system. Furthermore, regulation changes over time and from one country to another. Several existing methods handle managing variability in requirements specifications. Yet, few of them address modeling variability in regulatory requirements. Formalizing requirements variability is crucial to easily perform the certification of safety systems in different countries.

- **Few automated approaches are addressing variability extraction from informal documentation.** Constructing variability models is a very arduous and time-consuming task for domain engineers, especially if they are presented with heterogeneous and unstructured documentation (such as interview transcripts, business models, technical specifications, marketing studies and user manuals) from which to derive the variability model. And as natural language is inherently inaccurate, even standardized documentation will contain ambiguity, vagueness and conflicts. Thus, deriving an accurate variability model from informal documentation remains a hard and complex activity and still mostly relies on the experience and expertise of domain engineers. Several approaches have been proposed to mine variability and support domain analysis [ACP+12a, YZZ+12, DDH+13a, LHLG+14, BABN15]. However, few of them pay attention to adopt automated techniques for the construction of variability models from unstructured and ambiguous documents.

- **Limits of existing works when reverse engineering feature models from requirements.** Data mining techniques, such as clustering, have been considered by many authors to elicit (text mining) and obtain a feature diagram [CZZM05, NE09, NvdBWR09, ASB+08b, WCR09]. Heuristics and algorithms have been developed forgetting in most cases the study of transversal relationships. These methods currently proposed are not affordable for large scale projects. However, several techniques have been developed to help the analyst in the decision maker process by prioritizing and triage requirements. They are centered based on

stakeholder participation and negotiation and still remain quite manual. It will be useful to develop a method that could handle a large number of requirements and assist domain experts when building feature model using automated techniques.

- **Limits of existing approaches when extracting PCMs from informal product descriptions.** Many existing techniques that have been implemented to extract variability from informal product descriptions rely on PCMs, but with only boolean features. However, as showed earlier, PCMs contain more than simple boolean values and products themselves have variability, calling for investigating (a) the development of novel synthesis techniques capable of handling such values and deliver a compact, synthetic, and structured view of related products, and (b) the use of tools supporting to visualize, control, review and refinement of information in the PCM.

- **Lack of tracing variability.** Variability modeling approaches often do not address traceability between artifacts across problem and solution space. Traceability will improve the understanding of system variability, as well as support its maintenance and evolution. With large systems the necessity to trace variability from the problem space to the solution space is evident. Approaches for dealing with this complexity of variability need to be clearly established.

2.6 Conclusion

In this chapter we have surveyed several approaches in literature that are close related to the main contributions of this thesis. In particular, we reviewed several statistical techniques for mining knowledge from text and building variability models. We also presented the state of the art of synthesizing feature models from different artifacts and discussed their limitations.

In the next part, we present the contributions of this thesis in our two case studies and provide a comparison of these latters. In Chapter 3, we propose an approach to reverse engineering feature models from regulatory requirements in the nuclear domain. Chapter 4 describes our approach for synthesizing product comparison matrices from informal product descriptions. Finally, Chapter 5 provides a comparison, lessons learned and discussion regarding these two case studies.

Part II

Contributions

Chapter 3

Moving Toward Product Line Engineering in a Nuclear Industry Consortium

In this chapter, we instantiate our global contribution in the first case study to reverse engineering feature models from regulatory requirements in order to improve safety systems certification in the nuclear domain. Basically, the chapter contains two parts. In the first part, we formalize manually the variability in safety requirements and establish tracing variability with the architecture. This manual work is also a contribution that provides great value to industry partners and that introduces formal variability modeling in their engineering processes. Based on this experience, we have acquired solid knowledge about the domain of regulatory requirements, which is the basis to propose a meaningful automatic process for synthesizing feature models from these regulations. This constitutes the second part of this chapter. The general objective of the chapter is to adopt natural language processing and data mining techniques capable of extracting features, commonalities, differences and features dependencies from regulatory requirements among different countries. We proposed an automated methodology based on semantic analysis, requirements clustering and association rules to assist experts when constructing feature models from these regulations.

The chapter is organized as follows. Sections 3.1 and 3.2 present the context and discuss the lack of variability awareness of the nuclear industry. Sections 3.3 and 3.4 provide additional background. Section 3.5 gives an overview of the general approach. In Section 3.6, we formalize manually the variability in safety requirements and we motivate Section 3.7 which investigates the use of automated techniques to mine and model variability from these requirements. In Section 3.8, we address tracing variability between artifacts across problem and solution space. Section 3.9 describes the different techniques used to implement our method. Sections 3.10 and 3.11 successively present our case study and evaluate our proposed approach. Section 3.12 provides lessons learned and discussion. In Section 3.13, we discuss threats to validity.
The contributions of this chapter are published in [NSAB14].

3.1 Context

3.1.1 The CONNEXION Project

Since 2011, the CONNEXION[1] project is a national work program to prepare the design and implementation of the next generation of digital Instrumentation and Control (I&C) systems for nuclear power plants, with an international compliance dimension. The CONNEXION project is built around a set of academic partners (CEA, INRIA, CNRS / CRAN, ENS Cachan, LIG, Telecom ParisTech) and on collaborations between large integrators such as AREVA and ALSTOM, EDF and "technology providers" of embedded software (Atos Worldgrid, Rolls-Royce Civil Nuclear, Corys TESS, Esterel Technologies, All4Tec, Predict). For the specific concern of having a high level and global perspective on requirements and architecture variability modeling, the working group started on November 2013 and was constituted of CEA, Inria, AREVA, EDF, Atos Worldgrid, Rolls-Royce Civil Nuclear, and All4Tec engineers and researchers. In the group, Inria, CEA and All4Tec are considered as variability experts.

EDF Context. EDF (Electricité de France) is the national electricity provider in France and owns and operates a fleet of 58 nuclear power units. Besides the continuous maintenance and surveillance during operation, the systems of a unit may be replaced or upgraded during the periodic outages that are necessary for refueling and inspection. Such changes are under close regulatory scrutiny and subject to approval from the concerned safety authorities. Experience from the field, technological progress or societal evolutions may lead authorities to modify their expectations regarding the different systems. Consequently, they ask the licensee to justify the system's safety based on new or modified criteria. Safety justifications are grounded on technical arguments justifying technological choices, design decisions but also rely on practices that have been accepted in previous projects.

In France, EDF nuclear power units are built in series. The units of a series have the same general design, with relatively minor differences to take account of the specific site constraints. For example, the cooling systems need to take consideration of whether the unit is on the seashore, along a big river, or along a small river. There are currently four main series: the "900MW" series has 34 units, the "1300MW" series has 20 units, the "N4" series (1450MW) has 4 units, and the Evolutionary Pressurized Reactor (EPR) series has one unit still under construction. A large number of functions are necessary to operate a nuclear power unit, and guarantee its safety. Functions are categorized (A, B, C or Not Classified) based on their importance to safety, category A functions being the most important to safety, and Not Classified functions being those that are not important to safety. In parallel to functions categorization, the I&C systems are classified based on the categories of the functions they implement. As mentioned earlier, each system important to safety (i.e., implementing at least one category A, B or C function) goes through a safety justification process. In general, the safety authority lets EDF propose and present its solution and then decides whether the solution is acceptable or needs to be improved.

[1] `http://www.cluster-connexion.fr/`

3.1.2 Qualification of Safety Systems and National Practices regarding I&C Systems

Software systems designed to perform safety functions must conform to an increasing set of regulatory requirements. In the nuclear energy domain, a licensee must therefore demonstrate that his system meets all regulatory requirements of a regulator. These requirements can be contained in regulatory documents, in guides, standards and even in tacit knowledge [SGN11] acquired from anterior projects in the past. This lays applicants with a huge and increasing amount of documents and information.

This work takes its root on I&C systems in nuclear power plants. I&C systems include instrumentation to monitor physical conditions in the plant (e.g., temperature, pressure, or radiation), redundant systems to deal with accidental conditions (safety systems) and all the equipment for human operators to control the behavior of the plant. While digital components are replacing most of the older conventional devices in I&C systems, confidence in digital technologies remains low. Consequently, regulatory practice evolves and new standards appear regularly while domain expertise is heavily involved for certification.

In a quite recent history, answering to a nuclear industry motto: "to cope with complex safety problems, the simpler the solution is, the better the solution is", nuclear industry was used to utilize relays and conventional (not digital) technologies, which were simple enough to be used and qualified for complex and critical safety functions.

Digital systems have now become essential in all industries and these conventional components are not available anymore in the market and less and less specified for nuclear industry sole usage like COTS (Commercial Off-The-Shelf). Unfortunately, it represents a monumental effort to try to demonstrate, if feasible, the complete absence of error into these digital systems. The situation becomes worse while relating to some famous failures due to software during the last decades.

Based on their experience acquired from past or recent projects, regulators of each country have built a unique and specific practice related to nuclear energy and safety concerns. This section provides an overview of the regulatory requirements corpus related to safety in nuclear I&C systems. We focus on all the links that must be established in order to certify a system.

3.1.2.1 Operators and Regulations

Figure 3.1 gives an overview of the different kind of documents and actors involved in the safety assessment process for a candidate plant project. We detail this figure and illustrate it within the scope of digital I&C systems.

When licensees, like EDF, plan a project (realization of a new power plant, substitution of obsolete technologies in existing plants, renewal of an exploitation license), they rely on their experience acquired on past projects or take into account other existing projects. They may have issued technical codes to ease reusability along their different projects. They also rely on their engineering expertise to cope with complex emerging technical issues when innovation is required.

The proposed solution must comply with regulatory requirements. These requirements or recommendations are expressed in multiple documents: legal documents issued by national authorities; standards, issued by international organizations; regulatory practices, which arise from specific questions from regulators and following discussions. These different types of requirements, shown at the left and top of Figure 3.1, are detailed in the following.

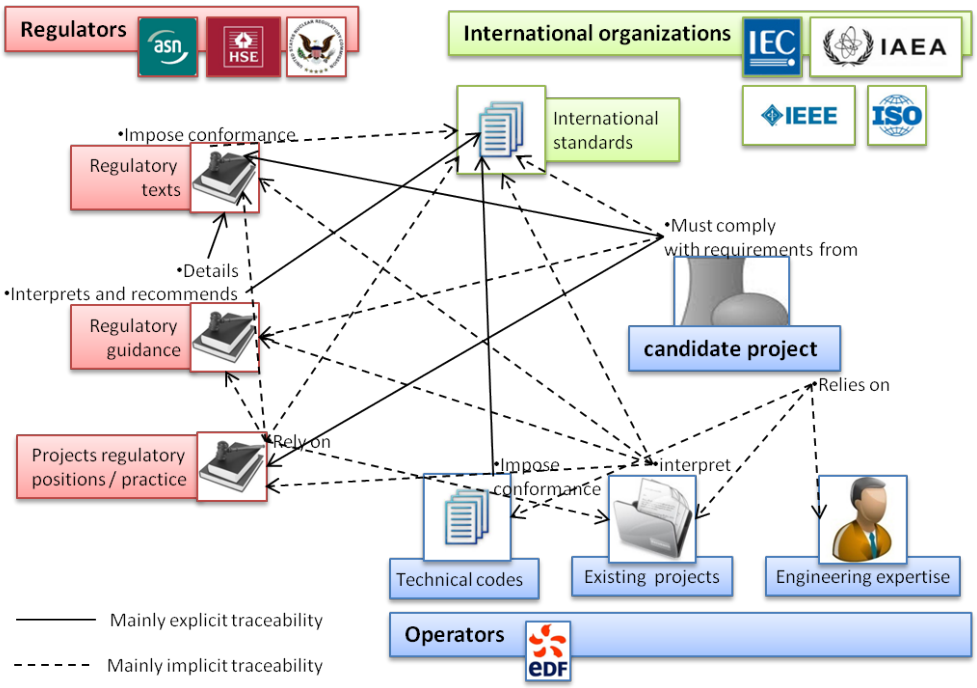


Figure 3.1: Overview of the nuclear regulatory landscape [SB12a]

3.1.2.2 Different Kind of Regulatory Texts

Regulatory Texts with Regulatory Requirements. Regulatory texts issued by public authority, express very high level requirements, principles or objectives related to people's life and environment protection, applicants responsibilities and duties. These texts do not provide guidance to achieve these requirements [SB12a]. In France, such documents and requirements are collected in the "Basic safety rules" documents (RFS II.4.1.a related to software). In the USA, they are expressed through the Code of Federal Regulations 10CFR50 and its appendices. In the UK, the requirements are collected in the "Safety Assessment Principles" (SAPs).

Regulatory Guidance. Regulatory guides describe the regulator's position and what he considers as an acceptable approach. These guides, endorse (or not) parts of standards and may provide interpretations of some specific parts. In France, there is no such document available. In the USA, the Nuclear Regulatory Commission (NRC) publishes regulatory guides such as the Regulatory Guide 1.168 for "Verification validation, re-

lease and audit for digital computer software used in safety systems". In the UK, one can find the Technical Assessment Guides (TAGs), for example the TAG 003 titled "safety systems" and 046 titled "Computer-based safety systems".

Regulatory Positions and Practice. During projects submissions, realizations, operations, maintenance, licensees still have to deal with regulators and issue documentations related to a specific project or installation. It can be the case for example for the renewal of an obsolete I&C system which raises a problem of qualification of a new device. This leads to regulatory positions while accepting or refusing propositions (for instance, the authorization of operation for ten more years for one reactor in France) or requiring improvements on specific topics. This is the most explicit highlight of the regulatory practice.

3.1.2.3 International Standards and Practice

International standards are state of the art propositions covering specific domains. It is important to notice that the requirements and recommendations in these standards are meant to be applied in a voluntary way, except when a regulator imposes or recommends its application. At this moment, standards requirements are considered as regulatory requirements. One other important aspect to consider is that different standards may exist to deal with the same subject. In Europe, nuclear actors mainly follow the IEC/IAEA corpus whereas in the US, IEEE/ISO standards are applied. These two corpora have been written independently from each other.

Information Sample from an IEC Standard.
6.2 Self-supervision
6.2.A The software of the computer-based system *shall* supervise the hardware during operation within specified time intervals and the software behavior (A.2.2). This is considered to be a primary factor in achieving high overall system reliability.
6.2.B Those parts of the memory that contain code or invariable data *shall* be monitored to detect unintended changes.
6.2.C The self-supervision *should* be able to detect to the extent practicable:

- Random failure of hardware components;
- Erroneous behavior of software (e.g. deviations from specified software processing and operating conditions or data corruption);
- Erroneous data transmission between different processing units.

6.2.D If a failure is detected by the software during plant operation, the software *shall* take appropriate and timely response. Those *shall* be implemented according to the system reactions required by the specification and to IEC 61513 system design rules. This may require giving due consideration to avoiding spurious actuation.
6.2.E Self-supervision *shall* not adversely affect the intended system functions.
6.2.F It *should* be possible to automatically collect all useful diagnostic information arising from software self-supervision.

3.1.3 (Meta)modeling Domain Knowledge and Requirements

The previous text box proposes a sample from the standard IEC60880. It illustrates the abstraction level of textual information we have to handle as well as the different characteristics highlighted in the previous section. Chapter 6 of the IEC60880 deals with software requirements and its section 6.2 deals with software self-supervision. It contains 6 main text fragments (listed from 6.2.A to 6.2.F). Fragment 6.2.A is considered as a requirement due to the presence of the word *shall*. It also makes a reference to annex A.2.2 section. The following sentence ("this is considered to be...software behavior"), as it is not in the same paragraph, as no *shall*/*should* keyword, is then considered as an information note relating to this requirement. Fragment 6.2.C is considered as a recommendation (missing *shall* and presence of *should*). Fragment 6.2.D is a multiple sentences requirement due to the double presence of *shall*. It references IEC61513 standard.

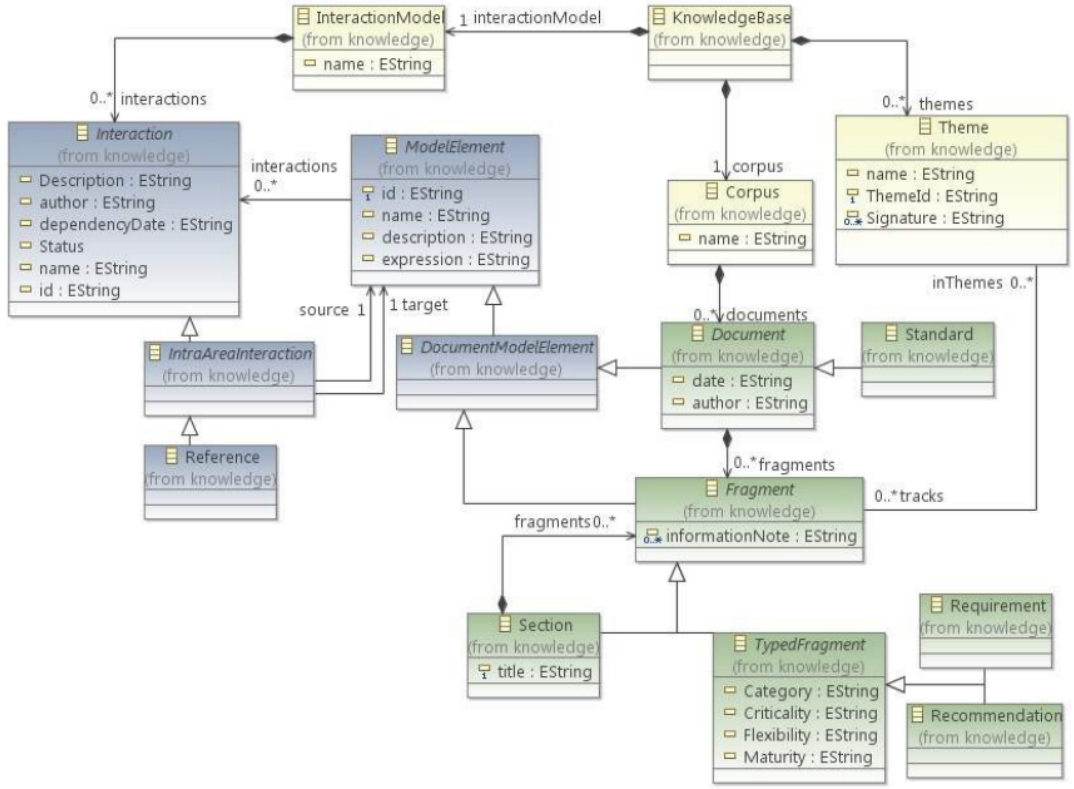


Figure 3.2: A Metamodel for Structuring Requirements Collections [SB12b]

Sannier *et al.* [SB12b] proposed a metamodel for a textual requirements collection which can offer the necessary canvas to understand the text-based business domain. In order to take into account traceability purposes, this initial structure is enriched with the necessary concepts to allow the representation of some traceability information such as rationales for a requirement or refinement information.

For instance, Sannier *et al.* manipulated here, at a coarse grain level, the different concepts of *standard* (the document itself), *section* (part of the document), *requirement*, and *recommendation* (leaves of part of a document), which are strong typing properties of different text fragments. They added an additional concept which is related to specific concerns clustering (such as "self-supervision") and that is encapsulated in the metamodel under the name "*Theme*". In the standard, requirement 6.2.D mentions another standard IEC61513, illustrates one explicit traceability link that is available within the text fragments and that has to be represented.

Figure 3.2 presents an excerpt of a metamodel that contains the minimal subset to formalize requirements in a multiple documents organization. Yet, it is worth noticing that instead of representing only requirements within a linear organization, the authors here represented a corpus of different kinds of documents, which contains different kinds of fragments such as structural groups (Section) or typed units (TypedFragment). This allows us not only to represent requirements, but to do so in a multi-level environment. For instance, the entire standard, or a section or requirements become a searchable artifact and can be handled at each of the three levels described.

3.2 Motivation and Challenges

3.2.1 On the Regulation Heterogeneity and Variability

Safety critical systems must comply with their requirements, where regulatory requirements are first class citizens. These requirements are from various natures, from regulations expressed by national and international bodies, to national explicit or implicit guidance or national practices. They also come from national and international standards when they are imposed by a specific regulator [SB14a].

In the specific context of nuclear energy, one applicant has to deal with very heterogeneous regulations and practices, varying from one country to another. This heterogeneity has a huge impact in the certification process as the regulators safety expectations, evidences and justification to provide can vary [SB12a, dlVPW13].

At this level, the main concern comes from the difference between national practices and the set of documents (regulatory texts and standards) to comply with. The nuclear industry has an unstable and growing set of safety standards. Worse, the set of safety standards is increasing within two main standards areas. On the one hand, there are the IEEE/ISO standards that are mainly applied in the US and eastern Asia. On the other hand, the IAEA/IEC standards and recommendations followed in Europe [SB12a]. This heterogeneity and lack of harmonization of the different nuclear safety practices has been highlighted by the Western Europe Nuclear Regulators Association (WENRA) in 2006 [RHW06].

Proposing one system, when having to perform a safety function, in different countries then leads to a huge problem of variability that concerns, not only the set of requirements to comply with and the certification process, but also the system's architecture itself.

3.2.2 Lack of Product Line Culture

Rise and Fall of the EPR Reactor Out of France. In France, EDF owns and operates 58 nuclear power units, following four different designs or series (same design but specific projects). Born from a European program, the EPR design represents the new power plant generation and has been expected to be built in several countries: France, Finland, the United-Kingdom, China, and later, in the USA.

The British safety authorities reference the same set of IEC standards as in France. However, their acceptable practices differ on some significant points and lead to differences in I&C architectures. In particular, safety approaches in the UK rely in large part on probabilistic approaches, whereas in France probabilistic analyses are considered as complementary sources for safety demonstration. Consequently, the safety justification for the same item has to be done twice, in two different ways. This example clearly highlights the gaps between different practices and the gaps between the possible interpretations of the same documents.

Now, EDF wishes to build EPR units in the USA. US authorities provide detailed written regulatory requirements and guidance (contrary to France were only very high level Basic Safety Rules are issued). Also, the standards endorsed by the US authorities are not the IEC standards cited earlier, but IEEE documents. In this case, this is not only a subset of requirements interpretation that will differ but the full content of the provided documents to support the different developments.

As a consequence, the concept of series that enabled to design and maintain the nuclear power plants in France can no longer be applied as such for export. Thus, since 2008, in the five most advanced EPR projects (construction in Finland, France and China, certification in progress in the USA and UK), EDF and Areva have been with four different I&C architectures and five different and ad hoc certification processes, specific to each country.

Conforming to Different Regulations. Comparing each IEC standard (and their interpretations) with its approximately relevant IEEE corresponding standard is difficult, time consuming and does not ensure to have the correct interpretation of the different standards. Though the domain owns a very precise and established vocabulary, ambiguities [Poh10] and interpretations are legions. Vocabulary: terms are not the same. IEC60880 speaks about activities and IEEE1012 considers tasks. Semantics: In the end, are we talking about the same thing when using "task" and "activity"?

Legal documents and standards contain intended and unintended ambiguity [BA07, Kam05b], causing interpretations, misunderstandings and negotiations between stakeholders to agree on a common definition. Scope of regulations may also differ as there is no direct mapping from one standard to another but many overlaps and differences. IEC60880 standard covers all the software lifecycle, whereas the IEEEstd1012 focuses only on software validation and verification and needs to be completed with other references. Though the task is very difficult, formalizing the requirements variability and finding the common core that will enable the next I&C architecture generation is more than necessary from the industrial perspective. In the context of the CONNEXION Project, a product line approach consists to define a generic foundation that is refined

for a given project by taking into account the specific requirements. This is an important challenge for building I&C systems on EPR units or other types of reactors in several countries in order to avoid the questioning of the initial design principles.

3.3 Common Variability Language (CVL)

The Common Variability Language (CVL) [Com] is a domain-independent language to specify and resolve variability over any instance of any language defined based on MOF (Meta-Object Facility[2]). We present here the three pillars of CVL and introduce some terminology that will be used in our approach: a variability abstraction model (VAM) expressing the features and their relationships; a variability realization model (VRM), containing the mapping relations between the VAM and the artifacts; the resolutions (i.e, configurations) for the VAM; and the base models conforming to a Domain-Specific Language (DSL).

Variability Abstraction Model (VAM) expresses the variability in terms of a tree-based structure. VAM retains the concrete syntax of feature model and supports different types such as choice (Boolean feature), classifier (feature with cardinality) and a constraint language for expressing dependencies over these types [CGR+12]. In the reminder of this chapter, we want to use only one terminology for the sake of understandability. Therefore, we adopt the **feature model (FM)** terminology and the well-known SPL engineering vocabulary while keeping the graphical notation of CVL.

Base Models (BMs) is a set of models, each conforming to a domain-specific modeling language (DSML). The conformance of a model to a modeling language depends both on well-formedness rules (syntactic rules) and business, domain-specific rules (semantic rules). The Object Constraint Language (OCL) is typically used for specifying the static semantics. In CVL, a base model plays the role of an asset in the classical sense of SPL engineering. These models are then customized to derive a complete product.

Variability Realization Model (VRM) contains a set of Variation Points (VP). They specify how features are realized in the base model(s). An SPL designer defines in the VRM what elements of the base models are removed, added, substituted, modified (or a combination of these operations) given a selection or a deselection of a feature in the VAM.

Having separate models for each concern favors modularization and reusability; this is a step towards *externalizing* variability from the domain language and *standardizing* it for any DSL.

3.4 Information Retrieval and Data Mining Techniques

Latent Semantic Analysis. Information Retrieval (IR) refers to techniques that compute textual similarity between different documents, relying on an indexing phase

[2] `http://www.omg.org/mof/`

that produces a term-based representation of the documents. If two documents share a large number of terms, those documents are considered to be similar [GF12].

Latent semantic analysis (LSA) is a technique in natural language processing, in particular in vectorial semantics, for analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms. A matrix containing term counts per document (rows represent unique terms and columns represent each document) is constructed. This matrix is then factorised by singular value decomposition (SVD). The reason SVD is useful, is that it finds a reduced dimensional representation of our matrix that emphasizes the strongest relationships and throws away the noise. In other words, it makes the best possible reconstruction of the matrix with the least possible information. To do this, it throws out noise, which does not help, and emphasizes strong patterns and trends, which do help. Documents are then compared by taking the cosine of the angle between the two vectors formed by any two columns. Values close to 1 represent very similar documents while values close to 0 represent very dissimilar documents.
Latent Semantic Analysis has many advantages:

1. First, the documents and terms end up being mapped to the same concept space; in this space we can cluster documents.

2. Second, the concept space has vastly fewer dimensions compared to the original matrix. Not only that, but these dimensions have been chosen specifically because they contain the most information and least noise. This makes the new concept space ideal for running further algorithms such as testing clustering algorithms.

3. Third, LSA can handle two fundamental problems: synonymy and polysemy. Synonymy is often the cause of mismatches in the vocabulary used by the authors of documents and the users of information retrieval systems.

4. Last, LSA is an inherently global algorithm that looks at trends and patterns from all documents and all terms so it can find things that may not be apparent to a more locally based algorithm.

Association Rules. The objective of association rule mining [CR06] is the elicitation of interesting rules from which knowledge can be derived. Those rules describe novel, significant, unexpected, nontrivial and even actionable relationships between different features or attributes [AK04], [JZ05].

Association rule mining is commonly stated as follows [AIS93]: Let $\{i_1,i_2,...,i_n\}$ be a set of *items*, and D be a set of transactions. Each transaction consists of a subset of items in I. An association rule, is defined as an implication of the form $X \rightarrow Y$ where $X, Y \subseteq I$ and $X \cap Y = \varnothing$. X and Y are called antecedent (left-hand-side or LHS) and consequent (right-hand-side or RHS) of the rule.

Support, confidence, Chi square statistic and the minimum improvement constraint among others might be considered as measures to assess the quality of the extracted rules [TSK06]. Support determines how often a rule is applicable to a given data set of

an attribute and it represents the probability that a transaction contains the rule. The confidence of a rule $X \rightarrow Y$ represents the probability that Y occurs if X have already occurred $P(Y/X)$; then it estimates how frequently items Y appear in transactions that contain X.

Chi square statistics combined with its test (see next section) might be used as a measure to estimate the importance or strength of a rule from a given set of transactions and by this way to reduce the number of rules [LHM99]. Finally the minimum improvement constraint measure not only indicates the strength of a rule but it prunes any rule that does not offer a significant predictive advantage over its proper sub-rules [BJAG99]. In this work for the process for obtaining rules, we consider the Apriori Algorithm [AIS93] that is supported on frequent itemsets and is based on the following principle:

"If an itemset is frequent, then all of it subsets must also be frequent" Conversely "If an itemset is infrequent, then all of it supersets must be infrequent to".

Chi Square and Independence Test. This test is based on Chi square value measure [LHM99], [MR10]. The measure is obtained by comparing the observed and expected frequencies, and using the following formula:

$$X^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

where O_i stands for observed frequencies, E_i stands for expected frequencies, and i runs from $1, 2, ..., n$, where n is the number of cells in the contingency table.

The value obtained in this equation is then compared with an appropriated critical value of Chi square. This critical value chi-square X_0^2 depends of the degrees of freedom and level of significance. The critical value chi-square X_0^2 will be calculated with $n-1$ degrees of freedom and α significance level. In other words, when the marginal totals of a 2 x 2 contingency table is given, only one cell in the body of the table can be filled arbitrarily. This fact is expressed by saying that a 2 x 2 contingency table has only one degree of freedom. The level of significance α means that when we draw a conclusion, we may be $(1-\alpha)\%$ confident that we have drawn the correct conclusion (Normally the α value is equal to 0.05). For 1 degree of freedom and a significance level of 0.05 critical value chi-square $X_0^2 = 3.84$.

The most common use of the test is to assess the probability of association or independence of facts [MR10]. It consists on testing the following hypothesis:

Hypothesis null H_0: The variables are independent.

Alternative hypothesis H_1: The variables are NOT independent.

In every chi-square test the calculated X^2 value will either be (i) less than or equal to the critical X_0^2 value OR (ii) greater than the critical X_0^2 value. If calculated $X^2 \leq X_0^2$ we conclude that there is sufficient evidence to say that cross categories are independent. If calculated $X^2 > X_0^2$, then we conclude that there is no sufficient evidence to say that cross categories are independent and we can think on dependency.

Cross Table Analysis. The cross table analysis [MR10] consists in a paired based comparison among the different features. Normally, it is represented as a n x n matrix that provides the number of co-occurrence between features. It is obvious that a conditional probability could be established by dividing the co-occurrence by the number of occurrence of a single feature.

3.5 Overview of the General Approach

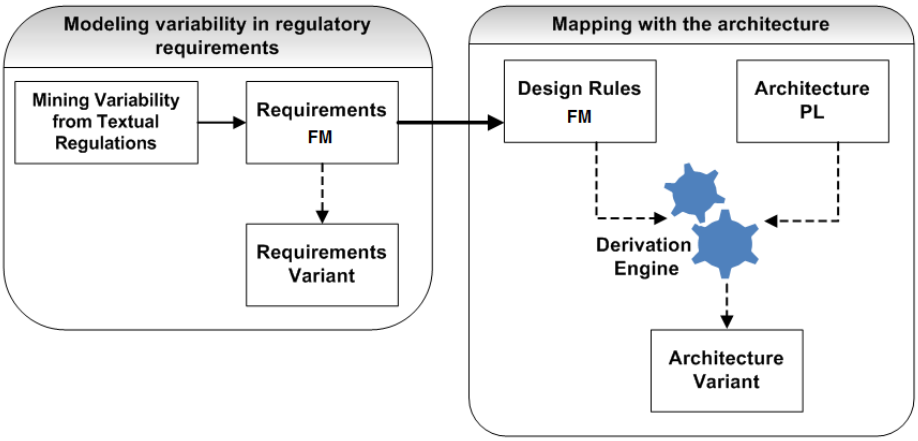


Figure 3.3: Overview of the General Approach

Figure. 3.3 depicts the approach we developed to tackle the variability issues [NSAB14]. The long term goal is to configure a robust I&C architecture from features related to regulatory requirements. The gap between textual regulatory requirements and the architecture is obviously important and variability cross-cuts both parts. Therefore the key idea is to exploit architectural design rules as an intermediate between the regulatory requirements and the architecture.

Intensive interactions between all the involving partners and numerous workshops and meetings lead to the adoption of the approach (see also Section 3.12 for more details). Two separate areas of variabilities are part of the approach (1) variability among regulatory requirements which represents our contribution, (2) variability among the architecture led by another partner. Regarding the requirements variability, it can take place at two levels: the variability of one particular requirement and the variability of a set of requirements within a product line. A first key task is to determine the variabilities within the set of requirements we want to satisfy. At the same time, the other key task is related to the adaptation of these variable elements by orchestrating the possible configurations from the architecture perspective.

The first stage aims at handling the multiple interpretations of ambiguous regulations using mining techniques. The second stage addresses the *impact* of requirements variability on the architecture and its certification through the variability in design rules. In Section 3.6, we formalize manually the variability in safety requirements. In

Section 3.7, we propose an approach to automatically synthesize feature models from these regulations. In Section 3.8, we establish traceability between variable requirements and variable architecture elements.

3.6 Handling Variability in Regulatory Requirements

In this section, we present how we manage variability with nuclear regulatory requirements and its modeling with the OMG Common Variability Language (CVL) [Com]. This domain is complex because of the variety of documents one has to handle; the number of requirements they contain; their high level of abstraction and ambiguity, etc. We proposed to analyze variability in regulatory documents with the smaller scope of topics (A topic is a concern within a corpus (e.g., "independence", "safety Classification", etc.)), on different corpora and on the same abstraction level: regulatory text, regulatory guidance or standards (see Section 3.6.3). Our industrial partners proposed to model variability in IEC and IEEE standards for each of these two topics: independence (see Table 3.1) and safety classification (see Table 3.2).

I&C Architecture Concepts. In order to ease the understanding of the following sections, we briefly describe the main concepts of a classic I&C architecture. An I&C architecture can be decomposed into systems that perform functions. Systems and functions are classified with respect to their safety importance. These systems and functions are organized within lines of defense (LDs) and many constraints drive the organization of the architecture in order to prevent common cause failures. These constraints mainly deal with communication or independence (physical separation and/or electrical isolation) between lines of defense or systems with respect to their safety classification.

3.6.1 Requirements Similarity Identification

The first step of Figure 3.3 is based on the intuition that features are made of clusters of related requirements. In order to form these clusters, requirements are considered related if they concern similar matters. Thus, the subject matter of the requirements has to be compared, and requirements with similar subject matter will be grouped. For example, in Table 3.1, the following safety requirements IEC 60709.1, IEC 60709.11, IEC 60709.12, IEEE 384.1 and IEEE 384.5 are similar because all of them are addressing the independence between systems. In particular, IEC 60709.1, IEC 60709.11 and IEC 60709.12 are dealing with preventing system degradation, while IEEE 384.1 and IEEE 384.5 specify how this must be achieved.

3.6.2 Requirements Clustering

The requirements clustering step creates a feature tree based on the similarity measures from the previous stage. Requirements which are semantically similar, i.e., have the most in common, are "clustered" to form a feature. These smaller features are then clustered with other features and requirements to form a parent feature. To return to our previous examples of Section 3.6.1, in standards, IEC 60709.1, IEC 60709.11,

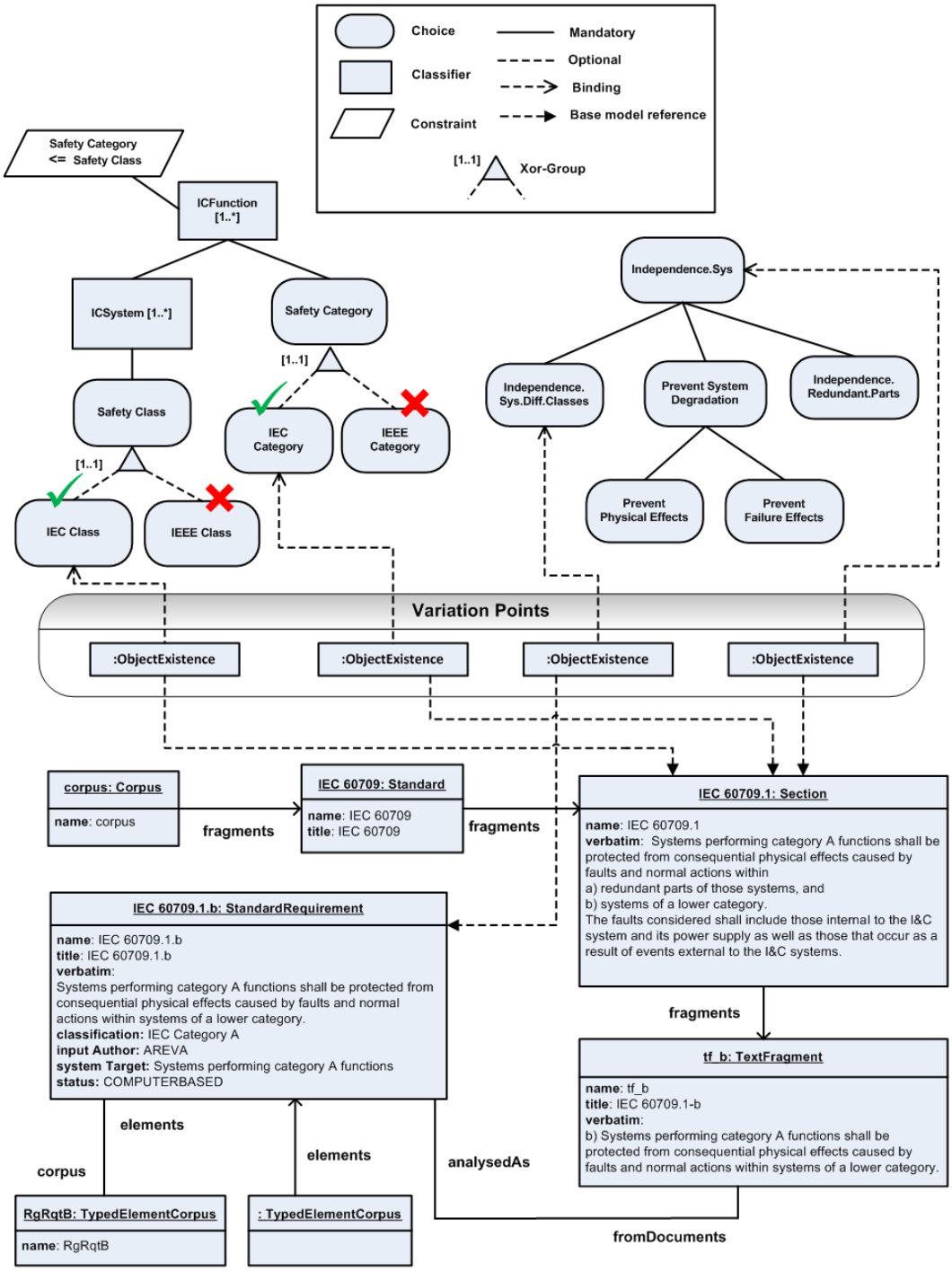


Figure 3.4: Mapping between standards BM and standards FM

Table 3.1: Mining variability in *Independence* topic

Information sample from IEC and IEEE standard		Design Rules		Countries
Index	Verbatim	Index	Rule	
IEC 60709.1	Systems performing category A functions shall be protected from consequential physical effects caused by faults and normal actions within a) redundant parts of those systems, and b) systems of a lower category.	SA10	A lower classified system can not send information to a higher classified system or at least it should not disturb any of these features.	France and UK
IEC 60709.11	Failures and mal-operations in the non-category A systems shall cause no change in response, drift, accuracy, sensitivity to noise, or other characteristics of the category A system which might impair the ability of the system to perform its safety functions.			
IEC 60709.12	Where signals are extracted from category B or C systems for use in lower category systems, isolation devices may not be required; however, good engineering practices should be followed to prevent the propagation of faults.	SA12	A higher classified system can not directly send information to a lower classified system.	France and UK
IEEE 384.1	Physical separation and electrical isolation shall be provided to maintain the independence of Class 1E circuits and equipment so that the safety functions required during and following any design basis event can be accomplished.	SA54	No communication between systems with different classes.	US
IEEE 384.5	1) Non-Class 1E circuits shall be physically separated from Class 1E circuits and associated circuits by the minimum separation requirements specified in 6.1.3, 6.1.4, ... 2) Non-Class 1E circuits shall be electrically isolated from Class 1E circuits and associated circuits by the use of isolation devices, shielding, and wiring techniques or separation distance.			

Table 3.2: Mining variability in *Safety Classification* topic

Information sample from IEC and IEEE standard		Design Rules		Countries
Index	Verbatim	Index	Rule	
IEC 60964.11	The design basis for information systems, including their measurement devices, shall take into account their importance to safety. The intended safety function of each system and its importance in enabling the operators to take proper pertinent actions ...	SA5	Every system and sensor is associated with safety class.	US, France and UK
IEC 61513 .3	d) Each IC system shall be classified according to its suitability to implement IC functions up to a defined category.	SA8	Function with safety category n can be allocated only on systems of safety classes n or >n.	US, France and UK
IEC 61226.18	There shall be adequate separation between the functions of different categories.	FA11	A lower classified function can not send information to a higher classified function.	US, France and UK
IEC 61226.3a	An IC function shall be assigned to category C if it meets any of the following criteria and is not otherwise assigned to category A or category B: a) plant process control functions operating so that the main process variables are maintained within the limits assumed in the safety analysis not covered by 5.4.3 e).	SA57 SA58	The FA6 function associated category B. The FA5 function associated category A.	France

Table 3.3: Identification of features from IEC and IEEE standards and design rules

Features	IEC 60709.1	IEC 60709.11	IEC 60709.12	IEEE 384.1	IEEE 384.5	IEC 60964.11	IEC 61513.3	IEC 61226.18	IEC 61226.3a
ICSystem	✓	✓	✓	✓	✓	✓			
ICFunction	✓	✓	✓		✓	✓	✓		
Safety Classes	✓	✓	✓	✓	✓	✓			
IEC Classes	✓	✓	✓						
IEEE Classes				✓	✓				
Safety Categories	✓	✓	✓			✓	✓	✓	
IEC Categories	✓	✓	✓						✓
IEEE Categories									
Independence. Sys.	✓	✓	✓	✓	✓				
Independence. Sys.Diff.Classes		✓	✓	✓	✓				
Independence. Redundant. Parts	✓								
Independence. Func. Diff.Categories							✓		
Prevent System Degradation	✓	✓	✓						
Prevent Physical Effects	✓								
Prevent Failure Effects		✓							
Communication Separation Physical Separation Electrical Isolation	✓	✓	✓	✓	✓		✓		

Features	SA5	SA8	SA10	SA12	SA54	FA11	FA13
Safety Classes	✓	✓	✓	✓	✓		
Sender Class			✓	✓			
Receiver Class			✓	✓			
Safety Categories					✓		
Sender Category					✓		
Receiver Category						✓	
Sender LD							✓
Receiver LD							✓
Separation.Sys. Diff.Classes		✓	✓	✓			
Separation.Func. Diff.Categories					✓		
Separation .Func.Diff.LD							✓
Communication. Without. Perturbation		✓	✓				
No.Communication. Sys.Diff.Classes				✓			

IEC 60709.12, IEEE 384.1 and IEEE 384.5 are clustered to form *Independence between Systems*. IEC 60709.1, IEC 60709.11 and IEC 60709.12 are clustered to create *Prevent System Degradation* feature, while IEEE 384.1 and IEEE 384.5 are clustered to give *Electrical Isolation* feature. Table 3.3 reports the global traceability between identified features and standards requirements.

3.6.3 Modeling Regulatory Requirements Variability

We propose to illustrate the complexity of safety requirements corpus through the manual search of similar requirements dealing with similar matter. As a reminder, this requirements analysis will be made on three different corpora, France, UK and US and on two standards: IEC and IEEE for each of these two topics: independence (see Table 3.1) and safety classification (see Table 3.2). Figure 3.4 shows an extract from the requirements model and its related feature model. Standards and regulatory texts concerns can be organized as variably concepts and properties like *ICFunction*, *Independence between Systems* (see Figure 3.4), *Independence between Functions* and *Communication Separation* (see Figure 3.6) which correspond to mandatory features.

In Figure 3.4, *ICSystem* and *ICFunction* are two classifiers having an instance multiplicity [1..*] (i.e., at least one instance of *ICSystem* and *ICFunction* must be created). Each *ICSystem* is associated with a *Safety Class* (See IEC 60964.11 in Table 3.2) and each *ICFunction* is associated with a *Safety Category*. Each *ICFunction* is allocated to at least one *ICSystem* while *Safety Category* must be lower or equal to *Safety Class*. See the OCL constraint attached to *ICFunction* and IEC 61513.3 in Table 3.2.

There are two alternatives for *Safety Class*: *IEC Class* and *IEEE Class* form an *Xor-group* (i.e., at least and at most one feature must be selected). Similarly, *IEC Category* and *IEEE Category* form two alternatives of *Safety Category*. *Independence between Redundant Parts*, *Independence between Systems of Different Classes* and *Prevent System Degradation* are mandatory child features of *Independence between Systems*. On the other hand, in Figure 3.6, *Independence between Functions of Different Categories* (See IEC 61226.18 in Table 3.2) and *Independence between Functions of Different Lines of Defense* are two mandatory child features of *Independence between Functions*.

As mentioned earlier in Section 3.1.3, Sannier and Baudry [SB14a] proposed a formalization of nuclear regulatory requirements into a requirements model using Domain specific languages (DSLs). We rely on this DSL in our work. Yet, it is worth noticing that instead of representing only requirements within a linear organization, they represent a corpus of different kinds of documents, which contains different kinds of fragments with different semantics.

Figure 3.4 depicts an excerpt of a standards BM that contains the minimal subset to formalize IEC 60709.1 requirement. From IEC 60709 standard, we present some transformation elements into text fragments and the traceability to requirements that are created and will be the analyzed elements. Moreover, this figure illustrates bindings between standards BM and the standards FM. For instance, the "object existence" variation points against the IEC 60709.1 *Section* refer to *Independence between Systems*, *IEC Class* and *IEC Category* meaning it will exist only when these features are selected.

The "object existence" variation point against the IEC 60709.1.b *Standard Requirement* is bound to *Independence between Systems of Different Classes.*

3.7 Automating the Construction of Feature Model from Regulations

In section 3.6, we performed a manual formalization of variability from a subset of 142 safety requirements. The input dataset was provided by the industrial partners to implement our proposed method (see more details in section 3.10). Yet, this activity becomes impossible and impractical as the number of considered requirements grows especially as we are dealing with large regulatory documents. To resolve the variability issue, an automatic support is essential to assist the domain expert during the construction of a feature model. Manually handling variability in safety requirements allowed us to develop solid knowledge about the domain of regulatory requirements. Based on this experience, given a set of regulatory documents in different countries, we propose a meaningful automatic process for synthesizing a feature model.

In this approach, regulations are first analyzed in each country, and a set of corresponding product models are built. Then, the process guides the construction of the feature model by detecting structural dependencies (candidate parent-child relationships, mandatory and optional relationships) and transversal dependencies such as requires and excludes. The domain of statistics provides several mining techniques that could be used to support this process [CZZM05, MKS06, AOAB05]. The research challenge was thus to identify which techniques could be used to efficiently detect the target items at each step of the method. Our research strategy was to experiment the available techniques on a real case. Once a technique was detected, further work was needed to identify with which parameter it should be used (e.g. thresholds). Last the overall method have been tested in a case study carried out in nuclear power plants. Sections 3.10 and 3.11 successively present our case study and evaluate our proposed approach. The findings are: (1) cross table analysis can be used to determine exclude relationships; (2) association rules analysis allows the retrieval of mandatory and optional relationships; (3) chi-square independence test combined with association rules are an effective way to identify require relationships;

3.7.1 Overview of Feature Model Synthesis

The approach combines semantic analysis, requirements clustering and mining techniques to assist experts when constructing feature models from a set of regulations. The overall process of this approach is depicted in Figure 3.5. The process starts with a collection of regulatory requirements $(R_1, R_2, ..., R_n)$ applied in different countries. The output is a single product line model specified with feature notation.

There are two main stages in this process: the construction of product models (PMs) (steps ❶ to ❺) and the construction of the feature model (steps ❻ to ❽). In the first stage, regulatory documents in each country are analyzed individually. For each of them

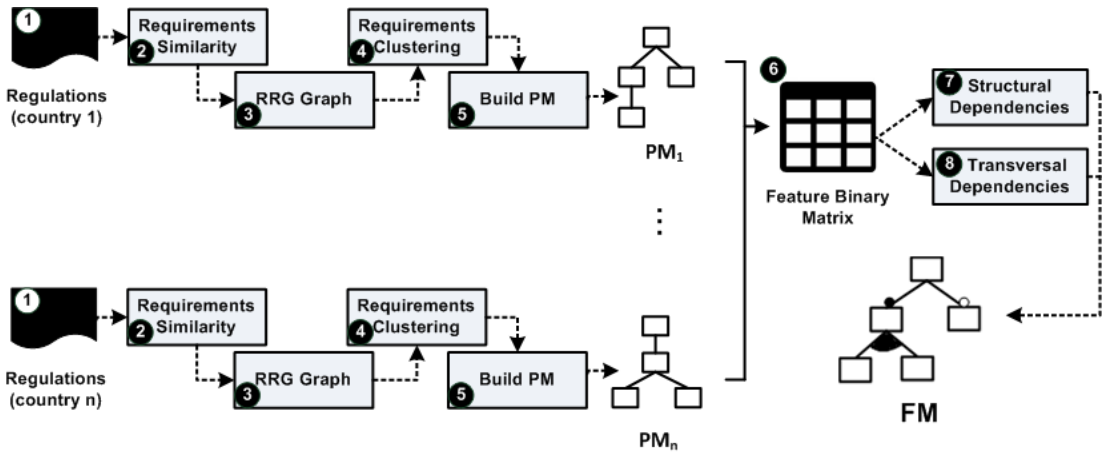


Figure 3.5: Feature Model Synthesis

there are five steps.

Step ❶ is to elicit individual requirements. In step ❷, we compute requirements similarity using Latent Semantic Analysis (LSA). Step ❸ is to model individual requirements and the relationships between them in an undirected graph, called Requirements Relationship Graph (RRG). In step ❹, we identify and organize features by applying the clustering algorithm in RRG which has previously been shown to perform well for requirements clustering [CZZM05]. The underlying idea is that a feature is a cluster of tight-related requirements, and features with different granularities can be generated by changing the clustering threshold value. In step ❺, we build the product model hierarchy.

The second stage takes the resulting product models and builds the feature model. The method starts by transforming the product models into a feature binary matrix (step ❻). This step consists in identifying all the possible features and highlighting their presence in product models. Then, the process guides the construction of the general tree architecture by detecting candidate parent-child dependencies (step ❼) and guides the identification of transversal dependencies such as requires and excludes (step ❽).

3.7.2 Requirements Relationship Graph

To perform regulatory requirements extraction (step ❶), we have adopted a configurable parser proposed by [SB14b] that uses, for each requirement document, a set of regular expressions that defines the parsing rules to determine the different fragments types such as requirements and recommendations while reading the input file. After individual safety requirements are elicited for a given country, the requirements relationship graph is built (step ❸).

Assume there are n requirements, they and their relationships are modeled as an undirected graph $G = (V, E)$, in which $V = \{R_i \mid R_i \text{ is an individual requirement}, 1 \leq i \leq n\}$, and $E = \{E_{ij} \mid E_{ij} \text{ is the relationship between requirements } R_i \text{ and } R_j, 1 \leq$

$i, j \leq n\}$. The key point is to determine the weight of each edge E_{ij} to express the strength of the relationship between requirements R_i and R_j. We adopt as quantification strategy the semantic similarity between requirements using LSA (step ❷). Among many such schemes of term indexing, LSA has been shown to be able to filter noisy data and absorb synonymy i.e. the use of two different terms that share the same meaning, and polysemy i.e. the use of a single term to mean to distinct things, in large corpus [DDF+90, Dum93, Dum95, BB05]. LSA was widely used in document-based mining. In our case, documents are regulatory requirements. A matrix containing term counts per requirement is constructed. Rows represent unique terms and columns represent each requirement. The basic derivation of LSA is as follows. Let X be the term by requirement matrix:

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix}$$

$t_i^T = \begin{bmatrix} x_{i,1} & \cdots & x_{i,n} \end{bmatrix}$ is the occurrence vector of term i, and $d_j = \begin{bmatrix} x_{1,j} & \cdots & x_{m,j} \end{bmatrix}$ is the vector of requirement j. The dotproduct $t_i^T t_p$ then gives the correlation between terms, and matrix XX^T contains all of the correlations. Likewise, $d_j^T d_q$ represents the correlation between requirements, and matrix $X^T X$ stores all such correlations. Singular Value Decomposition (SVD) is applied to X to produce three components:

$$X = U \Sigma V^T$$

where U and V are orthonormal matrices and Σ is a diagonal square. Requirements are similar if they contain semantically similar terms. Requirements are then compared by taking the cosine of the angle between the two requirements vectors. Values close to 1 represent very similar requirements while values close to 0 represent very dissimilar requirements.

3.7.3 Requirements Clustering

After building the requirements relationship graph, we apply requirements clustering (step ❹) in this graph to identify and organize features. The underlying idea is that a feature is a cluster of tight-related requirements, and features with different granularities can be generated by changing the clustering threshold value t. If there is an edge between two requirements and its weight is greater than or equal to t, they will be put in the same cluster. So the edges whose weights are above or equal to the threshold value are set to be valid; otherwise, the edges are invalid. Then connected components are

computed by the valid edges. Each connected component is a cluster of tight-related requirements sharing the same concern which represents the feature. As we decrease the threshold value, more edges are set to be valid, and we get clusters with coarser granularity. In our context, we choose empirically two sorted and distinct edge weight as clustering thresholds t_1 and t_2 between 0 and 1, since our industrial partners aim to construct simple and readable hierarchy (see more details in the next section).

3.7.4 Build the Product Model Hierarchy

All the individual requirements are placed in the lowest level of the tree, and they are the features with the finest granularity. For each clustering threshold, the generated features are put at the corresponding level: the lower is the threshold value, the higher is the level of the tree that contains the corresponding clusters. The features (clusters) in successive levels of the tree are explored. If a lower-level feature is a subset of another higher-level feature, we build the refinement relationship between them; if they are identical, we only reserve one of them (step ❺).

After all the levels are examined, all the refinement relationships are built. The resultant hierarchy structure may need to be adjusted according to domain knowledge. We may add features or remove features. After the adjustment, we should examine whether there are two or more trees. If so, we add an artificial root node as the parent node of their root nodes. Thus, we get a product feature tree (a product model) for a specific country. In the tree, each node is a feature, and all the features are organized by refinement relationships. But the tree contains no information about variability; how to model variability is the topic of the next section.

3.7.5 Variability Modeling

Running Apriori Algorithm. To obtain rules, we use in this work the Apriori Algorithm [AIS93] that is supported on frequent itemsets. For the purpose of this work items will be considered as features and transactions as product models, and the result of this pair wise is what we call the feature binary matrix (step ❻). The feature takes the value 1 if it is present in a product model and zero otherwise.

Once the binary feature matrix is built, we have the input to apply the association rule data mining tool. In fact the most complex task of the whole association rule mining process is the generation of frequent itemsets (in this part an itemset is considered as feature set) [LMSM10]. Many different combinations of features and rules have to be explored which can be a very computation-intensive task, especially in large databases. By setting the parameter association rule length equals to 1 for the Apriori

algorithm [AIS93], we can study only singles relations between features to avoid those computation complexities. Often, a compromise has to be made between discovering more complex rules and computation time. To filter those rules that might be not valuable, it is important to calculate its support. As we have already seen, the support determines how frequent the rule is applicable to the product P. This value compared with the minimum support accepted by a user (min Support threshold), prunes the uninteresting rules; those that may be not aggregate some value to knowledge.

To evaluate the relevance of the inference performed by a rule, we compute a confidence metric. The task is now to generate all possible rules in the frequent feature set and then compare their confidence value with the minimum confidence (which is again defined by the user). All rules that meet this requirement are regarded as interesting. Furthermore the calculation of other measures is relevant to refine the process of selecting the appropriate association rule. For that we propose to calculate the Improvement and chi-square measures. Those measures indicate the strength of a rule. In previous section we have already discuss about Chi-square statistic. Now we will consider the minimum improvement constraint because it prunes any rule that does not offer a significant predictive advantage over its proper sub-rules.

Mandatory and Optional Relationships. This section illustrates how we identify structural dependencies between features (step ❼). Removing all association rules that do not satisfy the minimum improvement constraint, offer us the most relevant rules available for the study; those ones with a significant predictive value. It is obvious that those relationships that are always present in all the product models may be considered as mandatory. Now, if some ambiguous information is present in the database and this one is not reliable at $\lambda\%$, in order to obtain mandatory relationships, the analyst may establish as a minimum confidence threshold the value $(100 - \lambda)\%$. Those rules whose support is greater than the $(100 - \lambda)\%$ may be considered as mandatory relationships.

In another hand bidirectional rules such as $F_1 \rightarrow F_2$ and $F_2 \rightarrow F_1$ may be also considered as mandatory relationships [BT06]. The relationship is classified as mandatory if at least one of the two properties mentioned before (high frequent features and bidirectional rules) occurs and, of course, the relationships belong to a parent child. Once parent child and as well mandatory relationships are identified the remaining parent child relationship may be classified as optional.

In the following we describe how we identify transversal dependencies (excludes and requires relationships) between features (step ❽).

Exclude Relationships. Feature cross table display relationships between features.

Let $F = \{F_1, F_2, ..., F_n\}$ be a set of n features. F x F can be represented as a n x n cross table describing the joint occurrence between the feature i and j. When the joint distribution of (F_i, F_j) for all $i \neq j$ is equal to zero, that can be interpreted that there is no probability that F_i and F_j may occur at the same time. Thus, they are mutually exclusive and the relationship between F_i and F_j is considered as an exclude relationship.

Requires Relationships. To identify requires relationships it is necessary to apply a chi-square independence test. The test is performed for each single rule with 1 degree of freedom in order to prove with a significance level $\alpha = 0.05$ that the relationships between non parent-child features F_i, F_j for all $i \neq j$ are independent or not. Thus, the association between F_i, F_j for all $i \neq j$ is considered as dependent if the X^2 value for the rule with respect to the whole data exceeds the critical $X^2 = 3.84$ (X^2 critical value with one degree of freedom and a significance level $\alpha = 0.05$). In another hand the association between F_i and F_j for all $i \neq j$ is considered as independent if the X^2 value for the rule with respect to the whole data does not exceed the critical $X^2 = 3.84$.

3.8 Traceability between Requirements and Architecture

In this section, we address tracing variability between artifacts across problem and solution space to investigate the robustness of the derived architecture against regulations variability. Indeed, we provide a variability-aware bridging of these two levels of abstraction through modeling variability in design rules [NSAB14]. Section 3.8.1 handles modeling variability in design rules using CVL, Section 3.8.2 addresses tracing variability between requirements and design rules and Section 3.8.3 illustrates the mapping between variable design rules and variable architecture elements to derive a complying architecture.

3.8.1 Modeling Variability in Design Rules

Design Rules to Bridge Requirements and Architecture Elements. Modeling requirements variability is useful, however there is no direct mapping from requirements to the architecture. To brige the gap between textual regulatory requirements and the architecture, we move towards variability in design rules.

Design rules, edited by EDF and endorsed by the French safety authority, are intermediate elements to bridge the gap between an architecture and the regulatory or normative requirements. Our industrial partners rely on these rules to validate the architecture against regulations. A design rule can satisfy fully or partially one or more

requirements: in Table 3.1 SA10 (resp. SA54) completely satisfies IEC 60709.1 and IEC 60709.11 (resp. IEEE 384.1 and IEEE 384.5).

Identifying and Modeling Variability in Design Rules. Similarly to requirements, the identification of features in design rules consists in comparing the subject matter of rules followed by a clustering step. For instance, SA10, SA12 and SA54 are similar because they are dealing with the separation of systems of different classes. In particular, SA10 and SA12 deal with communication without perturbation between systems of different classes whereas SA54 forbids the communication between them (see Table 3.1). Table 3.3 reports the traceability between identified features and design rules.

Comparing design rules interpretations in the three countries leads to the variability specification in Figure 3.7. The concept of design rules are decomposed into the following mandatory features: *ICFunction*, *Communication Separation* in Figure 3.6 and the two kinds of communication: *Functions Communication* (communication between functions) and *Systems Communication* (Communication between Systems). Similarly in standards FM, each *ICFunction* is allocated to at least one *ICSystem* and only one *Line of Defense*.

France and UK allow the communication between systems of different classes only if it will not cause systems perturbation using isolation devices (see SA10 and SA12 in Table 3.1), however USA forbids it (see SA54 in Table 3.1). In Figure 3.7, *Communication without Perturbation* and *No Communication between Systems of Different Classes* are two alternatives for *Separation between Systems of Different Classes*. Moreover, *decouplingType* is an optional classifier of *Systems Communication*. The latter has an OCL constraint written in its context comparing the *Sender Class* and the *Receiver Class*. If the *Sender Class* is lower than the *Receiver Class*, it requires isolation: the function non Empty() is used to state that there is at least one instance of the *decouplingType* classifier.

The three countries forbid the communication from lower to higher classified functions (see FA11 in Table 3.2). An OCL constraint is attached to *Functions Communication*, requiring that the *Sender Category* must be higher or equal than *Receiver Category*. Furthermore, a function allocated to a line of defense shall not communicate with a function allocated to another line of defense. Consequently, a second OCL constraint, attached to *Functions Communication* is added.

3.8.2 Mapping Between the Standards FM and the Design Rules FM

Since design rules represent intermediate elements between the requirements and the architecture, the design rules FM acts like a pivot between the standards FM and the architecture variability model. Figure 3.6 depicts two extracts from both standards FM and design rules FM; and also the mapping between them. For instance, *Separation between Systems of Different Classes* in design rules FM is related to *Independence between Systems of Different Classes* in standards FM. This mapping is due to the fact that SA10 satisfies IEC 60709.1 and IEC 60709.11 and SA54 satifies IEEE 384.1 and IEEE 384.5, at the same time, SA10 and SA54 are related to *Separation between Systems of Different Classes* (see Table 3.3 right-hand side) while IEC 60709.1, IEC 60709.11, IEEE 384.1 and IEEE 384.5 refer to *Independence between Systems of Different Classes* (see Table 3.3 left-hand side).

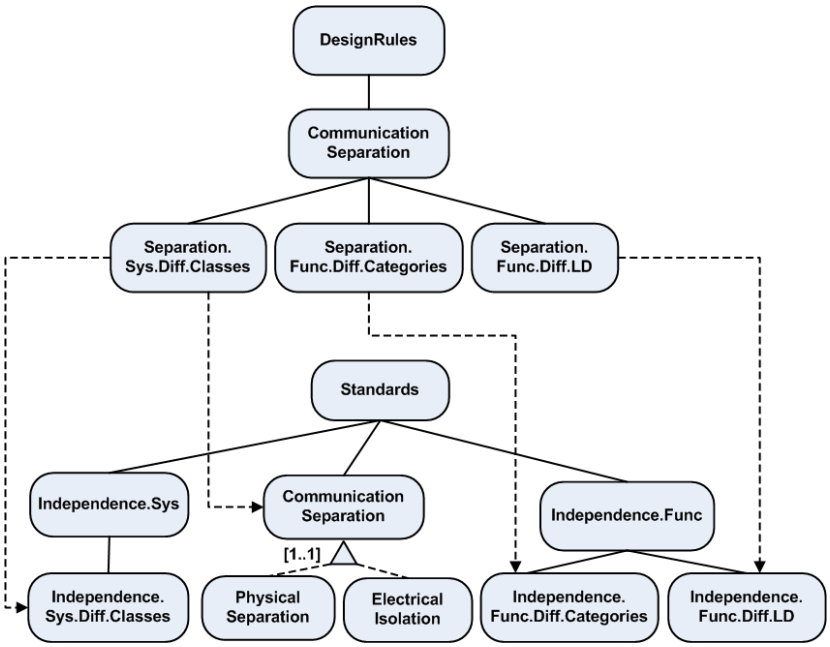


Figure 3.6: Mapping between the standards FM and the design rules FM

3.8.3 Mapping Between the Design Rules FM and the I&C Architecture

An architecture metamodel is defined by one partner: the CEA, based on the different elements that characterize an I&C system of a nuclear power plant through a SysML profile. CEA uses the SysML modeler Papyrus with the Sequoia add-on for managing

I&C architecture product line (see Figure 3.8). As mentioned earlier, industrial partners rely on design rules to validate the architecture against regulations. However, they do it for each derived architecture. Thus, we propose to consider both of the design rules FM and the architecture variability model during the derivation of a particular architecture.

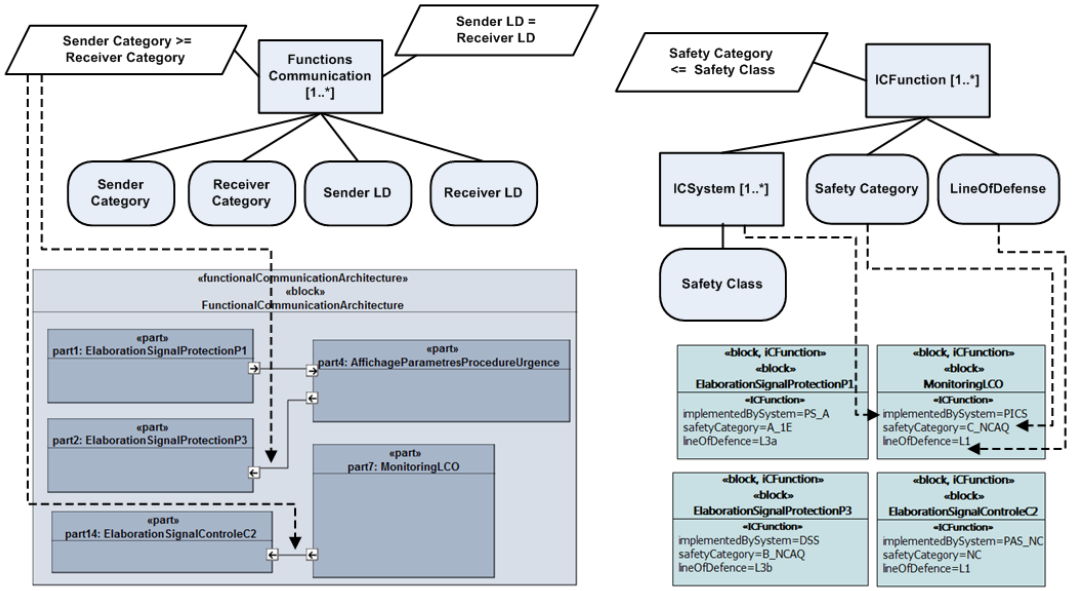
Figure 3.8 shows the binding between the architecture product line model and the corresponding feature model and Figure 3.7 illustrates the impact of the design rules FM on the derived architecture. For instance, if we select *Communication Without Perturbation* in Figure 3.7b, then we allow the communication between systems of different safety classes. Yet, the communication from a lower classified system to a higher classified system requires isolation: *decouplingType* (see OCL constraint). As shown in this figure, *System communication architecture* block in the derived architecture contains the following links: 1. from higher to lower classified system :from *SICS* (Class 1) to *DSS* (Class 2). 2. from lower to higher classified system: from *PICS* (Class 3) to *RCSL* (Class 2) by isolation means. 3. between equal classified systems: from *PS_A* (Class 1) to *SICS*.

Considering the two OCL constraints in Figure 3.7a left-hand side, we forbid the communication between functions of different lines of defense or from lower to higher classified function. As a result, *Functional communication architecture* block in the derived architecture contains a link from *Monitoring LCO*(Category: C_NCAQ and Line of Defense: L1) to *Elaboration signal control C2*(Category: NC and Line of Defense: L1).

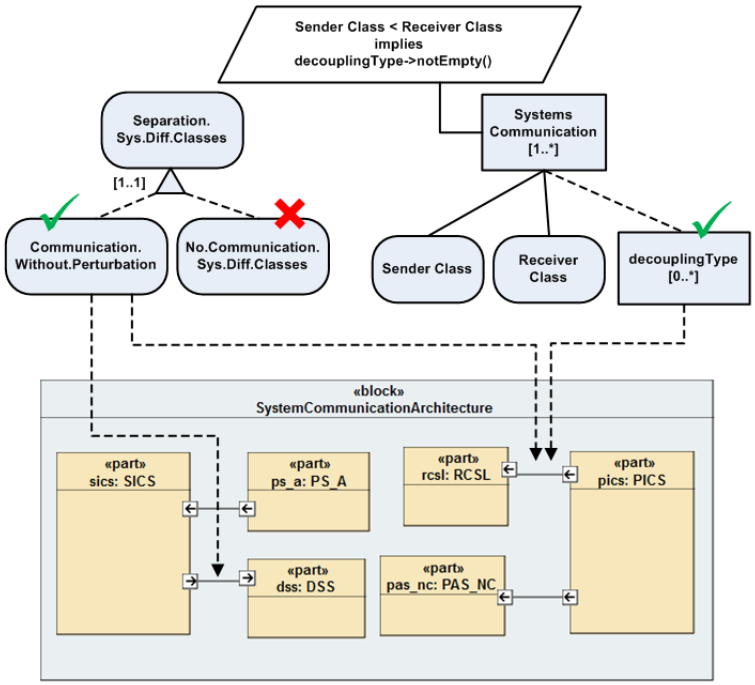
3.9 Implementation

In this section, we describe the different techniques used to implement our methodology including the automatic process to synthesize the feature model; and the manual formalization of variability in regulations and its traceability with the architecture.

Automatic construction of feature model from requirements. To compute similarity using Latent semantic analysis (LSA), we use the S-Space Package, an open source framework for developing and evaluating word space algorithms [JS10]. The S-Space Package is a collection of algorithms for building Semantic Spaces as well as a highly-scalable library for designing new distributional semantics algorithms. Distributional algorithms process text corpora and represent the semantic for words as high dimensional feature vectors. These approaches are known by many names, such as word spaces, semantic spaces, or distributed semantics and rest upon the Distributional Hypothesis: words that appear in similar contexts have similar meanings. Mining as-



(a) Mapping of Design Rules FM with Functions Communication and Functions Decomposition



(b) Mapping Design Rules FM with Systems Communication

Figure 3.7: Mapping between Design Rules FM and I&C Architecture

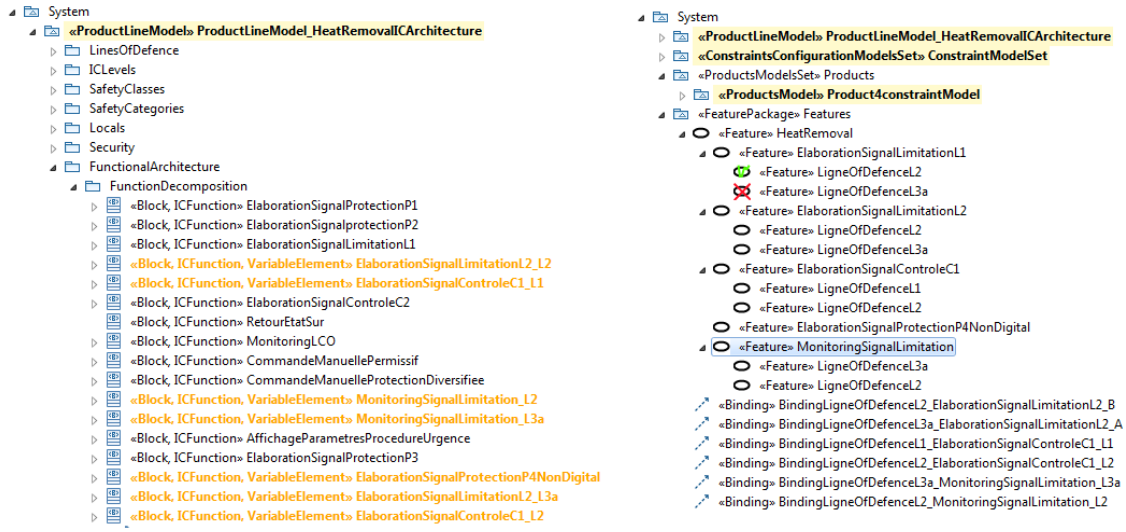


Figure 3.8: I&C Architecture PL (Sequoia)

sociation rules and computing the statistical metrics (Support, Confidence, Chi square, Improvement) are implemented using R scripts.

Manual formalization of variability in requirements and its traceability with the architecture. As mentioned earlier, we use CVL language and tool support for modeling variability in requirements and design rules, while CEA addresses modeling variability in the architecture using Sequoia tool. The goal of the Sequoia approach, developed by the CEA LIST, is to help designers to build product lines based on UML /SysML models [DSB05]. Variability in Sequoia is defined through a UML profile [TGTG05]. To specify an optional element, the designer simply adds the stereotype *VariableElement* to the item. The stereotype *ElementGroup* introduces additional information through its properties, such as constraints between variable elements. In Sequoia, the decision model is used as a guide enabling to analyze all available variants and paths leading to a completely defined product. Once the derivation activity is launched, the choices described by the decision model are proposed to the user as a series of questions. The output of this process is a completely defined product and the user is not able to make any kind of modification to the initial model until the derivation step is over.

3.10 Case Study and Evaluation Settings

The methodology described above, including manually modeling variability in safety requirements, the automatic retrieval of feature model and the variability-aware bridging of requirements and architecture, has been tested in a case study carried out in nuclear power plants. In this section, we describe the dataset considered in the evaluation.

Table 3.4: Case Study Data Description

Data	Common Elements	Variable Elements		
		France	UK	US
# Requirements	21	27	34	18
# Design rules	3	4	4	4
# Functions	11	2	2	5
# Functions Communications	9	2	2	4
# Systems	19	0	0	1
# Systems Communications	33	15	15	27

Table 3.4 summarizes the input data in the illustrative example provided by the industrial partners to implement our proposed methods. The case study contains one generic project which includes all common elements in the product line and three other projects containing the variable elements, specific for these three countries (products): France, UK and USA. In particular, it describes the following information for each country:

- Safety Requirements: excerpts from national regulations and international standards.
- Design rules with their corresponding OCL constraints implemented in the architecture
- I&C Functions with their various properties (safety category, line of defense, number of redundancies, etc.).
- Functions communications: describes the different communications between functions (unidirectional and bidirectional)
- I&C Systems and their different characteristics (safety class, I&C level, line of defense, number of redundancies)
- Systems communications: a derivation of functional communications on the system architecture.

To complete the case study, our industry partners provided us an association matrix that maps requirements and design rules.

Threshold Settings. To compute requirements clusters, we choose empirically two sorted and distinct edge weights as clustering thresholds t_1 and t_2 between 0 and 1 since we compute cosine similarity. The thresholds values t_1 and t_2 have been set

empirically after several experiments at 0.45 and 0.7. We have found that the majority of well-formed clusters actually occur when the clustering thresholds are set at these two values: most of lower-level features are found when $t_2 = 0.7$ and most of higher-level features are extracted when $t_1 = 0.45$. To select interesting association rules when mining structural and transversal dependencies, we set empirically the minimum thresholds on support and confidence respectively at 0.7 and 0.8.

3.11 Evaluation

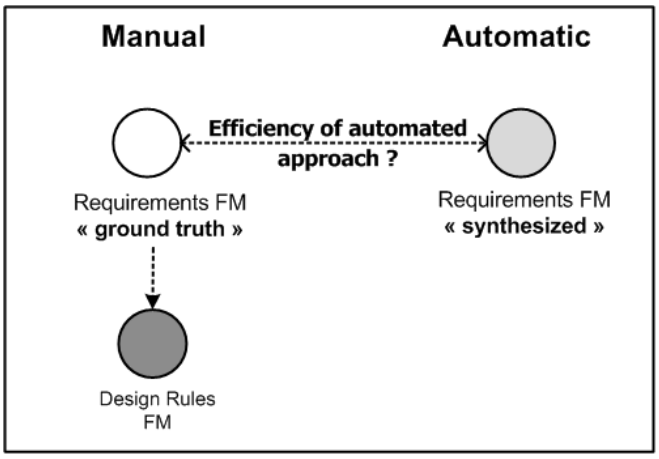


Figure 3.9: Evaluation Overview

So far, we have presented a sound procedure and automated techniques for mining and modeling variability in regulatory requirements. We also provided a methodology to investigate the compliance of the architecture with these regulations. Several more specific research questions were raised under this heading, but one in particular is worth noting: **"How effective are our techniques to automatically reverse engineering a feature model from regulations?"**. Before we address this research question, we will first describe the properties of the ground truth feature models. These latters correspond to the requirements and design rules feature models which are constructed manually based respectively on requirements clustering and design rules clustering (see sections 3.6 and 3.8.1). We computed the following metrics over our dataset:

- **Number of features:** the fewer features are, the more readable is the feature model.
- **% Mandatory features:** the more there are mandatory features, the more reusable the feature model is.

- **% Optional features:** the more exist optional features, the more customized products are.
- **% Requires and exclude constraints:** the higher is the number of requires and exclude constraints, the lower is the number of possible configurations.
- **% OCL constraints:** the greater is the number of OCL constraints, the lower is the number of possible configurations.

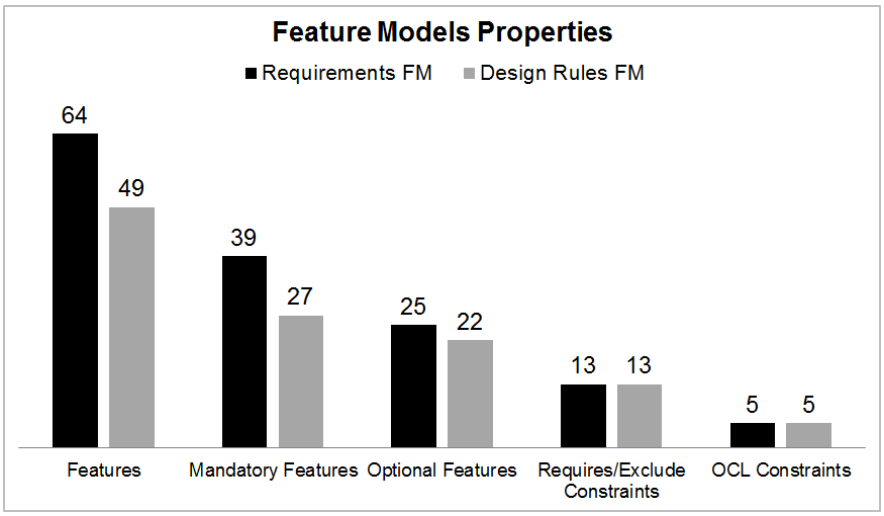


Figure 3.10: Properties of Requirements FM and Design rules FM

The results show that most of the features in the resulting requirements FM are mandatory. Indeed, from a set of 142 individual requirements, we report 64 features out of which 39 features (60.93%) are mandatory and 25 features (39.06%) are optional (see Figure 3.10). This reflects that regulations in the three countries (France, UK and USA) are sharing a significant portion of common concepts related to safety despite the big number of variable requirements (55.6%). Indeed, our goal is not a customization of products, but, to maximize the common core to gain in terms of money and time during I&C systems certification. This requirements feature model which was built manually represents the ground truth for our research question.

Regarding design rules, we obtain a total of 49 features: 27 features (55.1%) are mandatory and 22 features (44.9%) are optional. Despite the limited amount of design rules (21 elements) we get a good number of features. This is explained by the fact that a design rule can deal with one or more preoccupation. Thus, a same design rule can belong to different clusters. Both requirements and design rules FMs contain 13 requires and exclude constraints and only five OCL constraints. Having such number of constraints restricts the number of possible configurations which is reasonable since

we have only three configurations (the three countries). Both feature models have a good level of granularity which is beneficial to nuclear expert to understand and maintain. Furthermore, having non complex feature models facilitates their traceability with development artifacts as well as tracing variability across the different levels of abstraction.

How effective are our techniques to automatically reverse engineering a feature model from regulations?

Objects of Study. Our goal is to evaluate the effectiveness of an automated synthesis technique to produce meaningful clusters and identify correct dependencies between features (structural and transversal).

Experimental Setup. The first issue that we want to verify is whether our techniques are in fact able to generate meaningful clusters of similar requirements. For this, we compute the number of correct clusters with respect to the ground truth feature model. Indeed, we evaluate the results of the method in terms of precision. The ground truth FM corresponds to the requirements feature model which was constructed manually where each feature forms a cluster of similar requirements.

Experimental Results. The results show that 69% of retrieved clusters are correct in one step and without any user intervention. As a result, these techniques are in some way able to identify relevant clusters in the data. On the other hand, 31% of faulty requirements clusters need to be reviewed and corrected. Therefore the role of the domain analyst remains crucial.

A possible explanation for the proportion of the faulty clusters is that the dimensionality reduction can result in a loss of significant information for the similarity measure. As a reminder, LSA uses the mathematical technique SVD to reduce dimensional representation of the matrix that emphasizes the strongest relationships and throws away the noise. In our case study, for a regulatory document of 47 requirements, we obtain a term-requirement matrix of 420 terms in average. However, in the reality, a regulatory document contains a huge number of requirements. Yet, the term-requirement matrix is likely to have several hundreds of thousands of terms and requirements, where a reduced dimensional representation is crucial. A possible solution to improve the quality of clusters is taking into account the proximity of requirements in the document. More specifically, similar requirements tend to be physically close in the requirement document. The intuition behind this is that, as requirement documents are written by humans in natural language, these latters tend to put together related requirements in

order to improve the document flow.

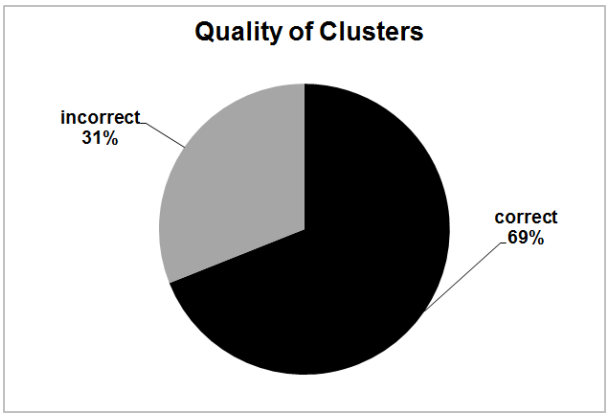


Figure 3.11: Quality of Clusters

In the second step, the resultant hierarchy structures (product models) are adjusted according to domain knowledge by removing incorrect clusters, adding missing features and then renaming the final features. Now that we obtain complete product models, we evaluate the effectiveness of our approach to automatically identify structural dependencies (mandatory and optional relationships) and transversal dependencies (requires and excludes relationships). We notice that structural dependencies show a high predictive capacity: 95% of the mandatory relationships and 60% of optional relationships are found. We also observe that the totality of requires and exclude relationships are extracted. Structural relationships mainly depend on the composition of the products. Thus, they depend on the parent-child relationships or the product models obtained in the first stage of the approach. On the other hand, transversal dependencies are related to relationships attributes.
We observe that our approach is really efficient but it presents some limitations. Our experience shows that there is a need for techniques capable of identifying group cardinalities and also a method that is able to deal with more complex than Boolean-type features, as for instance, features with multiple instantiations.

3.12 Lessons Learned and Discussion

What is the best variability modeling paradigm accordingly to the domain practices?

Though the concepts of similarities and variabilities were roughly understood by our partners, we first had to introduce them to variability and, more specifically variability

modeling. The aim of this introduction is to lay the foundation of the concepts but also identify the boundaries of the discipline. We set up several half to full-day workshops to allow our industrial partners to localize where there will be valuable variability to model, and elicit what they expect from the variability modeling.

We presented several different modeling paradigms, analysis capabilities, current limitations, etc. We discussed the choice of the modeling paradigm and the format of data that we will have to handle. From the numerous approach, we chose to present CVL as our working variability language. CVL attracts a lot of attention due to the fact that it does not require changing the complexity of the development artifacts and can be used in conjunction with different development artifacts.

Interestingly, CVL is domain independent thus it supports a broad range of types including multiple instances and a constraint language for expressing dependencies over these types. This terminology avoids the confusion caused by the ambiguous meaning of the term feature. Having separate models for each concern favors modularization and reusability; this is a step towards externalizing variability from the domain language and standardizing it for any DSL. Moreover, when specifying variability in separate models, it is easier to address variability-aware bridging of different levels of abstraction.

How to deal with the search space in regulatory corpora and what is the right granularity level for modeling requirements variability?

As mentioned earlier, nuclear domain is complex because of the variety of documents one has to handle, the number of requirements they contain, their high level of abstraction and ambiguity, etc. Following a naive method leads to feature models with fine granularity, verbose, and as a result, hard to understand and maintain. To narrow this problem space, the key idea was to analyze variability in regulatory documents by topic on different corpora and on the same abstraction level. Thus, our industrial partners proposed to apply this approach on two standards and for different topics. Furthermore, using traceability matrix improves the understandability and the maintainability of feature models.

How to handle the traceability of the original requirements and the requirements feature model?

When building requirements feature model, we keep the traceability between the identified features and the original regulatory requirements. Indeed, features provide an abstraction of requirements. Every feature covers a particular set of requirements which refine that feature. Feature models are domain models which structure requirements by

mapping them to a feature and by forming relations between them. In this way, domain experts can validate the final feature model against regulations.

How to address the requirements variability on the architecture?

Modeling requirements variability is useful, however there is no direct mapping from requirements to the architecture. Therefore, we proposed heading toward modeling variability in design rules since they act as a pivot between requirements and the architecture. The industrial partners rely on these rules to validate the architecture against safety requirements because they claim to satisfy one or several requirements. This idea provides a mapping between variable requirements and variable architecture elements to investigate the robustness of the derived architecture against regulations variability. Interestingly, we have a limited number of design rules, contrary to requirements, which leads to fewer features to map with the architecture. This is very beneficial, since it facilitates the traceability between variable requirements and variable architecture elements. As a result, we could reduce the complexity of tracing variability from the problem space to the solution space by separating the concepts. This is the reason behind modeling variability in design rules.

3.13 Threats to Validity

As a primary external threat, we have not evaluated the consistency and the adoption of the tool at a larger scale. The size of individual requirements is not big enough (142 requirements for three countries) although the latter is already significant for a manual analysis when mining variability knowledge from text in order to construct the ground truth FM. We plan to extend this evaluation with more formal and quantitative measures in a more advanced dissemination phase.

A first internal threat is that the extraction of a feature model from requirement documents is a mix between automated techniques and manual directives. Indeed, the resulting product models are adjusted by a domain expert according to domain knowledge. The manual intervention essentially consists in removing incorrect clusters, adding missing features and renaming the final features. Domain experts can have different interpretation within a same product model which can lead to a different hierarchy and thus a different set of dependencies between features in the resulting FM.

Another internal threat comes from the manual optimization of the clustering thresholds for the evaluation of the heuristic. Another set of thresholds could generate less favorable results. It is unclear whether this difference would be significant. Similarly,

a manual optimization of minimum thresholds of support and confidence could impact the quality of the obtained rules and in consequence the quality of dependencies among features.

3.14 Conclusion

With the renewal of the nuclear industry, the French nuclear energy industrials are aimed to sell and develop products outside France. A major challenge is the conformance of products to multiple different and heterogeneous regulations, which introduce a necessary product line perspective.

In this chapter, we proposed a formalization of the variability in safety requirements using CVL and provided a variability-aware bridging of different levels of abstraction. We proposed an automated approach to synthesize feature models from these regulations. The approach relied on information retrieval and data mining techniques capable of extracting features and their dependencies: structural dependencies to build the hierarchy and transversal dependencies. The evaluation shows that our approach is able to retrieve automatically 69% of correct clusters. We noticed that structural dependencies show a high predictive capacity: 95% of the mandatory relationships and 60% of optional relationships are found. We also observed that the totality of requires and exclude relationships are extracted.

Chapter 4

Automated Extraction of Product Comparison Matrices From Informal Product Descriptions

This chapter instantiates our general contribution in the second case study to synthesize product comparison matrices from informal product descriptions. In this chapter we propose an approach to automate the extraction of PCMs from unstructured descriptions written in natural language, investigate the complementarity aspect between products descriptions and technical specifications and implement our approach in a tool, *MatrixMiner*. The chapter is structured as follows. Section 4.1 provides additional background on PCMs and elaborates on the PCM synthesis challenge. Section 4.2 gives a general overview of our approach. Sections 4.3 and 4.4 describe the main steps of our approach, namely terms and information extraction, and subsequent construction of the PCM. In Section 4.5, we describe and illustrate the integration of the synthesis techniques into the *MatrixMiner* environment which is published in [BNBA+15]. Section 4.6 presents our case study. Sections 4.7 and 4.8 analyse successively the results of an empirical evaluation and a user study. In Section 4.9, we discuss threats to validity.

4.1 Context

Sellers describe the products they sell on their website using different categories of text forms. It goes from plain text in a single paragraph, formatted text with bullets, to matrices with product specifications. There is a spectrum of product descriptions ranging from structured data (matrices) to informal descriptions written in natural



Synthesis

Features	HP - ENVY	Lenovo - Yoga	Apple® - MacBook	Dell - Inspiron
High-quality images	YES	YES	YES	YES
Voice recognition software	NO	YES	NO	YES
Touch Screen	YES	YES	NO	YES
Media Reader	Digital	Multiformat	Digital	5-in-1
Webcam	Front-facing TrueVision high-definition	N/A	Built-in high-definition	Built-in high-definition
Hard Drive	1TB	500 GB	750 GB	1TB
Resolution	1920 x 1080	1366 x 768	2560 x 1600	1920 x 1080
Processor Speed	2.4 Ghz	1.5 Ghz	2.8 Ghz	3.0 Ghz
Memory	12GB DDR3L SDRAM	4GB DDR3L	16 GB	16GB DDR3L
Processor	5th Gen Intel Core i7-5500U	4th Gen Intel Core i3-4012Y	Fourth-generation quad-core Intel Core i7	5th Gen Intel Core i7-5500U
Operating System	Microsoft Windows 8.1 64-bit	Microsoft Windows 8.1 64-bit	Mac OS X 10.9 Mavericks	Microsoft Windows 8.1 64-bit

Figure 4.1: Automatic Extraction of PCM from Textual Product Descriptions

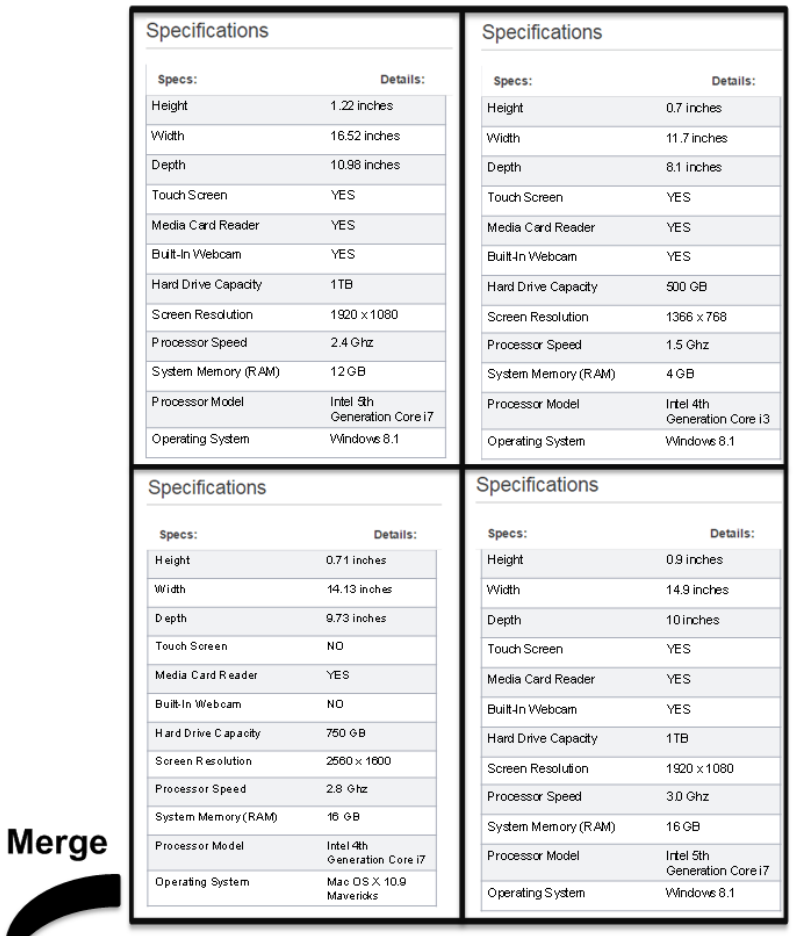


Figure 4.2: Computing Technical Specifications PCM

languages. Both have strengths, weaknesses, and have the potential to comprehensively describe a set of products. BestBuy provides descriptions for hundreds of thousands of products, including: (1) *products overviews*, texts describing features of products using natural language (see Figure 4.1); (2) *technical specifications*, which describe the technical characteristics of products through feature lists (see Figure 4.2).

Figure 4.1 illustrates the common scenario in which a customer needs to buy a laptop on BestBuy website and has to decide among a diversity of products. He/she has to go through many textual descriptions (product overviews) and reasons over the different features of the product. A typical question is to figure out if a particular feature is supported by existing products (if any) and what are the alternatives. In domain analysis, the biggest challenge is related to the number of products and the number of features an analyst has to gather and organize. The more assets and products, the harder the analysis. Our goal is to automate the manual task of analyzing each product with respect to its textual description and clustering information over several products, and provide a reader with an accurate and synthetic *product comparison matrix (PCM)*, as shown in Figure 4.1.

The manual elaboration of a PCM from textual overviews can be done as follows. First, it requires the ability to detect from the text the potentially relevant domain concepts expressed as single or multi words including domain specific terms and numerical information, such as those that are highlighted in the text of Figure 4.1. Once detected, multiwords have to be split between the feature name and its value. We observed different value types for features in a previous work [SAB13]. Each of these value types imply a different interpretation for the feature. For instance, the feature "Touch Screen" means the availability of the feature, which has to be interpreted as a YES/NO value (see the PCM of Figure 4.1). Feature values can also mix letters and numbers, for instance the following snippet: "5th Gen Intel Core i7-5500U". Consequently, determining features and their related values is not a trivial problem.

4.1.1 Toward Automatic Extraction of PCMs

Our objective is to automate the identification of features, their values, and collect information from each product to create a complete PCM. This comes with a set of challenges, mostly due to the informal and unstructured nature of textual overviews.

First, the representation aims to provide a *structured* view of all available products and all available features. From a parsing and natural language processing perspective, plain text and PCMs have different organizations schemes. On the one hand, text is grammatically organized but may not been organized in terms of feature definitions nor

description. As being part of open initiatives such as consumer associations, mainstream initiatives like Wikipedia, or e-commerce websites, one cannot rely on the quality of the textual descriptions, in terms of both wording and organization. For instance, textual descriptions may present features in different orders as to put emphasis on a particular one, or may have different authors that do not share the same writing patterns. On the other hand, a PCM is clearly organized as a set of products, features, and associated values. If a product description provides for free the product's name, it is not trivial to determine its features and their values, which have to be mined from the description, as stated previously.

Second, it is not only a matter of parsing products features and their respective values. It is also a matter of making the most *synthetic* and relevant PCM to enable *comparison*. The number of features depends on both (1) the textual description length, precision, and quality, and (2) the capability to cluster features as they share the same meaning but different names. Finding the right name for a feature can have an impact on the number of features. Being generic (for instance, "processor") increases the possibility to have different values for this feature whereas a series of too specific features ("5th Gen Intel... processor") will only have a YES/NO value with a high risk of features explosion. Ideally we would rather like to extract a feature (e.g. processor) together with a value (e.g. 5th Gen Intel...) out of an informal text.

4.1.2 The Complementarity Aspect of Product Overviews and Technical Specifications

Another interesting observation is the nature of relationship that can exist between product overviews and product specifications. Again, with the same example, but now considering technical specifications, we automatically compute the output PCM (see Figure 4.2).

With our automated extraction from overviews, there is also a potential to complement or even refine technical specifications of products (see the two PCMs in Figure 4.1 and Figure 4.2). Considering the verbosity aspect of natural language, the overview can contain information that refine the information of the specification. If we compare the cell values of the same feature or two equivalent features in the overview and the specification, we observed that the cell value in the overview PCM can refine the cell value in the specification PCM.

For example, "Media Reader" exists in both overview PCM and specification PCM of laptops. In the first case, it has "Digital", "Multiformat", "5–in–1" as possible values, while in the second case, it is simply a boolean feature. "Webcam" is also boolean



Figure 4.3: Complementarity Aspect of Product Overviews and Technical Specifications

in specification PCM and non boolean in overview PCM ("Front–facing TrueVision..." and "Built–in high–definition"). In the specification PCM, "Memory" has "12 GB" as a possible value, while in the overview PCM, the value contains also the type of memory: "12GB DDR3L SDRAM". At the same time, "Operating System" has "Windows 8.1" as a possible value in the specification PCM, however it includes also the architecture in the overview PCM ("Microsoft Windows 8.1 64-bit").

Furthermore, in an overview PCM, we can obtain additional features that could refine features existing in specification PCM. For instance, "High–quality images" and "Voice Recognition Software" are two features in the overview PCM. However, they do not exist in the specification PCM. Hence, overviews can also complement the information of technical specifications.

4.2 Overview of the Automatic Extraction

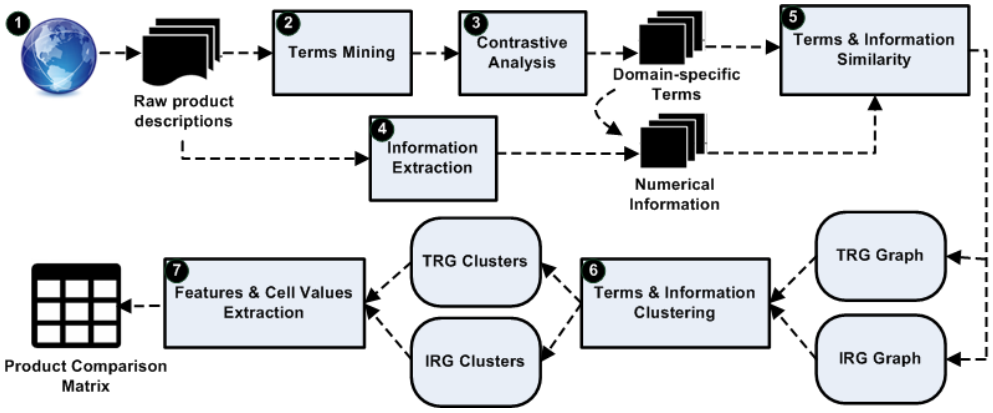


Figure 4.4: Approach Overview

Our approach is summarized in Figure 4.4 and consists of two primary phases. In the first phase, *domain specific terms* and *numerical information* are extracted from a set of informal product descriptions (steps ❶ to ❹), while in the second phase the PCM is constructed (steps ❺ to ❼). For step ❶, the raw product descriptions are extracted along different categories of products. We provide means to either (1) manually select the products to be included in the comparison; or (2) group together closest products within a category. We outline in the following the rest of the procedure.

Mining Domain Specific Terms. Steps ❷ and ❸ are based on a novel natural language processing approach, named *contrastive analysis* [BDVM10], for the extraction of *domain specific terms* from natural language documents. In this context, a *term* is a conceptually independent linguistic unit, which can be composed by a single word or by multiple words. A multi-word is conceptually independent if it occurs in different context (*i.e.* it is normally accompanied with different words). For instance, "Multiformat Media Reader" is a term, while "Reader" is not a term, since in the textual product descriptions considered in our study it often appears coupled with the same word (*i.e.* "Media"). Combining single and compound words is essential to detect features and their values.

The *contrastive analysis* technology aims at detecting those terms in a document that are *specific* for the domain of the document under consideration[BDVM10, Del09]. Roughly, contrastive analysis considers the terms extracted from domain–generic documents (e.g., newspapers), and the terms extracted from the domain-specific document to be analyzed. If a term in the domain–specific document highly occurs also in the domain–generic documents, such a term is considered as domain-generic. On the other hand, if the term is not frequent in the domain–generic documents, the term is considered as domain-specific.

Information Extraction. Step ❹ aims at mining *numerical information* since they are capable to describe precisely the technical characteristics of a product. These information are domain relevant multi-word phrases which contain measures (*e.g.*"1920 x 1080 Resolution") including intervals (*e.g.*"Turbo Boost up to 3.1 GHz").

Inspired by the "termhood" concept used earlier, the extracted multi-words should be conceptually independent from the context in which they appear. For instance, suppose we have in the text this phrase "the processor has 3 MB cache and 2.0 GHz processor speed". Here, "2.0 GHz Processor Speed" is conceptually independent whereas "2.0 GHz Processor" is not. We use statistical filters inspired by the "termhood" metric applied in step ❷, to extract these numerical domain relevant multi-words from text.

Building the PCM. Once the top list for the terms[1] and respectively for numerical information are identified for each product, we start the construction of the PCM. This process requires creating some intermediate structures. The key idea is to perform separately terms clustering from information clustering. A terms cluster gives the possible descriptor values (*e.g.*"Multiformat") while an information cluster provides the potential quantifier values (*e.g.*"1920 x 1080") for the retrieved feature. In step ❺ we compute similarity between terms and correspondingly between information to generate two weighted similarity relationship graphs: a Terms Relationship Graph (TRG) and an Information Relationship Graph (IRG). To identify coherent clusters, we first determine the similarity of each pair of elements by using syntactical heuristics. In step ❻ we apply clustering in each graph to identify terms clusters and information clusters. Finally, step ❼ extracts features and cell values to build the PCM. Elements which are not clustered will be considered as boolean features. We distinguish different types of features (see Figure 4.1): *boolean* which have Yes/No values, *quantified* when their values contain measures (*e.g.*"Resolution", "Hard Drive", etc.), *descriptive* if their values contain only noun and adjectival phrases (*e.g.*"Media Reader"). The resulting PCM can be visualized and refined afterwards.

In the following sections, we elaborate these three main tasks. We address mining terms and information in Section 4.3 and the construction of the PCM in Section 4.4.

4.3 Terms & Information Extraction

In this section, we describe the first half of our approach which handle the terms and information extraction from textual descriptions. Several successful tools have been developed to automatically extract (simple or complex) terms [Dro03, BDVM10]. The reason why we develop our own terms extractor is that we propose later an extraction of numerical information inspired by the termhood concept. This section includes mining domain specific terms (steps ❶ to ❸) in Sections 4.3.1 and 4.3.2, and information extraction (step ❹) in Section 4.3.3.

4.3.1 Terms Mining

Terms mining consists in the first two steps of Figure 4.4. Firstly, raw feature descriptors are mined from each product overview via the BestBuy API. The product overview tend to include a general product description followed by a list of feature descriptors. Therefore, given n products of the same category, we have $D_1 ... D_n$ documents (products

[1] Domain specific terms are generally known as terms.

overviews). From each one of these documents we identify a ranked list of terms. In this section, we discuss the candidate extraction process, that makes use of: i) linguistic filters; ii) stoplist; iii) statistical filters (C-NC Value).

4.3.1.1 Linguistic filters.

The linguistic filters operate on the automatic Part–of–speech (POS) tagged and lemmatized text, making use of different kinds of linguistic feature. POS tagging is the assignment of a grammatical tag (e.g. noun, adjective, verb, preposition, determiner, etc.) to each word in the corpus. It is needed by the linguistic filter which will only permit specific strings for extraction. After POS tagging, we select all those words or groups of words (referred in the following as multi–words) that follow a set of specific POS patterns (i.e., sequences of POS), that we consider relevant in our context. Without any linguistic information, undesirable strings such as of the, is a, etc., would also be extracted.

Since most terms consist of nouns and adjectives, [Sag90], and sometimes prepositions, [JK95], we use linguistic filters that accepts these types of terms (see F1, F2 and F3). It extracts terms like *operating system*, *digital media reader*, *wide array of streaming media*, etc.

The choice of linguistic filters affects the precision and the recall of the output list, e.g. a restrictive filter will have a positive effect on precision and a negative effect on recall [BMPZ01]. We are not strict about the choice of a specific linguistic filter, since different applications require different filters. We will present our method combined with each of these three filters:

F1: Noun^{+} Noun

F2: $(\text{Adj}|\text{Noun})^{+}$ Noun

F3: $(\text{Noun Prep} \mid \text{Adj})^{*}\ \text{Noun}^{+}$

In our method, we use a filter which also constrains the maximum number of words of which a complex term can be made. In fact, we operate on the candidate terms length (l) as one of the main linguistic constraints to be ruled. We believe that such a measure is to be considered as domain–dependent, being related to the linguistic peculiarities of the specialized language we are dealing with. In arts for example, terms tend to be shorter than in science and technology. The length also depends on the type of terms we accept. Terms that only consist of nouns for example, very rarely contain more than 5 or 6 words.

The process of finding this maximum length is as follows: we attempt to extract strings of a specific length. If we do not find any strings of this length, we decrease the number

by 1 and make a new attempt. We continue in this way until we find a length for which strings exist. At this point, extraction of the candidate strings can take place.

4.3.1.2 Stoplist.

A stop-list is a list of words which are very common words. These words are not included in standard representations of documents because they are common to all the documents and cannot be good discriminators. Removing the stop words allows us to focus on the sole important words in the representations.

4.3.1.3 Statistical filters based on C-NC Value.

Terms are finally identified and ranked by computing a "termhood" metric, called C-NC value [BDVM10]. This metric establishes how much a word or a multi-word is likely to be conceptually independent from the context in which it appears. Here we give an idea of the spirit of the metric. Roughly, a word/multi-word is conceptually dependent if it often occurs with the same words (i.e., it is nested). Instead a word/multi-word is conceptually independent if it occurs in different context (i.e., it is normally accompanied with different words). Hence, a higher C-NC rank is assigned to those words/multi-word that are conceptually independent, while lower values are assigned to words/multi-words that require additional words to be meaningful in the context in which they are uttered.

C Value. The C-Value calculates the frequency of a term and its subterms. If a candidate term is found as nested, the C-Value is calculated from the total frequency of the term itself, its length and its frequency as a nested term; while, if it is not found as nested, the C-Value, is calculated from its length and its total frequency. Given the candidate term t, and being |t| its length, the C-Value of t is given as:

$$C - value(t) = \begin{cases} \log_2 |t| \cdot f(t) & \text{if } t \text{ is not nested,} \\ \log_2 |t| \cdot (f(t) - \frac{1}{P(T_t)} * \sum_{b \in T_t} f(b)) & \text{otherwise.} \end{cases}$$

where $f(t)$ is the frequency of t in the corpus, T_t is the set of terms that contains t, $P(T_t)$ is the number of candidate terms in T_t, and $\sum_{b \in T_t} f(b)$ is the sum of frequencies of all terms in T_t.

NC Value. The NC-Value measure [FA99] aims at combining the C-Value score with the context information. A word is considered a context word if it appears with the extracted candidate terms. The algorithm extracts the context words of the top list

of candidates, and then calculates the N-Value on the entire list of candidate terms. The higher the number of candidate terms with which a word appears, the higher the likelihood that the word is a context word and that it will occur with other candidates. If a context word does not appear in the extracted context list, its weight for such term is zero.

Formally, given w as a context word, its weight will be: $weight(b) = \frac{t(w)}{n}$ where $t(w)$ is the number of candidate terms w appears with, and n is the total number of considered candidate terms; hence, the N-Value of the term t will be $\sum_{w \in C_t} f_t(w) * weight(w)$, where $f_t(w)$ is the frequency of w as a context word of t, and C_t is the set of distinct context words of the term t. Finally, the general score, NC-Value, will be:

$$NCValue = \alpha * CValue(t) + \beta * NValue(t) \tag{4.1}$$

where, in our model, α and β are set empirically ($\alpha = 0.8$ and $\beta = 0.2$).

After this analysis, for each D_i, we have a ranked list of words/multi-words that can be considered *terms*, together with their ranking according to the C-NC metric, and their frequency (i.e., number of occurrences) in D_i. The more a word/multi-word is likely to be a *term*, the higher the ranking. From the list we select the k terms that received the higher ranking. The value of k shall be empirically selected. A higher value guarantees that more domain-specific terms are included in the list. On the other hand, higher values for k might also introduce noisy items, since also words/multi- words with low rank might be included.

4.3.2 Contrastive Analysis

The previous step leads to a ranked list of k terms where all the terms might be domain-generic or domain-specific. With the contrastive analysis step, terms are re-ranked according to their domain-specificity. To this end, the proposed approach takes as input: 1) the ranked list of terms extracted from the document D_i; 2) a second list of terms extracted with the same method described in Section 4.3.1 from a set of documents that we will name the contrastive corpora. The contrastive corpora is a set of documents containing domain-generic terminology. In particular, we have considered the Penn Treebank corpus, which collects articles from the Wall Street Journal. The new rank $R_i(t)$ for a term t extracted from a document D_i is computed according to the function:

$$R_i(t) = \arctan(\log(f_i(t)) \cdot (\frac{f_i(t)}{\frac{F_c(t)}{N_c}}) \tag{4.2}$$

where $f_i(t)$ is the frequency of the term t extracted from D_i, $F_c(t)$ is the sum of the frequencies of t in the contrastive corpora, and N_c is the sum of the frequencies of all the terms extracted from D_i in the contrastive corpora. Roughly, if a term is less frequent in the contrastive corpora, it is considered as a domain-specific term, and it is ranked higher. If two terms are equally frequent in the contrastive corpora, but one of them is more frequent in D_i, it is considered as a term that characterizes the domain more than the other, and, again, it is ranked higher.

After this analysis, for each D_i, we have a list of terms, together with their ranking according the function R, and their frequency in D_i. The more a term is likely to be domain-specific, the higher the ranking. From each list, we select the l terms that received the higher ranking. The choice of l shall be performed empirically: higher values of l tend to include terms that are not domain-specific, while lower values tend to exclude terms that might be relevant in the subsequent phases.

4.3.3 Information Extraction

Besides domain-specific terms, we also consider *numerical information* defined as domain relevant multi-word phrases containing numerical values, since they are capable to describe precisely the technical characteristics of a product.

We use filters that extract multi-words including numbers (Integer, Double, percentage, degree, etc.): *3.0 GHz Processor Speed*, *Microsoft Windows 8.1 64-bit Operating System*; multiplication of numbers: *1920* x *1080 Resolution*; and intervals: *Turbo Boost up to 3.6GHz*, *Memory expandable to 16GB*. Our method is combined with each of these three filters:

F4: *Nb-Exp* (Adj|Noun)* Noun
F5: (Adj|Noun)* Noun *Nb-Exp*
F6: (Adj|Noun)* Noun *Nb-Exp* (Adj|Noun)* Noun

where, *Nb-Exp* is a measure following these patterns:

- Number (Integer, Double, percentage, degree, etc.): Nb, $Nb\%$, $Nb°$.
- Multiplication of numbers: $Nb \times Nb$.
- Interval: $Nb - Nb$, *up to* Nb, *down to* Nb, *expandable to* Nb, $\leq Nb$, $\geq Nb$, etc.

Inspired by the "termhood" concept used earlier, the extracted multi-words should be conceptually independent from the context in which they appears. For instance, "3.0 GHz Processor Speed" is conceptually independent whereas "3.0 GHz Processor" is not. Similarly, we identify a ranked list of domain relevant multi-words from each

document D_i by applying first linguistic filters (F4, F5, F6) using POS tagging and second statistical filters inspired by C-NC Value.

When combining the C-Value score with the context information, the algorithm extracts the context words (obviously not numbers) of the top list of candidates and then calculates the N-Value on the entire list of candidate multi-words. When computing the weight of a context word w, $t(w)$ is not only the number of candidate multi-words w appears with but also the number of domain-specific terms containing w and n is the total number of considered candidate multi-words and domain-specific terms.

Hence, for each D_i, we have a ranked list of multi-words that can be considered domain relevant, together with their ranking according to the C-NC metric. The more a multi-word is likely to be a domain relevant, the higher the ranking. From the list we select the k multi-word that received the higher ranking. The value of k shall be empirically selected.

4.4 Building the PCM

Now that we have for each product a list of domain specific terms ranked according to the C-NC metric and their frequency in the corresponding product descriptions and also a list of numerical information ranked according to the C-NC Value, the whole challenge consists in building a sound and meaningful PCM. This process requires to find out the final features and compute the cell value for each couple product-feature.

To extract features and cell values, a first natural strategy is to perform clustering based on the similarity of the elements (terms or information) to compute groups of elements. The intuitive idea is that clusters of syntactically similar elements can be exploited to identify the common concern, which can be organized as variability concept, and its possible values, since elements in a cluster are likely to share a common feature but with different quantifications (in the case of information clusters) or descriptions (in the case of terms clusters). Cell values can be (see the PCM of Figure 4.1):

- **Boolean:** can take a value of True or False, to represent whether the feature is present or not.
- **Descriptors:** noun phrases and adjectival phrases given according to this pattern: *(Adj | Noun)+*: "Digital" and "Multiformat" are two descriptor values of "Media Reader"; and "Front-facing TrueVision high-definition" and "Built-in high-definition" represent two potential values of "Webcam".
- **Quantifiers:** measures that can be Integer, Double, Partial, etc; in compliance with *Nb-Exp ((Adj | Noun)* Noun)** pattern. For instance, "1366 x 768" as

"Resolution", "12GB DDR3L SDRAM" as "Memory"; "up to 3.1 GHz" as "Turbo Boost"; and "Microsoft Windows 8.1 64-bit" as "Operating System".

4.4.1 Terms and Information Similarity

The goal here (step ❺ in Figure 4.4) is to determine a weighted similarity relationship graph among terms and respectively among numerical information. Two graphs were constructed: *Terms Relationship Graph* (TRG) and *Information Relationship Graph* (IRG) in which nodes represent respectively terms and information. To identify coherent clusters, we first determined the similarity of each pair of elements through computing syntactical heuristics.

Syntactical heuristics use edit distance and other metrics based on words' morphology to determine the similarity of two elements. We used the so-called Levenshtein edit distance [WF74] that computes the minimal edit operations (renaming, deleting or inserting a symbol) required to transform the first string into the second one.

4.4.2 Terms and Information Clustering

After building the two relationship graphs, we apply terms clustering in TRG and information clustering in IRG to identify respectively terms clusters and information clusters (step ❻ in Figure 4.4). The underlying idea [CZZM05] is that a cluster of tight-related elements with different granularities can be generated by changing the clustering threshold value t. If there is an edge between two elements and its weight is greater than or equal to t, they will be put into the same cluster. So the edges whose weights are above or equal to the threshold value are set to be valid; otherwise, the edges are invalid. Then connected components are computed by the valid edges. Each connected component is a cluster of tight-related elements sharing the same concern which represents the feature. As we decrease the threshold value, more edges are set to be valid, and we get clusters with coarser granularity.

4.4.3 Extracting features and Cell Values

Finally to construct the PCM, we need to extract the features and the cell values from terms clusters and information clusters (step ❼ in Figure 4.4). To retrieve the feature name from a cluster, we developed a process that involved selecting the most frequently occurring phrase from among all elements (terms or information) in the cluster. This approach is similar to the method presented in [HL04] for summarizing customer reviews. To identify the most frequently occurring phrase we reuse the POS

tags identified earlier (see Section 4.1). The elements are then pruned to retain only *Noun*$^+$ for terms clusters and *(Adj/Noun)** *Noun* for information clusters, as the other expressions were found not to add useful information for describing a feature.

Frequent itemsets are then discovered for each of the clusters. In this context, frequent itemsets are sets of expressions that frequently co-occur together in the elements assigned to the same cluster. More formally, the support of an itemset I is the number of elements in the cluster that contain all the expressions in I. Given a pre-determined itemset support threshold, s, I is considered frequent if its support is equal or larger than s[2]. Various algorithms exist for mining frequent itemsets including the Apriori [AS+94] and Eclat algorithms. We chose to use Apriori as it is shown to be memory-efficient and hence suitable for the size of our data set. To select a feature name, the frequent itemset of maximum size, FIS_{max} is selected. Finally, to extract cell values, we substitute FIS_{max} from each element within the cluster. For example, "Digital Media Reader" and "Multiformat Media Reader" form a terms cluster. "Media Reader" is the feature name, while "Digital" and "Multiformat" are two possible values. At the same time, "1920 x 1080 Resolution" and "1366 x 768 Resolution" represent information cluster that gives "Resolution" as a feature name and two potential values: "1920 x 1080" and "1366 x 768". Elements which are not clustered will be considered as boolean features. Each cluster adds one column in the PCM containing the feature and the corresponding cell value for each product in the family.

4.5 Tool Support

MatrixMiner offers an interactive mode where the user can import a set of product descriptions, synthesize a complete PCM, and exploit the result. We also have precomputed a series of PCMs coming from different categories of BestBuy (Printers, Cell phones, Digital SLR Cameras, Dishwashers, Laptops, Ranges, Refrigerators, TVs, Washing Machines). Our tool also provides the ability to visualize the resulting PCM *in the context* of the original textual product descriptions and also the technical specification typically to control or refine the synthesized information [BNBA+15].

4.5.1 Implementation and Used Technologies

Stanford CoreNLP[3] provides a set of natural language analysis tools which can take raw text input and give the base forms of words, their parts of speech, etc. Stanford

[2]We set this threshold to 1 since we want to find out itemsets that occur in all elements in the cluster.

[3]`http://nlp.stanford.edu`

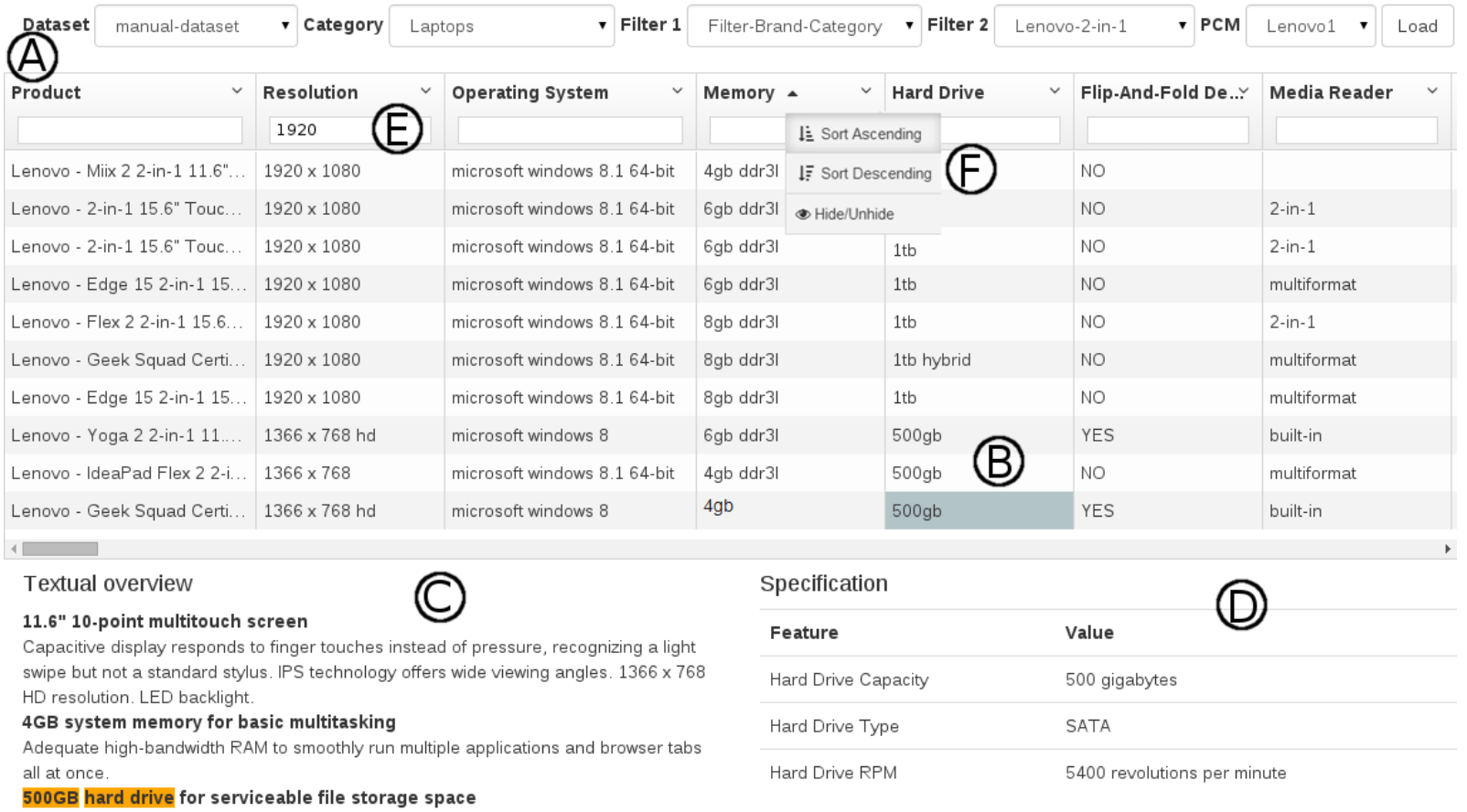


Figure 4.5: The editor of *MatrixMiner* in action

CoreNLP integrates many of NLP tools, including the Part-Of-Speech (POS) tagger that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc. To tokenize and remove stop words from text we use Lucene[4] which is a high-performance, scalable Information Retrieval (IR) library for text indexing and searching. *Levenshtein* computes syntactical similarity based on words' morphology. It comes from the Simmetrics[5] library. Mining frequent itemsets is implemented using R script. The specific source code of the extraction procedure is available online[6]. Our Web environment reuses the editor of OpenCompare[7].

4.5.2 Importing, Visualizing, and Editing

The *MatrixMiner* environment is dedicated to the visualization and edition of PCMs. Human intervention is beneficial to (1) refine/correct some values (2) re-organize the matrix for improving readability of the PCM.

As a result we developed an environment for supporting users in these activities. Our tool provides the capability for *tracing* products and features of the extracted PCM to the original product overviews and the technical specifications. Hence the PCM can be interactively controlled, complemented or refined by a user. Moreover users can restructure the matrix through the grouping or ordering of features. Overall, the features available are the following:

- select a set of comparable products. Users can rely on a number of filters (e.g. category, brand, sub categories, etc. See Figure 4.5, Ⓐ);
- ways to visualize the PCM with a traceability with original product descriptions. For each cell value, the corresponding product description is depicted with the highlight of the feature name and value in the text. For instance, "500GB Hard Drive" is highlighted in the text when a user clicks on "500GB" (see Figure 4.5, Ⓑ and Ⓒ);
- ways to visualize the PCM with a traceability with the technical specification (see Figure 4.5, Ⓓ). For each cell value, the corresponding specification is displayed including the feature name, the feature value and even other related features. Regarding our running example, "Hard Drive Capacity" and two related features ("Hard Drive Type" and "Hard Drive RPM") are depicted together with their corresponding values;

[4] `https://lucene.apache.org`
[5] `http://sourceforge.net/projects/simmetrics`
[6] `https://github.com/sbennasr/matrix-miner-engine`
[7] `https://github.com/gbecan/OpenCompare`

- basic features of a PCM editor. Users can remove the insignificant features, complete missing values, refine incomplete values or revise suspect values if any – typically based on information contained in the textual description and the technical specification;
- advanced features of a PCM editor: means to filter and sort values (see Figure 4.5, Ⓔ and Ⓕ); ways to distinguish Yes, No and empty cells using different colors to improve the readability of the PCM; prioritize features by changing the columns order, etc.

4.6 Case Study and Evaluation Settings

The PCM extraction methodology described above has been tested in a case study carried out in BestBuy website. In this section, we describe the dataset considered in the evaluation of our approach, we present the evaluation settings and introduce our research questions for both empirical and user studies.

4.6.1 Data

We selected 9 products categories that cover a very large spectrum of domains (Printers, Cell phones, Digital SLR Cameras, Dishwashers, Laptops, Ranges, Refrigerators, TVs, Washing Machines) from Bestbuy. Currently, we have implemented a mining procedure on top of BestBuy API[8] for retrieving numerous product pages along different categories. We mined 2692 raw product overviews using Bestbuy API. The characteristics of the dataset are summarized in Table 4.1.

Table 4.1: Overview dataset

Products Category	#Products Overviews	#Words per Overview (Avg)
Laptops	425	350
Cell Phones	99	225
Cameras	141	279
Printers	183	277
TVs	253	283
Refrigerators	708	187
Ranges	538	275
Washing Machines	107	255
Dishwashers	238	263
Total	2692	897,020

4.6.2 Threshold Settings

Among the automatically extracted terms, for each D_i we have selected the k = 30 items that received the higher ranking according to the C-NC Value. The value for

[8] https://developer.bestbuy.com

k has been empirically chosen: we have seen that the majority of the domain-specific terms – to be re-ranked in the contrastive analysis phase – were actually included in the first 30 terms. We have seen that higher values of k were introducing noisy items, while lower values were excluding relevant domain-specific items.

The final term list is represented by the top list of 25 terms ranked according to the contrastive score: such a list includes domain–specific terms only, without noisy common words. It should be noted that the two thresholds for top lists cutting as well as the maximum term length can be customized for domain–specific purposes through the configuration file. As it was discussed in Section 4.3.1.1, the length of multi–word terms is dramatically influenced by the linguistic peculiarities of the domain document collection. We empirically tested that for the electronics domain, multi–word terms longer than 7 tokens introduce noise in the acquired term list.

Now regarding automatically retrieved numerical information, for each D_i we have selected the k = 15 items that received the higher ranking according to the C-NC Value. To calculate clusters of similar terms (resp. information), the threshold of similarity t has been set empirically after several experiments at 0.6 (resp. 0.4): we have seen that the majority of well-formed clusters actually occur when the similarity thresholds are set at these values.

4.6.3 Research Questions

So far, we have presented a sound procedure and automated techniques, integrated into the *MatrixMiner* environment, for synthesizing PCMs. Our evaluation is made of two major studies:

Empirical Study. It aims to evaluate the extraction procedure (when considering product overviews) and also investigate the relationships between overviews and technical specifications. We address three research questions:

- **RQ1.1:** What are the properties of the resulting PCMs extracted from textual overviews?
- **RQ1.2:** What is the impact of selected products on the synthesized PCMs (being from overviews or technical specifications)?
- **RQ1.3:** What complementarity exists between an "overview PCM" and a "technical specification PCM"?

User Study. The purpose here is to evaluate the quality of the generated PCMs and also the overlap between overview PCMs and specification PCMs from a user point of

view. Two main research questions emerge:

- **RQ2.1:** How effective are the techniques to fully extract a PCM ?
- **RQ2.2:** How effective is the overlap between overview PCMs and specification PCMs?

The research questions RQ1.1 to RQ1.3 are addressed in Section 4.7, RQ2.1 and RQ2.2 are addressed in Section 4.8.

4.7 Empirical Study

4.7.1 Dataset

We create two main datasets: overviews dataset and specifications dataset. Each of them comprises two sub-datasets (random and supervised) which contain respectively a random and supervised selection of groups of 10 products belonging to the same category (e.g. laptops).

Overviews Dataset (D1).

SD1.1: Overviews Dataset (random). We randomly select a set of products (also called clusters hereafter) in a given category (e.g. laptops) and we gather the corresponding products overviews. To reduce fluctuations caused by random generation [AB11], we run 40 iterations for each category. Results are reported as the mean value over 40 iterations.
SD1.2: Overviews Dataset (supervised clustering). A domain expert manually selected 169 clusters of comparable products against product overviews. To this end, he relies on a number of filters proposed by Bestbuy (brand, sub categories, etc.). The key idea is to *scope* the set of products so that they become comparable.

Specifications Dataset (D2).

SD2.1: Specifications Dataset (random). We keep the *same* set of products as SD1.1 (that is based on a random strategy). This time we consider technical specifications.
SD2.2: Specifications Dataset (supervised). We keep the *same* set of products as SD1.2. (that is based on a supervised clustering). We consider technical specifications.

4.7.2 RQ1.1. Properties of the resulting PCMs extracted from textual overviews

Objects of Study. When extracting PCMs, an intuitive feeling is to group together comparable products within a given category (SD1.2). Regarding this research question, we aim to describe the properties of the synthesized overview PCMs and comparing them to the specification PCMs adopting a supervised scoping (SD2.2).

Experimental Setup. To answer our research question, we compute the following metrics over these two datasets: SD1.2 and SD2.2.

- **PCM size:** the smaller is the size of the PCM, the more exploitable is the matrix.
- **% Boolean features:** the fewer boolean features there are, the more readable is the PCM.
- **% Descriptive and quantified features**: the more quantified and descriptive features there are, the more usable and exploitable is the PCM.
- **% Empty cells (N/A):** the fewer empty cells there are, the more compact and homogeneous is the PCM.
- **% Empty cells per features category:** in particular, we measured the percentage of boolean empty cells, the percentage of quantified empty cells and the percentage of descriptive empty cells.
- **Number of empty cells per features category (Avg):** specifically, we measured the average of empty cells per boolean feature, the average of empty cells per quantified feature and the average of empty cells per descriptive feature.

Experimental Results. The results show that the synthesized PCMs exhibit numerous quantitative and comparable information (see Table 4.2). Indeed, the resulting overview PCMs contain in average 107.9 of features including 12.5% of quantified features and 15.6% of descriptive features. Only 13% of cell values are empty which demonstrate that our approach is able to generate *compact* PCMs.
When applying a supervised scoping, we notice that specification PCMs have 35.8% less features in average than overview PCMs. The nature of product overviews (and the verbosity of natural languages) partly explains the phenomenon. Interestingly, overview PCMs reduce the percentage of empty cells by 27.8 percentage points.

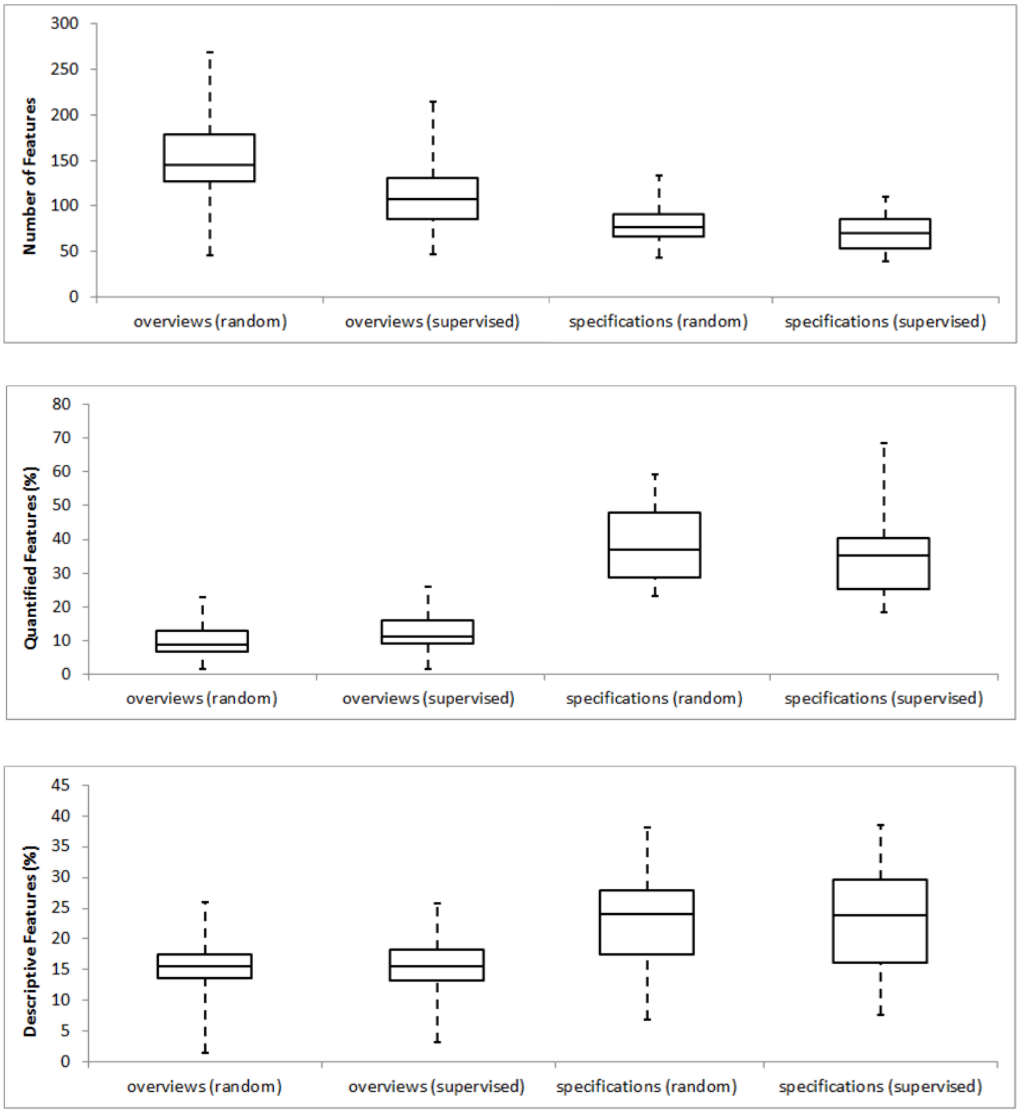


Figure 4.6: Features: Random vs Supervised Scoping

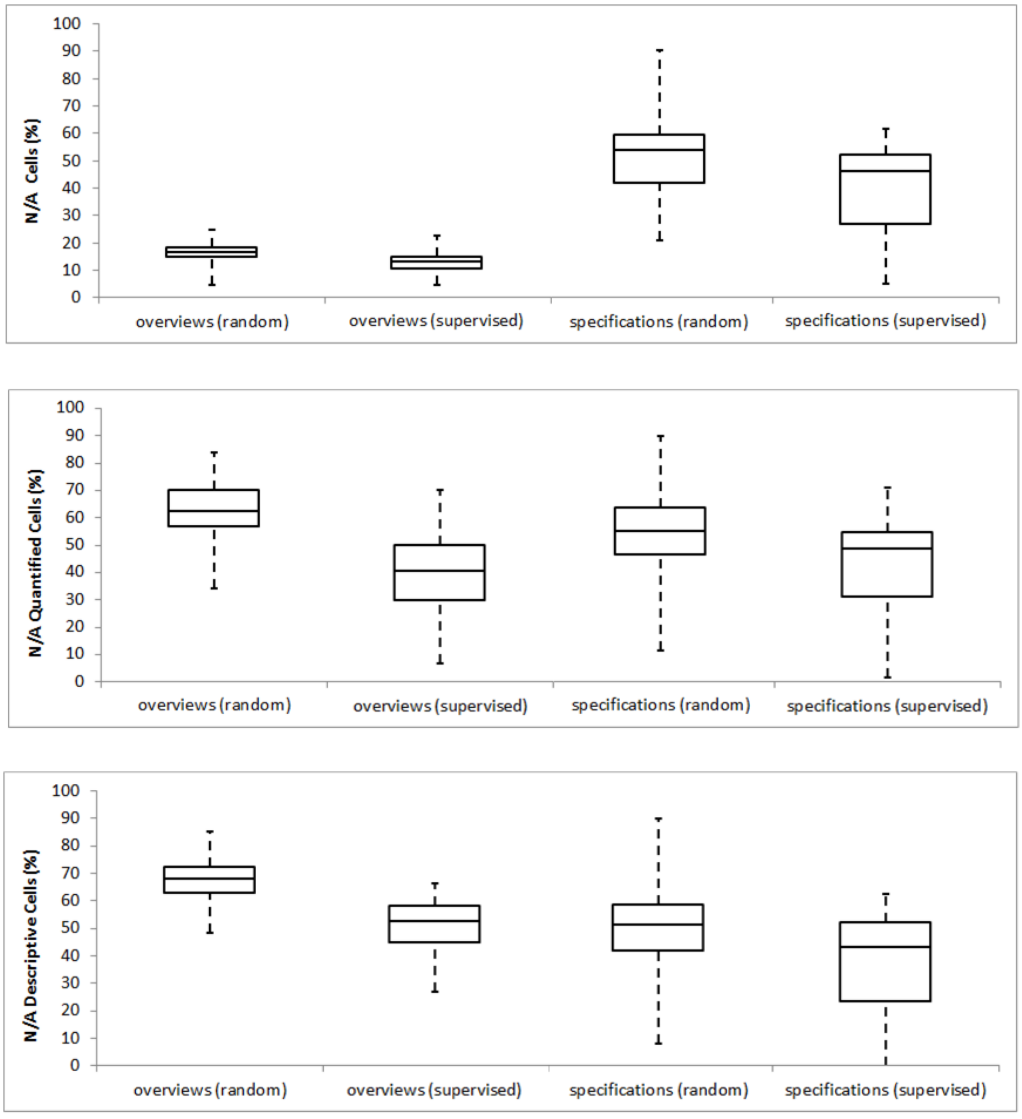


Figure 4.7: Cell Values: Random vs Supervised Scoping

Table 4.2: Properties of Synthesized PCMs: Random vs Supervised Scoping

(a) Features Properties

	Overviews (random)		Overviews (supervised)		Specifications (random)		Specifications (supervised)	
Metrics	Average	Median	Average	Median	Average	Median	Average	Median
Size of PCM	1552.7	1445	1079.7	1070	797.6	765	692.4	700
#Features	155.2	144.5	107.9	107	79.7	76.5	69.2	70
%Boolean Features	74.7	75	71.8	72.2	38.5	37.2	35.9	35.2
%Quantified Features	9.6	8.8	12.5	11.2	38.5	36.8	40.5	40.6
%Descriptive Features	15.5	15.6	15.6	15.5	22.6	24	23.5	23.9

(b) Cell Values Properties

	Overviews (random)		Overviews (supervised)		Specifications (random)		Specifications (supervised)	
Metrics	Average	Median	Average	Median	Average	Median	Average	Median
%N/A	16.4	16.7	13.0	13.1	51.9	54.0	40.8	46.2
%N/A Boolean Cells	0	0	0	0	51.2	52.6	39.7	42.7
Avg #N/A Boolean Cells per Feature	0	0	0	0	5.1	5.2	3.9	4.2
%N/A Quantified Cells	62.8	62.3	40.1	40.7	54.2	55.2	42.8	48.6
Avg #N/A Quantified Cells per Feature	6.2	6.2	4.0	4.0	5.4	5.5	4.2	4.8
%N/A Descriptive Cells	67.4	68.1	51.5	52.5	48.6	51.4	38.5	43
Avg #N/A Descriptive Cells per Feature	6.7	6.8	5.1	5.2	4.8	5.1	3.8	4.3

4.7.3 RQ1.2. Impact of Selected Products on the Synthesized PCMs

Objects of Study. The purpose of this experiment is to investigate the impact of selected products on the quality of the PCMs synthesized from textual overviews (overview PCMs) or obtained from technical specifications (specification PCMs). Does the set of considered products influence the properties of the PCM (e.g., number of empty cell values)? The hypothesis is that a naive selection of input products may lead to a non-compact and non-exploitable PCM. This question aims at quantifying this potential effect.

Experimental Setup. To answer our research question, we compare random and supervised techniques for products selection according to the metrics that we had used previously. Thus, we need to compute as well these metrics over random datasets: SD1.1 and SD2.1. Table 4.2, Figures 4.6 and 4.7 describe the properties of the synthesized PCMs when applying random and supervised scoping.

Experimental Results.

Complexity of PCMs. We compare the properties of overview PCMs generated using a random scoping and those engendered from a supervised scoping. We first notice that

a supervised manner reduces significantly the complexity of the derived PCMs with 30.4% less cells and as much less features than a random selection of products. These results also show that our extraction is capable of exploiting the fact that products are closer and more subject to comparison. Similarly, when we compare specification PCMs obtained respectively from a naive and supervised selection of products, we observe that a supervised scoping gives better results. Indeed, supervised PCMs contain 13.1% less cells and likely less features than random PCMs.

Homogeneity of PCMs. Following a naive scoping, we extracted overview PCMs with 16.4% of empty cells in average, whereas a manual clustering of products leads to a lower percentage of empty cells (13% in average). In particular, we observe that supervised matrices decrease by 22.7 (resp. 15.9) percentage points the percentage of quantified (resp. descriptive) empty cells. For both naive and manual selection, we obtained no boolean empty cells. Considering a supervised manner, our approach increases by around 3 percentage points the percentage of quantified features, with 2.2 less empty cells per feature in average. Supervised matrices have almost the same percentage of descriptive features as random matrices (15.6% in average) but with 1.6 less empty cells per feature in average.

Similarly, supervised scoping enhances the homogeneity of the specification PCMs. The percentage of empty cells declines by 11.1 percentage points. Specifically, supervised PCMs reduce the percentage of quantified (resp. descriptive) empty cells by 11.4 (resp. 10.1) percentage points. In the same time, the supervised selection increases by 2 percentage points the percentage of quantified features and around one percentage point the percentage of descriptive features.

Key findings for RQ1.1 and R1.2.

- Our approach is capable of extracting numerous quantitative and comparable information (12.5% of quantified features and 15.6% of descriptive features).
- A supervised scoping of the input products reduces the complexity (in average 107.9 of features and 1079.7 of cells) and increases the homogeneity and the compactness of the synthesized PCMs (only 13% of empty cells).
- An open problem, due to the nature of product overviews and specifications, is that the size of PCMs can be important. It motivates the next research question: we can perhaps rely on "overlapping" features between specifications and overviews in order to reduce the size.

4.7.4 RQ1.3. Complementarity of Overview PCM and Specification PCM

Objects of Study. The purpose of RQ1.3 is to analyze the relationship between overview PCMs and specification PCMs. Is there any overlap? Which proportions? Can the overview complement or refine the specification? Is the extraction reliable on the overlapping part? Our goal is to calculate the overlap between overviews matrices and specifications matrices extracted using a supervised clustering.

Experimental Setup. To address RQ1.3, we compared the features and the cell values for the same set of products in both overview and specification PCMs using the following metrics:

- % Correct features in the overview matrices comparing to the specification matrices (`Features Over in Spec`): we consider that a feature in an overview PCM is correct, if it is similar to another feature in the specification PCM.
- % Correct features in the specification matrices comparing to the overview matrices (`Features Spec in Over`): we follow the same principle described above.
- % Correct cell values in the overview matrices comparing to the specification matrices (`Cells Over in Spec`): for a given product and two similar features in the overview PCM and the specification PCM, we consider that the cell value in the overview PCM is correct if it is similar to the cell value in the the specification PCM.
- % Correct cell values in the specification matrices comparing to the overview matrices (`Cells Spec in Over`): we apply the same principle as `Cells Over in Spec`.

Two features are similar if at least one of them occurs in the other. Now, for two similar features and a given product, two cell values are similar if at least one of them contains the other. Figure 4.8 illustrates the overlap between overview PCMs and specification PCMs.

Experimental Results.

Features Overlap. Overview matrices cover approximately half of the features in the specification matrices (49.7% in average, 51.0% for median). However, these latters cover only 20.4% of features in the overview matrices (20.6% for median).

Cells Overlap. Overview matrices cover 26.2% of cell values in the specification PCMs (in average, 26.3% for median), while these latters cover only 8.6% of cell values in the overview PCMs (8.5% for median).

The results provide evidence that, with our automated extraction from overviews, there is also a potential to complement technical specifications of products. Another interesting point is that the user can rely on the overlapping features between specifications and overviews to prioritize features and then keep the most relevant ones, in order to reduce the complexity of the overview PCM.

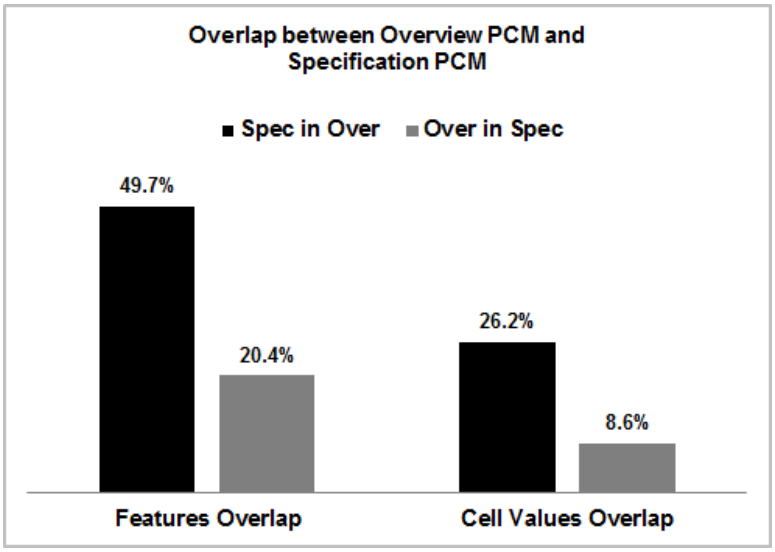


Figure 4.8: Complementarity of Overview PCMs and Specification PCMs (RQ1.3)

Key findings for RQ1.3.

- A significant portion of features (49.7%) and cell values (26.2%) is recovered in the technical specifications.
- The proportion of overlap of overview PCMs regarding specification PCMs is significantly greater than the overlap of the latter regarding overview matrices. This is explained by the fact that the natural language is richer, more refined and more descriptive compared to a list of technical specifications.
- In this context, the user can benefit from this overlap to prioritize features in the overview matrices to reduce the number of features and, consequently reduce the complexity of the matrix (see RQ1.2).

4.8 User Study

4.8.1 Experiment Settings

Dataset: We consider the same set of supervised overview PCMs used earlier in the empirical study: the dataset SD1.2 (167 PCMs in the total). These PCMs cover a very large spectrum of domains (Printers, Cell phones, Digital SLR Cameras, Dishwashers, Laptops, Ranges, Refrigerators, TVs, Washing Machines, etc.). These PCMs are made from various sizes, going from 47 to 214 columns (features), and 10 rows (products).

Participants: The PCMs are evaluated separately by 20 persons (mainly researchers and engineers) that were not aware of our work.

Evaluation Sessions: We organize one evaluation session in which we explain the goal of the experiment to the evaluators. We provide a tutorial describing the tool they would have to use, as well as the concepts they were about to evaluate and some illustrative examples. We display randomly one column at a time (from any PCM) and the evaluator has to attribute scores for the feature and cell values. The evaluation session took one hour.

The evaluators have to validate or not features and cell values in the PCM against the information contained in the original text. To this end, the tool provides ways to visualize the PCM with a traceability with original product descriptions. For each cell value, the corresponding product overview is depicted with the highlight of the feature name and the value in the text.

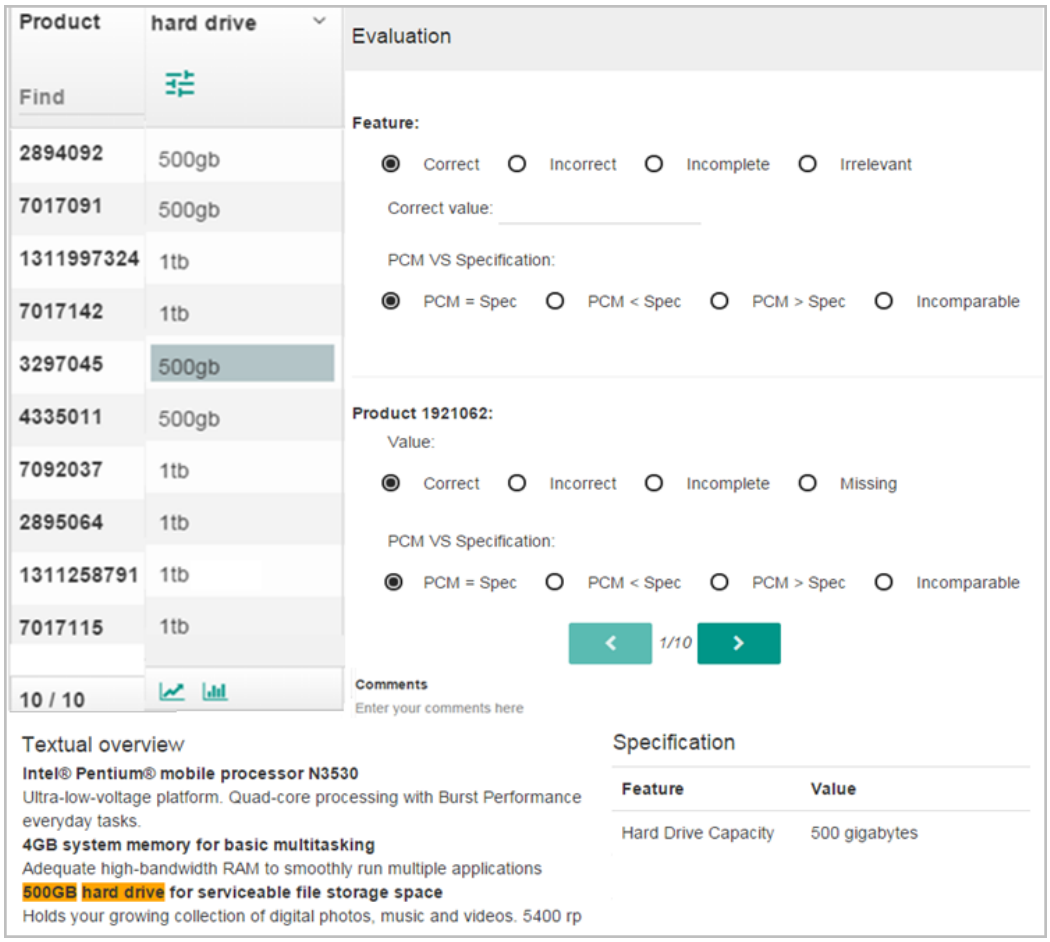


Figure 4.9: Overview of the environment during PCMs evaluation (by column)

For each displayed column, the checking process consists of:

1. Looking at the feature, the evaluators have to state whether the feature is correct, incorrect, incomplete or irrelevant.
2. Looking at each cell value, the evaluators have to state whether the expected cell value is correct, incorrect, incomplete or missing.

The evaluators can propose a correction of incorrect, incomplete or missing information. On the other hand, the evaluators have to specify for each column whether the PCM contains more/less refined information (features and cell values) than in the specification:

- PCM = Spec: the PCM and the specification contain the same information.
- PCM > Spec: the PCM contains more information comparing to the specification.
- PCM < Spec: the PCM contains less information comparing to the specification.
- incomparable: the information in the PCM and the specification do not match.

The tool offers ways to visualize the PCM with a traceability with the specification. For each cell value, the corresponding specification is depicted including the feature name and the cell value. The evaluators can add a comment at the end of the evaluation of each column.

Evaluation Scenario: We perform the evaluation by column. We display one column at a time and the evaluators have to validate or not the feature and cell values. Thus, they refer to the original text. At the same time, the evaluators have to state if the feature and cell values are more refined in the overview or the specification. This requires to refer to the corresponding specification depicted by the tool. Once the evaluation of one column is finished, the evaluator submits his/her evaluation to a database and starts again a new evaluation for a new column.

Evaluation Outputs: We obtained 118 evaluated features and 1203 evaluated cell values during an evaluation session of one hour. Overall, 50% of evaluated features belong to ranges, 24.57% come from laptops, 16.10% are related to printers, and 9.32% correspond to features of refrigerators, TV and washing machines. On the other hand, 45.95% of evaluated cell values are about ranges, 22.61% are contained in laptops PCMs, 16.90% of values belong to printers and 14.52% are related to refrigerators, TV and washing machines.

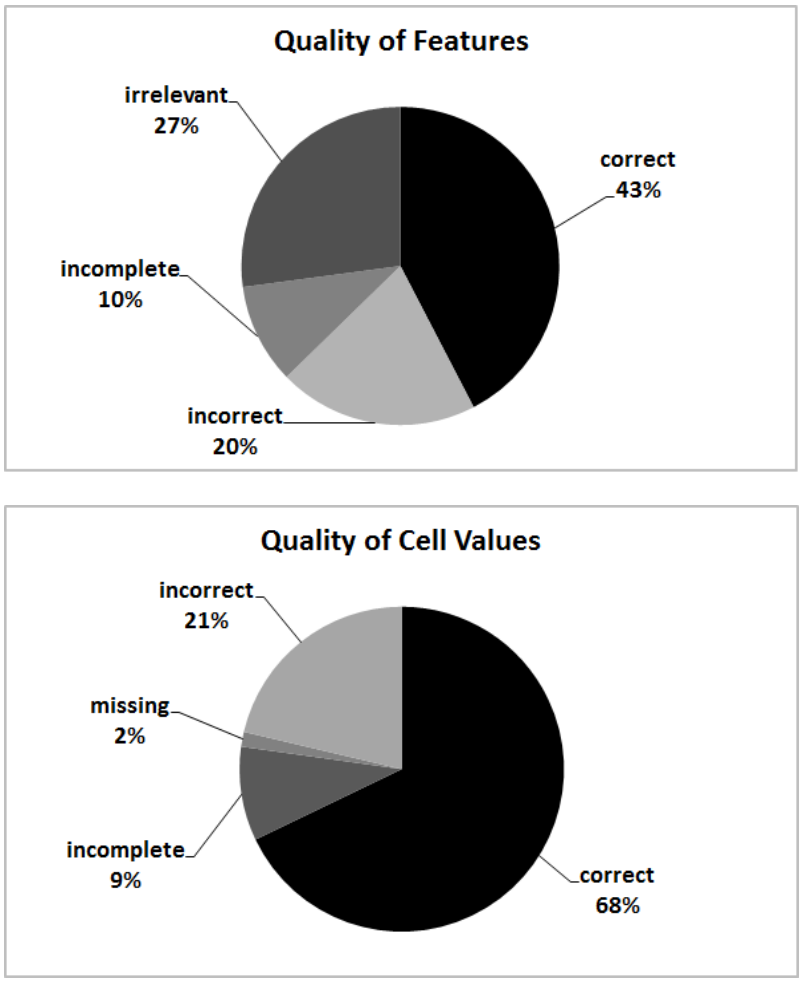


Figure 4.10: Quality of Features and Cell Values

4.8.2 RQ2.1. How effective are the techniques to fully extract a PCM from a user point of view?

Objects of Study. Our goal is to evaluate the effectiveness of a fully automated synthesis technique. The effectiveness corresponds to the percentage of correct and relevant features; and also the percentage of cell values that are correctly synthesized with respect to the original product overviews. When the whole extraction process is performed in a single step, without any user intervention, the resulting PCM may be far from the ground truth. This question aims at quantifying this potential effect.

Experimental Results. Results are reported in Figure 4.10 and show that our automatic approach retrieves 43% of correct features and 68% of correct cell values in one step and without any user intervention, showing the usefulness of our approach. We also note that 10% of features and 9% of cell values are incomplete which means that are correct but are not enough precise. This means that we are very close to the right values. Using the traceability with the original text, the user can easily retrieve the full information and complete the PCM.

Only 20% of features and 21% of cell values are incorrect with 2% of these latters are missing. In the same time, we observe that 27% of features extracted automatically are irrelevant (one cannot objectively know the preferred features for a user). Again, the results provide evidence that the role of the user remains crucial. Indeed, the user is able to correct or complete the information in the PCM thanks to the traceability with the original product descriptions and the specifications. Also, he/she can remove the features which he/she consider irrelevant.

4.8.3 RQ2.2. How effective is the overlap between overview PCMs and specification PCMs?

Objects of Study. We aim to evaluate the quality of the overlap between overview PCMs and specification PCMs. The effectiveness here is given by the percentage of overlap where information (features and cell values) in the synthesized PCM (overview PCM) are more refined than those in the specification (specification PCM).

Experimental Results. We compared the features and the cell values for the same set of products in both synthesized PCMs and the specifications. Figure 4.11 shows that regarding 56% (resp. 71%) of the total features (resp. cell values), we have as much or more information in the PCMs than in the specifications.

In particular, the PCMs outperform the specifications with 39% more refined features, while these latters contain only 24% more refined features than the PCMs. We

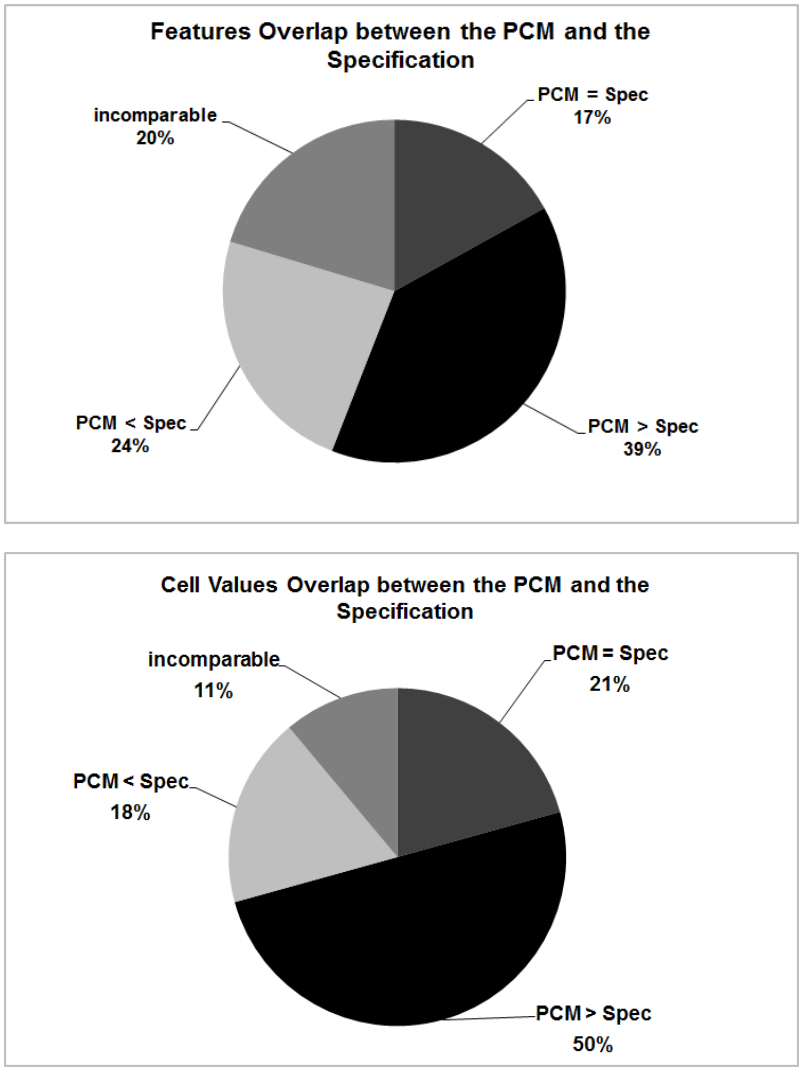


Figure 4.11: Overlap between PCMs and the Specifications

reported that 17% of features are equal in both PCMs and the specifications. Concerning cell values, PCMs are more accurate than the specifications in 50% of cases and equal to the specifications in 21% of cases. Only 18% of cell values are more detailed in the specifications.

Furthermore, we report that 20% of features and 11% of cell values are incomparable which means that the information are different in the PCMs and the specifications. These results are of great interest for the user since he/she can get a complete view when merging these incomparable information, and thus can maintain or refine the information in the resulting PCMs. This shows the importance of exploiting the text.

Key findings for RQ2.1 and RQ2.2

- Our automatic approach retrieves 43% of correct features and 68% of correct cell values in one step and without any user intervention.
- Regarding a significant portion of features (56%) and cell values (71%), we have as much or more information in the generated PCMs from product descriptions using our tool, than in the specifications.

4.9 Threats to Validity

An *external threat* to validity is that we have only applied our procedure to the Bestbuy dataset. We considered numerous categories and products to diversify the textual corpus. Yet we cannot generalize and claim that our approach is applicable to any Web sites exhibiting informal product descriptions. In fact an interesting research direction is to characterize the usefulness of our techniques w.r.t. the nature of textual artefacts and sets of product descriptions. We plan to consider other publicly available product descriptions. Our approach is independent from Bestbuy and can be technically configured for other Websites.

There are *internal threats* to validity. A first internal threat comes from the manual optimization of the clustering thresholds (regarding terms and information) for the evaluation of the heuristic. Another set of thresholds could generate less favorable results. Similarly, a manual optimization of top lists thresholds according to C-NC Value or domain-specificity metrics, might affect the quality of the domain specific terms. Also, the quality of information extracted from the text could be impacted if we consider another set of C-NC Value thresholds.

Second, the computation of overlapping parts between the specifications and the overviews is based on an equivalence between features names and cell values (see RQ1.3).

We chose a simple measurement based on occurrence of names to reduce the false positives. A more sophisticated measure (*e.g.*based on synonyms) could identify more overlapping information with the risk of providing false positives.

Besides we chose to consider only 10 products. The main rationale is that we wanted to obtain compact PCMs with numerous comparable information. Another number of products might give other results. A big number of products could increase the complexity of the PCM and a very small number of products could lead to few comparable information. It is an open problem to determine for which number the approach is applicable and useful.

The final threat we discussed here is related to the evaluation process in the user study. It is not always evident for the evaluators to decide whether the synthesized PCM has more or less information than in the specification. In some cases, the evaluator could find more and different refined information regarding a same feature in the two sides.

4.10 Conclusion

In this chapter, we described our proposed approach to synthesize product comparison matrices from informal product descriptions. We developed an automated process, based on terminology extraction, information extraction, terms clustering and information clustering. Our approach is able to mine relevant features and their values (boolean, numerical or descriptive) from the text and provide the user with a compact, synthetic, and structured view of a product line. It is then immediate to identify recurrent features and understand the differences between products.

Our empirical study showed that the resulting PCMs exhibit numerous quantitative and comparable information: 12.5% of quantified features, 15.6% of descriptive features and only 13% of empty cells. The user study showed that our automatic approach retrieves 43% of correct features and 68% of correct values in one step and without any user intervention. On the other hand, we investigate the complementarity aspect between products descriptions and technical specifications. Regarding a significant portion of features (56%) and values (71%), we obtained as much or more information in the generated PCMs than in the specifications.

Our tool provides the ability for tracing products, features and values of a PCM to the original product descriptions and technical specifications. Likewise users can understand, control and refine the information of the synthesized PCM within the context of product descriptions and specifications. The next chapter offers a comparison, lessons learned and discussion regarding the two case studies.

Chapter 5

Two Case Studies: Comparison, Lessons Learned and Discussion

Without a thorough understanding of the differences between the kind of input texts and variability models in each case study, deciding between variability extraction techniques is difficult. In this chapter, we compare the two case studies along four dimensions which are the nature of the input text (Section 5.1), the kind of variability model and its exploitable potential (Section 5.2), the adequate NLP techniques applied to mine variability (Section 5.3) and the kind of the provided traceability (Section 5.4). Table 5.1 summarizes the comparison between the two case studies.

5.1 Input Documentation

We will focus first on the type of text considered in the two case studies. As a reminder, the first case study handles regulatory requirements for safety systems certification in nuclear domain while the second case study deals with publicly available product descriptions found in online product repositories and marketing websites.

The regulatory requirements are provided in large and heterogeneous documents: regulatory documents, guides, standards and even tacit knowledge acquired from anterior projects in the past. These regulations are most often disconnected from the technical system requirements, which capture the expected system behavior. In many cases, regulatory documents provide very high level and ambiguous requirements that leave a large margin for interpretation. Worse, regulation changes over time and from one country to another. In Europe, nuclear actors mainly follow the IEC/IAEA corpus whereas in the US, IEEE/ISO standards are applied. These two corpora have been written independently from each other.

The second type of texts which are informal product descriptions describe features and benefits of products. These textual descriptions are provided in medium form (500-words description) and include technical characteristics of products. However, product descriptions lack of consistent and systematic structure to describe products, and constraints in writing these descriptions expressed in natural language.

5.2 Variability Models

In both case studies, we aim to identify and model variability from an informal text written in natural language. However, the formalization and exploitation of variability depends on the context. Indeed, to improve the certification of safety products in different countries, the major challenge is the conformance of products to multiple different regulations. This requires to tackle the problem of variability in both regulatory requirements and systems architecture. To derive an architecture that conforms to a requirements configuration, we obviously need to investigate the robustness of the architecture against requirements variability. However, when comparing publicly available related products, organizations or individuals need to capture, understand and compare the important features, differences and commonalities among them. Thus, we need to provide the reader with an accurate and synthetic data structure.

When handling variability in regulations, we choose to rely on feature models. FMs are by far the most popular notation for representing and reasoning about common and variable properties (features) of a system [AK09, BRN+13]. FMs offer a simple yet expressive way to define a set of legal *configurations* (i.e., combinations of features) [CKK06, TBK09, ACLF13]. A tree-like hierarchy and feature groups are notably used to organize features into multiple levels of increasing detail and define the ontological semantics [CKK06] of an FM. Meanwhile, when extracting variability from online product descriptions, we use product comparison matrices. Product descriptions are texts describing qualitative and quantitative features of products. To compare products on the web, PCMs give a clear and simple description of products along different features. It is then immediate to identify recurrent features and understand the differences between products. PCMs provide organizations or individuals with a synthetic, structured, and reusable model for the understanding of the differences and the comparison of products.

FMs capture features and the relationships among them. FMs address both structural and transversal relationships between features. Structural relationships exist between features and sub-features, they can be optional or mandatory, while transversal

relationships are cross-tree constraints over features which can be specified to restrict their valid combinations. PCMs contain more than simple boolean features, they also handle descriptive and numerical features. Thus, PCMs are capable to provide more complete and accurate representation when comparing products. Indeed, when using feature models to capture variability in regulatory requirements, it is easier to address variability-aware bridging of the two levels of abstraction (requirements and architecture); Meanwhile, when comparing products on the web, PCMs offer a clear *product line view* to practitioners. It is then immediate to identify recurrent features and understand the differences between products.

There is no best formalism to express variability knowledge mined from informal text, but factors that affect the choice of the formalism (nature of the text and the context including the further exploitation of the variability model). See Figure 5.1.

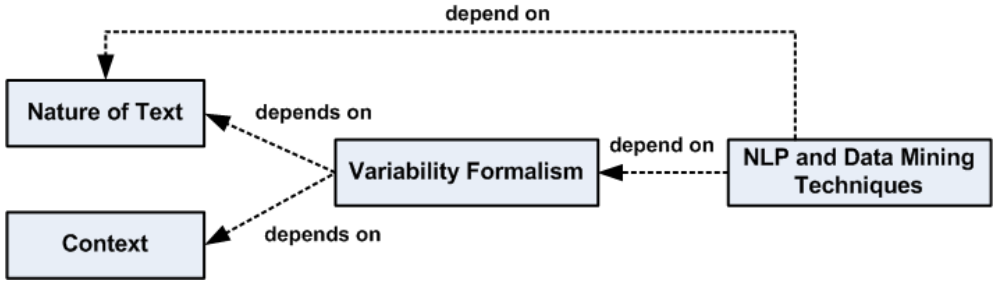


Figure 5.1: Lessons Learned

5.3 NLP and Data Mining Techniques for Mining Variability

Similarly, the techniques employed when mining variability depend on the formalism which has been considered. Indeed, when building a feature model, we need to adopt techniques capable of extracting features and their dependencies: structural dependencies to build the hierarchy (parent-child relationships, mandatory and optional relationships) and transversal dependencies (requires and exclude relationships). But when constructing a PCM, we need to apply techniques able to mine relevant features and their values (boolean, numerical or descriptive) from the text.

To automate reverse engineering FMs from regulations, we adopted information retrieval and data mining techniques to (1) extract features based on semantic analysis and requirements clustering and (2) identify features dependencies using association rules. However, to synthesize PCMs from informal product descriptions, we rely on contrastive analysis technology to mine *domain specific terms* from text, information

Table 5.1: Case Studies Comparison

Context		Nuclear power plants	Manufactured products
Criticity of the Domain		Safety-critical systems and functions, the most sophisticated and complex energy systems	Not critical
Target		Modeling variability to improve certification and safety of I&C systems in different countries	Modeling variability to compare products and select the best
Textual Input		Safety requirements: regulatory documents, guides, standards, regulatory practices	Textual product descriptions
Nature of the Input Text		Safety Requirements Informal Heterogeneous High abstraction level Disconnected from the technical system requirements	Product Descriptions Informal Not heterogeneous Not abstract Texts describing features of products including technical characteristics
Output		Feature Models	Product Comparison Matrices
Corpus Size		Huge number of requirements	Medium amount of text
Number of Products		3 Countries (France, US and UK)	10 products (total of 2692 Products)
Variability		Few variation points	Many variation points
Techniques	*Automation Level* *Feature Clustering* *Similarity* *Linguistic Filters* *Statistical Filters*	Semi-Automatic Similar Requirements Requirements Clustering Semantic (LSA) Lucene Association Rules, Apriori Algorithm, Support, Confidence, Chi square, Improvement	Automatic Similar Terms / Similar Information Terms Clustering / Information Clustering Syntactical (Levenshtein) POS Tagger, Lucene Termhood Metric (C-NC Value), Contrastive Analysis, Association Rules, Apriori Algorithm, Support
	Heuristics	Heuristics for computing requirements similarity, clusters, hierarchy, structural and transversal dependencies.	Heuristics for computing features and cell values.
Traceability		Traceability of resulting features with the original requirements. Mapping with the architecture elements.	Traceability of the synthesized PCM with the original product descriptions and technical specification for further refinement or maintenance by users.
User Effort		The expert adjusts product models by removing incorrect clusters, adding missing features and then renaming the final features. He/She also may need to maintain and refine the synthesized FM (the hierarchy and features dependencies).	The user can visualize, control and refine the information of the synthesized PCMs within the context of product descriptions and technical specifications.
Exploitability		Mapping feature model with architecture elements to derive a complying architecture.	To generate other domain models (such as feature models) To recommend features To perform automatic reasoning (e.g., multi-objective optimizations) To devise configurators or comparators

extraction, terms clustering and information clustering.

In the first case study, we apply semantic similarity to cluster tight-related requirements into features. The requirements are considered related if they concern similar matters. Yet, in the second case study, to identify a feature with its possible values, we need to adopt syntactical similarity. In particular, to extract features with descriptive values, we need to perform terms clustering while to retrieve features with quantified values, numerical information clustering is required. Elements (*i.e.* terms or information) which are not clustered will be considered as boolean features.

Overall, to extract PCMs, terms are first identified and ranked by computing a "termhood" metric, called C-NC value [BDVM10]. This metric establishes how much a word or a multi-word is likely to be conceptually independent from the context in which it appears. The contrastive analysis technology is applied to detect those terms in a document that are *specific* for the domain of the document under consideration. Inspired by the "termhood" concept, we also mine conceptually independent *numerical information* defined as domain relevant multi-word phrases containing numerical values. To identify terms (resp. information) clusters, we then compute syntactical similarity between terms (resp. information) using Levenshtein distance. We substitute the feature name and its possible values from each term (resp. information) within the cluster.

To build FM from regulations, we compute semantic similarity between requirements using LSA. Each cluster of similar requirements form a feature. To identify features dependencies, we use the Apriori Algorithm [AIS93] that is supported on frequent item sets. We rely on statistical measures to assess the quality of the extracted rules. In particular, we consider support, confidence, improvements and chi-square to compute structural dependencies and transversal dependencies.

There is no best NLP technique to apply when mining variability knowledge from textual artifacts. Actually, NLP and data mining techniques depend on which variability formalism we relied on and which kind of text we are dealing with (see Figure 5.1).

5.4 Traceability

This section illustrates the exploitability of the variability model in each context. In nuclear context, we offer two kinds of traceability:

Traceability with the original regulations: When building requirements variability model, we keep the traceability between the identified features and the original regulatory requirements. Indeed, features provide an abstraction of requirements. Every

feature covers a particular set of requirements which refine that feature. Feature models are domain models which structure requirements by mapping them to a feature and by forming relations between them. In this way, domain experts can validate the final feature model against regulations.

Mapping with the architecture elements: Feature models structure traceability links between requirements and the architecture. Modeling requirements variability is useful, however there is no direct mapping from requirements to the architecture. To bridge the gap between textual regulatory requirements and the architecture, we move towards variability in design rules. In the same time we keep the traceability between identified features and the original design rules. Our industrial partners rely on these rules to validate the architecture against regulations. This way allows us to bind a requirements variability model and an architecture variability model in order to derive an architecture that conforms to a requirements configuration.

Now regarding product descriptions case, we provide the ability to tracing products and features of a PCM to the *original product descriptions* and also *technical specifications* for further refinement or maintenance by users.

Traceability with the original product descriptions: Users can exploit *MatrixMiner* to visualize the matrix through a Web editor and review, refine, or complement the cell values based on the information contained in the text.

Traceability with the technical specifications: Similarly, our tool provides the ability to visualize the resulting PCM in the context of the technical specification typically to control or refine the synthesized information. In particular, our qualitative review shows that there is also a potential that technical specifications complement product description. So that user can find more detailed information in the specification.

5.5 Conclusion

The main lesson learnt from the two case studies is that the exploitability and the extraction of variability knowledge depends on the context, the nature of variability and the nature of text. In particular, the formalism to express variability depends on the context and the techniques employed when mining variability depend on the formalism. In this chapter, we compared the two case studies along four dimensions which are the nature of documentation, the type of variability model and its exploitability, the NLP and data mining techniques used for mining variability and finally the kind of traceability.

Part III

Conclusion and Perspectives

Chapter 6

Conclusion and Perspectives

In this chapter, we first summarize all the contributions of this thesis, recalling the challenges and how we addressed each of them. Next and finally, we discuss some perspectives for future research.

6.1 Conclusion

Domain analysis is the process of analyzing a family of products to identify their common and variable features. Domain analysis involves not only looking at standard requirements documents (e.g., use case specifications) but also regulatory documents, product descriptions, customer information packs, market analysis, etc. Looking across all these documents and deriving, in a practical and scalable way, a variability model that is comprised of coherent abstractions is a fundamental and non-trivial challenge.

Numerous approaches have been proposed to mine variability and support domain analysis. However, few of them adopt automated techniques for the construction of variability models from unstructured and ambiguous documents. Such techniques are essential for both feasibility and scalability of approaches, since many potentially large informal documents may be given as input to domain analysis, making a manual analysis of these documents time consuming or even prohibitive.

In this thesis, we have conducted two case studies on leveraging Natural Language Processing (NLP) and data mining techniques for achieving scalable identification of commonalities and variabilities from informal documentation. Accordingly, we considered two different contexts: (1) reverse engineering *Feature Models (FMs)* from regulatory requirements in nuclear domain and (2) synthesizing *Product Comparison Matrices (PCMs)* from informal product descriptions. The first case study handles regulatory requirements for safety systems certification in nuclear domain. In the specific context

of nuclear energy, one applicant has to deal with very heterogeneous regulations and practices, varying from one country to another. Our purpose was to automate the arduous task of manually building a feature model from regulatory requirements. The second case study deals with publicly available product descriptions found in online product repositories and marketing websites. Numerous organizations or individuals rely on these textual descriptions for analyzing a domain and a set of related products. Our goal was to automate the daunting task of manually analyzing and reviewing each product description and provide a reader with an accurate and synthetic PCM.

Our first contribution is a **semi-automated approach to reverse engineering feature models from regulatory requirements**. These regulations are provided in large and heterogeneous documents such as regulatory documents, guides, standards, etc. We adopted NLP and data mining techniques to extract features based on semantic analysis and requirements clustering; and identify features dependencies using association rules. The evaluation showed the effectiveness of our automated techniques to synthesize a meaningful feature model. In particular, the approach is able to retrieve 69% of correct clusters. We also noticed that structural dependencies show a high predictive capacity: 95% of the mandatory relationships and 60% of optional relationships are found. Furthermore, the totality of requires and exclude relationships are extracted.

Before the automatic construction of feature model, a comprehensive and in-depth analysis of regulations was required. Therefore, we performed **a manual formalization of variability in regulations**. We relied on Common Variability Language (CVL) since it is domain independent. As regulations are contained in huge amount of documents, the key concept to narrow the problem space was to analyze variability in regulatory documents by topic, in different countries and on the same abstraction level. When performing the same safety function in different countries, the variability concerns not only the set of requirements to comply with and the certification process, but also the system's architecture itself. Tracing variability from the problem space to the solution space is crucial to improve the understanding of system variability, as well as support its maintenance and evolution. For this purpose, we established a **variability-aware bridging of the two levels of abstraction** (requirements and architecture) in order to derive a complying architecture. This manual work is also a contribution that provides great value to industry partners and that introduces formal variability modeling in their engineering processes.

Our second contribution consists in an approach to **automate the extraction of product comparison matrices from informal descriptions of products**. We investigated the use of automated techniques for synthesizing a PCM despite the in-

formality and absence of structure in the textual descriptions. Indeed, the proposed method automates the identification of features, their values, and collects information from each product to deliver a compact, synthetic, and structured view of a product line. The approach is based on a contrastive analysis technology to mine domain specific terms from text, information extraction, terms clustering and information clustering. Overall, our empirical study revealed the powerful capabilities of our approach to deliver PCMs with rich and diversified information while keeping PCMs compact. In fact, the resulting PCMs include numerous quantitative and comparable information (12.5% of quantified features and 15.6% of descriptive features) with only 13% of empty cells. The user study showed the effectiveness and usefulness of our automatic approach. Actually, this latter can retrieve 43% of correct features and 68% of correct values in one step and without any user intervention.

Another interesting observation was the complementarity aspect that might exist between product overviews and product specifications. Our user study offered evidence that PCMs generated from product descriptions outperform the specifications. Indeed, regarding a significant portion of features (56%) and values (71%), we have as much or more information in the generated PCMs than in the specifications. We showed that there is a potential to complement or even refine technical information of products.

The main lesson learnt from the two case studies is that three key factors affect the choice of techniques to apply when mining and exploiting variability knowledge from informal documentation. These factors are: the context, the nature of variability and the nature of text. Specifically, formalizing variability depends on the nature of the input text and the context while the choice of NLP and data mining techniques, employed when mining variability, are influenced by the choice of the formalism and the kind of text.

As a conclusion, we provided efficient approaches to leverage natural language processing and data mining techniques for achieving scalable extraction of commonalities and variabilities from informal documentation in two different contexts. As a future work, we plan to apply, possibly adapt, and evaluate similar automated techniques for mining variability in other artefacts and contexts. The following section discusses some perspectives for future research.

6.2 Perspectives

In this section, we present some long- and short-term ideas for research around the contributions of this thesis. We will first outline the general perspectives and then

enumerate some future works for each case study explored in the thesis. The overall perspectives are summarized in Figure 6.1.

The short-term perspective of this thesis is to **explore automatic extraction and formalization of variability from other kinds of informal documentation** such as market analysis, customer information packs, functional requirements, configuration files, source code, etc; (see Figure 6.1, Ⓐ). By exploiting the latest techniques in human-language technology and computational linguistics and combining them with the latest methods in machine learning and traditional data mining, one can effectively mine useful and important knowledge from the continually growing body of electronic documents and web pages. We also aim to identify other possible key factors that might affect the variability extraction procedure.

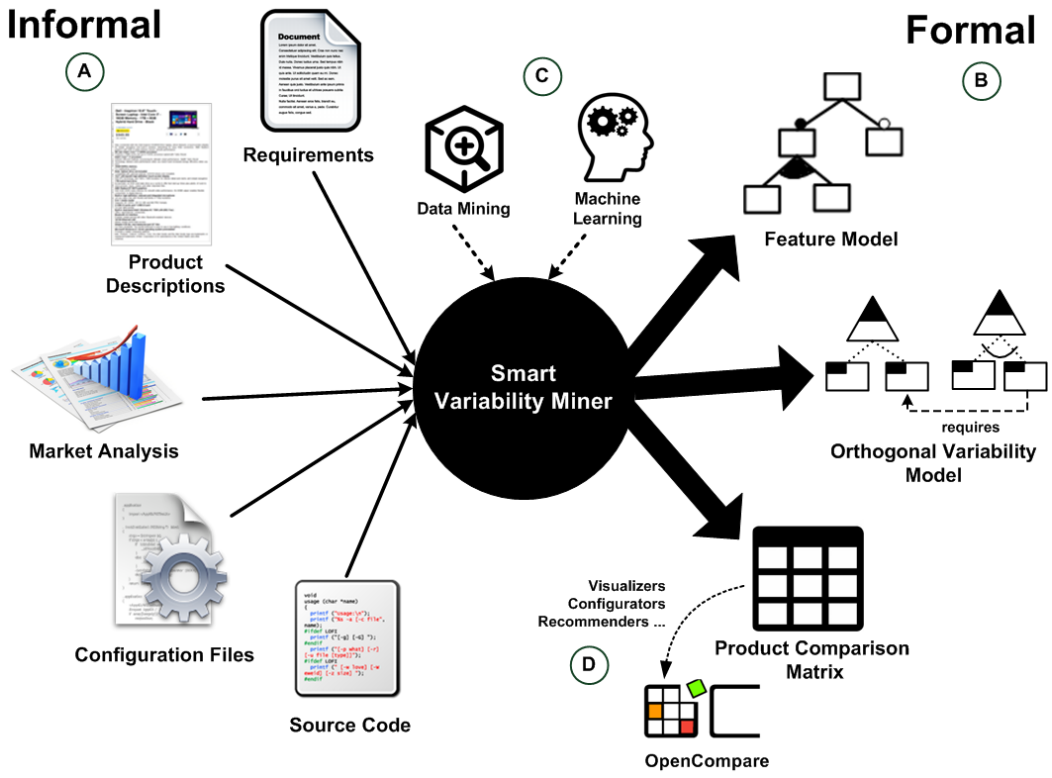


Figure 6.1: Context-independent methodology for mining and modeling variability from informal documentation

The long-term challenge is to generalize the extraction of variability knowledge from textual artifacts. In other words, we want to **deal with context-independent methodology for mining and modeling variability from informal documentation**. The

context here refers to the nature of the text and the variability formalism. The idea behind this is that given any textual artifacts (requirements, product descriptions, configuration files, source code, etc.) of a family of products and any variability formalism (feature model, product comparison matrix, orthogonal variability model, decision model, etc.), the tool could apply suitable mining techniques to generate automatically a meaningful variability model (see Figure 6.1, Ⓐ, Ⓑ and Ⓒ). A non-trivial task is to reveal the generic parameters of the automated procedure for retrieving and modeling variability, such as mandatory elements, optional elements, constraints over elements, etc.

In the reminder of this chapter we will describe possible improvements and extensions to the contributions of this thesis. Regarding reverse engineering feature models from regulations (Chapter 3), we propose these roadmaps for further research to enhance the quality of feature models.

Improve the determination of requirements similarity. Latent semantic analysis (LSA) determines relationship among the requirements, but assume a flat structure, *i.e.*, also known as bag of words. Although the requirement documents considered in the evaluation of the approach are textual, they still have latent structure, such as hierarchy and variability relationship between requirements, and proximity structure, *i.e.*, similar requirements are physically closer in the document. Such latent structure could be used to improve requirement similarity determination.

Make feature naming scalable. We note that due to the agglomerative nature of the clustering algorithm, features closer to the root comprise an increasingly high number of requirements. Therefore, naming them is a non-trivial task. Naming and finding a semantic definition of features that summarizes all its encompassing requirements should be addressed with a scalable approach. We suggest extensions to our approach in order to tackle this issue.

Our initial suggestion is to use Wmatrix [SRC05], an NLP tool that determines semantic, part-of-speech, and frequency information of words in text. Our second suggestion is to select the most frequently occurring phrase from among all of the requirements in the cluster [DDH+13b]. For this, we propose to apply Stanford Part-of-Speech (POS) tagger to tag each term in the requirements with its POS in order to retain only nouns, adjectives, and verbs, and then mining frequent itemsets using the Apriori or FPGrowth [HPY00] algorithms.

Deal with more complex relationships and constraints in the target feature model. Our experience showed that there is a need for a process which helps the analyst in assigning

the group cardinality value. It is interesting for the analyst to have a tool that allows him to estimate the cardinality for each optional bundle. There is also a need for techniques able to deal with more complex than Boolean-type features, as for instance, features with multiple instantiations. How can these be specified? Remain still an open question for future researches. Several other fundamental questions are still open and their solutions are envisaged for future works. For instance: How to deal with more complex constraints? What statistical tools could be used to support the aforementioned questions?

Automate tracing variability across problem space and solution space. We are currently improving the different variability modeling tools of both parts - requirements and architecture - of the project. We plan to further exploit traceability links in order to reason about the conformance between regulatory requirements and the architecture of safety systems. As future work, we aim to investigate the use of automated techniques to trace variability across these two levels of abstraction. Specifically, we need to: (1) automate traceability between requirements FM and design rules FM through features which correspond respectively to clusters of similar requirements and clusters of similar design rules. This is feasible since a design rule that belongs to a cluster in design rules FM can satisfy fully or partially one or more requirements contained in one or several clusters in requirements FM; (2) automate the mapping between design rules feature model and the architecture product line model.

Regarding the automated extraction of PCMs from informal product descriptions (Chapter 4), we identify the following perspectives.

Mine knowledge from text using template filling. We believe that information extraction have an enormous potential still to be explored. Traditionally information extraction tasks assume that the structures to be extracted are well defined. In some scenarios, e.g. in product descriptions, we do not know in advance the structures of the information we would like to extract and would like to mine such structures from large corpora. To alleviate this problem, recently there has been an increasing amount of interest in unsupervised information extraction from large corpora. We aim to enhance the extraction of information from product descriptions using automatically template induction from an unlabeled corpus and template filling.

Apply the approach on different websites other than BestBuy. To generalize and claim that our approach is applicable to any website exhibiting informal product descriptions, a short-term perspective is to apply our procedure on other websites than BestBuy. In fact an interesting research direction is to characterize the effectiveness of our techniques

w.r.t. the nature of textual artefacts and sets of product descriptions. We plan to consider other publicly available product descriptions. Our approach is independent from Bestbuy and can be technically configured for other websites.

Integrate the tool-supported approach as part of OpenCompare. The presented work has the potential to crawl scattered and informal product descriptions that abound on the web. We are integrating the tool-supported approach as part of OpenCompare an initiative for the collaborative edition, the sharing, the standardization, and the open exploitation of PCMs. The goal is to provide an integrated set of tools (e.g., APIs, visualizers, configurators, recommenders, editors) for democratizing their creation, import, maintenance, and exploitation (see Figure 6.1, Ⓓ).

Bibliography

[AAH05] Jean-Raymond Abrial, Jean-Raymond Abrial, and A Hoare. *The B-book: assigning programs to meanings*. Cambridge University Press, 2005.

[AB04] Alain Abran and Pierre Bourque. *SWEBOK: Guide to the software engineering Body of Knowledge*. IEEE Computer Society, 2004.

[AB11] Andrea Arcuri and Lionel Briand. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *Software Engineering (ICSE), 2011 33rd International Conference on*, pages 1–10. IEEE, 2011.

[ABH+13a] Mathieu Acher, Benoit Baudry, Patrick Heymans, Anthony Cleve, and Jean-Luc Hainaut. Support for reverse engineering and maintaining feature models. In *VaMoS'13*, page 20. ACM, 2013.

[ABH+13b] Mathieu Acher, Benoit Baudry, Patrick Heymans, Anthony Cleve, and Jean-Luc Hainaut. Support for reverse engineering and maintaining feature models. In Stefania Gnesi, Philippe Collet, and Klaus Schmid, editors, *VaMoS*, page 20. ACM, 2013.

[ACLF13] Mathieu Acher, Philippe Collet, Philippe Lahire, and Robert France. Familiar: A domain-specific language for large scale management of feature models. *Science of Computer Programming (SCP) Special issue on programming languages*, page 22, 2013.

[ACP+12a] Mathieu Acher, Anthony Cleve, Gilles Perrouin, Patrick Heymans, Charles Vanbeneden, Philippe Collet, and Philippe Lahire. On extracting feature models from product descriptions. In *VaMoS'12*, pages 45–54. ACM, 2012.

[ACP+12b] Mathieu Acher, Anthony Cleve, Gilles Perrouin, Patrick Heymans, Charles Vanbeneden, Philippe Collet, and Philippe Lahire. On extracting feature models from product descriptions. In *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*, pages 45–54. ACM, 2012.

[ACSW12] Nele Andersen, Krzysztof Czarnecki, Steven She, and Andrzej Wasowski. Efficient synthesis of feature models. In *Proceedings of SPLC'12*, pages 97–106. ACM Press, 2012.

[AE06] Robin Abraham and Martin Erwig. Type inference for spreadsheets. In *Proceedings of the 8th ACM SIGPLAN international conference on Principles and practice of declarative programming*, pages 73–84. ACM, 2006.

[AE07] Robin Abraham and Martin Erwig. Ucheck: A spreadsheet type checker for end users. *Journal of Visual Languages & Computing*, 18(1):71–95, 2007.

[AIS93] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.

[AK04] Bruno Agard and Andrew Kusiak*. Data-mining-based methodology for the design of product families. *International Journal of Production Research*, 42(15):2955–2969, 2004.

[AK09] Sven Apel and Christian Kästner. An overview of feature-oriented software development. *Journal of Object Technology (JOT)*, 8(5):49–84, July/August 2009.

[AMS06] Timo Asikainen, Tomi Mannisto, and Timo Soininen. A unified conceptual foundation for feature modelling. In *Software Product Line Conference, 2006 10th International*, pages 31–40. IEEE, 2006.

[AMS07] Timo Asikainen, Tomi Männistö, and Timo Soininen. Kumbang: A domain ontology for modelling variability in software product families. *Advanced Engineering Informatics*, 21(1):23–40, 2007.

[AOAB05] Turky N Al-Otaiby, Mohsen AlSherif, and Walter P Bond. Toward software requirements modularization using hierarchical clustering tech-

niques. In *Proceedings of the 43rd annual Southeast regional conference-Volume 2*, pages 223–228. ACM, 2005.

[AS+94] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[ASB+08a] Vander Alves, Christa Schwanninger, Luciano Barbosa, Awais Rashid, Peter Sawyer, Paul Rayson, Christoph Pohl, and Andreas Rummler. An exploratory study of information retrieval techniques in domain analysis. In *SPLC'08*, pages 67–76, 2008.

[ASB+08b] Vander Alves, Christa Schwanninger, Luciano Barbosa, Awais Rashid, Peter Sawyer, Paul Rayson, Christoph Pohl, and Andreas Rummler. An exploratory study of information retrieval techniques in domain analysis. In *Software Product Line Conference, 2008. SPLC'08. 12th International*, pages 67–76. IEEE, 2008.

[ASB14] Morayo Adedjouma, Mehrdad Sabetzadeh, and Lionel C Briand. Automated detection and resolution of legal cross references: Approach and a study of luxembourg's legislation. In *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pages 63–72. IEEE, 2014.

[ASC07] Jim Azar, Randy K Smith, and David Cordes. Value-oriented requirements prioritization in a small development organization. *Software, IEEE*, 24(1):32–37, 2007.

[BA07] Travis D. Breaux and Annie I. Anton. A systematic method for acquiring regulatory requirements: A frame-based approach. In *RHAS-6*, Pittsburgh, PA, USA, September 2007. Software Engineering Institute (SEI).

[BA+08] Travis D Breaux, Annie Antón, et al. Analyzing regulatory rules for privacy and security requirements. *Software Engineering, IEEE Transactions on*, 34(1):5–20, 2008.

[BABN15] Guillaume Bécan, Mathieu Acher, Benoit Baudry, and SanaBen Nasr. Breathing ontological knowledge into feature model synthesis: an empirical study. *Empirical Software Engineering*, pages 1–48, 2015.

[BB01] Felix Bachmann and Len Bass. Managing variability in software architectures. *ACM SIGSOFT Software Engineering Notes*, 26(3):126–132, 2001.

[BB05] Michael W Berry and Murray Browne. *Understanding search engines: mathematical modeling and text retrieval*, volume 17. Siam, 2005.

[BCK03] L Bass, P Clements, and R Kazman. Software architecture in practice, 2nd edn. sei series in software engineering, 2003.

[BDVM10] Francesca Bonin, Felice Dell'Orletta, Giulia Venturi, and Simonetta Montemagni. A contrastive approach to multi-word term extraction from domain corpora. In *Proceedings of the "7th International Conference on Language Resources and Evaluation", Malta*, pages 19–21, 2010.

[BEG12] Ebrahim Bagheri, Faezeh Ensan, and Dragan Gasevic. Decision support for the software product line domain engineering lifecycle. *Automated Software Engineering*, 19(3):335–377, 2012.

[BFG+02] Jan Bosch, Gert Florijn, Danny Greefhorst, Juha Kuusela, J Henk Obbink, and Klaus Pohl. Variability issues in software product lines. In *Software Product-Family Engineering*, pages 13–21. Springer, 2002.

[BFJZ14] Kevin M Benner, Martin S Feather, W Lewis Johnson, and Lorna A Zorman. Utilizing scenarios in the software development process. *Information system development process*, 30:117–134, 2014.

[BGCH10] Brian Berenbach, D Gruseman, and Jane Cleland-Huang. Application of just in time tracing to regulatory codes. In *Proceedings of the Conference on Systems Engineering Research*, 2010.

[BI96] Barry Boehm and Hoh In. Identifying quality-requirement conflicts. *IEEE software*, (2):25–35, 1996.

[BJA99] Roberto J Bayardo Jr and Rakesh Agrawal. Mining the most interesting rules. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 145–154. ACM, 1999.

[BJAG99] Roberto J Bayardo Jr, Rakesh Agrawal, and Dimitrios Gunopulos. Constraint-based rule mining in large, dense databases. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 188–197. IEEE, 1999.

[BM04] Razvan Bunescu and Raymond J Mooney. Collective information extraction with relational markov networks. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 438. Association for Computational Linguistics, 2004.

[BMPZ01] Roberto Basili, Alessandro Moschitti, Maria Teresa Pazienza, and Fabio Massimo Zanzotto. A contrastive approach to term extraction. In *Terminologie et intelligence artificielle. Rencontres*, pages 119–128, 2001.

[BNBA+15] Sana Ben Nasr, Guillaume Bécan, Mathieu Acher, João Bosco Ferreira Filho, Benoit Baudry, Nicolas Sannier, and Jean-Marc Davril. Matrixminer: a red pill to architect informal product descriptions in the matrix. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 982–985. ACM, 2015.

[Boo67] Taylor L Booth. *Sequential machines and automata theory*, volume 3. Wiley New York, 1967.

[BPSP04] Danilo Beuche, Holger Papajewski, and Wolfgang Schröder-Preikschat. Variability management with feature models. *Science of Computer Programming*, 53(3):333–352, 2004.

[BPZ01] Roberto Basili, Maria Teresa Pazienza, and Fabio Massimo Zanzotto. Modelling syntactic context in automatic term extraction. In *In Proc. of Recent Advances in Natural Language Processing (RANLP'01), Tzigov Chark*. Citeseer, 2001.

[BR89] Barry W Boehm and Rony Ross. Theory-w software project management principles and examples. *Software Engineering, IEEE Transactions on*, 15(7):902–916, 1989.

[BR02] Glenn J Browne and Venkataraman Ramesh. Improving information requirements determination: a cognitive perspective. *Information & Management*, 39(8):625–645, 2002.

[Bra90] John W Brackett. Software requirements. Technical report, DTIC Document, 1990.

[Bra91] Victor L Brailovsky. A probabilistic approach to clustering. *Pattern Recognition Letters*, 12(4):193–198, 1991.

[Bre09] Travis Durand Breaux. *Legal requirements acquisition for the specification of legally compliant information systems*. ProQuest, 2009.

[BRN+13] Thorsten Berger, Ralf Rublack, Divya Nair, Joanne M Atlee, Martin Becker, Krzysztof Czarnecki, and Andrzej Wasowski. A survey of variability modeling in industrial practice. In *VAMOS'13*, page 7. ACM, 2013.

[BSA+14] Guillaume Bécan, Nicolas Sannier, Mathieu Acher, Olivier Barais, Arnaud Blouin, and Benoit Baudry. Automating the formalization of product comparison matrices. In *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*, pages 433–444. ACM, 2014.

[BSL+10] Thorsten Berger, Steven She, Rafael Lotufo, Krzysztof Czarnecki, and Andrzej Wasowski. Feature-to-code mapping in two large product lines. In *SPLC*, pages 498–499. Citeseer, 2010.

[BSRC10] David Benavides, Sergio Segura, and Antonio Ruiz-Cortés. Automated analysis of feature models 20 years later: A literature review. *Information Systems*, 35(6):615–636, 2010.

[BT06] Don Batory and Sahil Thaker. *Towards safe composition of product lines*. Citeseer, 2006.

[BTRC05] David Benavides, Pablo Trinidad, and Antonio Ruiz-Cortés. Automated reasoning on feature models. In *Advanced Information Systems Engineering*, pages 491–503. Springer, 2005.

[BWL12] Lidong Bing, Tak-Lam Wong, and Wai Lam. Unsupervised extraction of popular product attributes from web sites. In *Information Retrieval Technology*, pages 437–446. Springer, 2012.

[CABA09] Lianping Chen, Muhammad Ali Babar, and Nour Ali. Variability management in software product lines: a systematic review. In *Proceedings of the 13th International Software Product Line Conference*, pages 81–90. Carnegie Mellon University, 2009.

[CB98] IEEE Computer Society. Software Engineering Standards Committee and IEEE-SA Standards Board. Ieee recommended practice for software requirements specifications. Institute of Electrical and Electronics Engineers, 1998.

[CBK13] Rafael Capilla, Jan Bosch, and Kyo-Chul Kang. *Systems and Software Variability Management.* Springer, 2013.

[CBUE02] Krzysztof Czarnecki, Thomas Bednasch, Peter Unger, and Ulrich Eisenecker. Generative programming for embedded software: An industrial experience report. In *Generative Programming and Component Engineering*, pages 156–172. Springer, 2002.

[CCCB+08] Marie Calberg-Challot, Danielle Candel, Didier Bourigault, Xavier Dumont, John Humbley, and Jacques Joseph. Une analyse méthodique pour l'extraction terminologique dans le domaine du nucléaire. *Terminology*, 14(2):183–203, 2008.

[CE00] Krzysztof Czarnecki and Ulrich W Eisenecker. Generative programming. *Edited by G. Goos, J. Hartmanis, and J. van Leeuwen*, page 15, 2000.

[CE09] Chris Chambers and Martin Erwig. Automatic detection of dimension errors in spreadsheets. *Journal of Visual Languages & Computing*, 20(4):269–283, 2009.

[CES10] Jácome Cunha, Martin Erwig, and Joao Saraiva. Automatically inferring classsheet models from spreadsheets. In *Visual Languages and Human-Centric Computing (VL/HCC), 2010 IEEE Symposium on*, pages 93–100. IEEE, 2010.

[CFMS12] Jácome Cunha, João Paulo Fernandes, Jorge Mendes, and João Saraiva. Mdsheet: A framework for model-driven spreadsheet engineering. In *Proceedings of the 34th International Conference on Software Engineering*, pages 1395–1398. IEEE Press, 2012.

[CFRS12] Jácome Cunha, João P Fernandes, Hugo Ribeiro, and João Saraiva. Towards a catalog of spreadsheet smells. In *Computational Science and Its Applications–ICCSA 2012*, pages 202–216. Springer, 2012.

[CGR+12] Krzysztof Czarnecki, Paul Grünbacher, Rick Rabiser, Klaus Schmid, and Andrzej Wąsowski. Cool features and tough decisions: a comparison of variability modeling approaches. In *Proceedings of the sixth international workshop on variability modeling of software-intensive systems*, pages 173–182. ACM, 2012.

[CHCGE10] Jane Cleland-Huang, Adam Czauderna, Marek Gibiec, and John Emenecker. A machine learning approach for tracing regulatory codes to product specific requirements. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pages 155–164. ACM, 2010.

[CHDCHM08] Carlos Castro-Herrera, Chuan Duan, Jane Cleland-Huang, and Bamshad Mobasher. Using data mining and recommender systems to facilitate large-scale, open, and inclusive requirements elicitation processes. In *International Requirements Engineering, 2008. RE'08. 16th IEEE*, pages 165–168. IEEE, 2008.

[Che76] Peter Pin-Shan Chen. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, 1(1):9–36, 1976.

[CHE05] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. Staged configuration through specialization and multilevel configuration of feature models. *Software Process: Improvement and Practice*, 10(2):143–169, 2005.

[CHKK06] Krzysztof Czarnecki, Chang Hwan, Peter Kim, and KT Kalleberg. Feature models are views on ontologies. In *Software Product Line Conference, 2006 10th International*, pages 41–51. IEEE, 2006.

[CJ08] Nathanael Chambers and Daniel Jurafsky. Unsupervised learning of narrative event chains. In *ACL*, volume 94305, pages 789–797. Citeseer, 2008.

[CJ11] Nathanael Chambers and Dan Jurafsky. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 976–986. Association for Computational Linguistics, 2011.

[CK05] Krzysztof Czarnecki and Chang H. Kim. Cardinality-based feature modeling and constraints: A progress report, October 2005.

[CKK06] Krzysztof Czarnecki, Chang Hwan Peter Kim, and Karl Trygve Kalleberg. Feature models are views on ontologies. In *SPLC '06*, pages 41–51. IEEE, 2006.

[CKPT92] Douglass R Cutting, David R Karger, Jan O Pedersen, and John W Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–329. ACM, 1992.

[CLH93] Nancy Chinchor, David D Lewis, and Lynette Hirschman. Evaluating message understanding systems: an analysis of the third message understanding conference (muc-3). *Computational linguistics*, 19(3):409–449, 1993.

[CN02] Paul Clements and Linda Northrop. Software product lines: practices and patterns. 2002.

[CN04] Teresa Mihwa Chung and Paul Nation. Identifying technical vocabulary. *System*, 32(2):251–263, 2004.

[CNL03] Hai Leong Chieu, Hwee Tou Ng, and Yoong Keok Lee. Closing the gap: Learning-based information extraction rivaling knowledge-engineering methods. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 216–223. Association for Computational Linguistics, 2003.

[Com] Common Variability Language (CVL). `http://www.omgwiki.org/variability/doku.php`.

[CR06] Aaron Ceglar and John F Roddick. Association mining. *ACM Computing Surveys (CSUR)*, 38(2):5, 2006.

[CSW08] K. Czarnecki, S. She, and A. Wasowski. Sample spaces and feature models: There and back again. In *SPLC'08*, pages 22–31, 2008.

[CVAS11] Jácome Cunha, Joost Visser, Tiago Alves, and João Saraiva. Type-safe evolution of spreadsheets. In *Fundamental Approaches to Software Engineering*, pages 186–201. Springer, 2011.

[CW07a] Krzysztof Czarnecki and Andrzej Wasowski. Feature diagrams and logics: There and back again. In *SPLC'07*, pages 23–34. IEEE, 2007.

[CW07b] Krzysztof Czarnecki and Andrzej Wasowski. Feature diagrams and logics: There and back again. In *Software Product Line Conference, 2007. SPLC 2007. 11th International*, pages 23–34. IEEE, 2007.

[CZZM05] Kun Chen, Wei Zhang, Haiyan Zhao, and Hong Mei. An approach to constructing feature models based on requirements clustering. In *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, pages 31–40. IEEE, 2005.

[Dav97] Stanley M Davis. *Future perfect*. Basic Books, 1997.

[DDF+90] Scott Deerwester, Susan T. Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[DDH+13a] Jean-Marc Davril, Edouard Delfosse, Negar Hariri, Mathieu Acher, Jane Cleland-Huang, and Patrick Heymans. Feature model extraction from large collections of informal product descriptions. In *ESEC/FSE'13*, 2013.

[DDH+13b] Jean-Marc Davril, Edouard Delfosse, Negar Hariri, Mathieu Acher, Jane Cleland-Huang, and Patrick Heymans. Feature model extraction from large collections of informal product descriptions. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, pages 290–300. ACM, 2013.

[Del09] Felice Dell'Orletta. Ensemble system for part-of-speech tagging. *Proceedings of EVALITA*, 9, 2009.

[DGH+11] Horatiu Dumitru, Marek Gibiec, Negar Hariri, Jane Cleland-Huang, Bamshad Mobasher, Carlos Castro-Herrera, and Mehdi Mirakhorli. On-demand feature recommendations derived from mining public product descriptions. In *Software Engineering (ICSE), 2011 33rd International Conference on*, pages 181–190. IEEE, 2011.

[DGR11] Deepak Dhungana, Paul Grünbacher, and Rick Rabiser. The dopler meta-tool for decision-oriented variability modeling: a multiple case study. *Automated Software Engineering*, 18(1):77–114, 2011.

[DJ85] Vasant Dhar and Matthias Jarke. Learning from prototypes. In *Proc. of the sixth Intl. Conf. on Information Systems*, pages 114–133, 1985.

[DL98] Guozhu Dong and Jinyan Li. Interestingness of discovered association rules in terms of neighborhood-based unexpectedness. In *Research and*

Development in Knowledge Discovery and Data Mining, pages 72–86. Springer, 1998.

[dlVPW13] Jose Luis de la Vara and Rajwinder Kaur Panesar-Walawege. Safetymet: A metamodel for safety standards. In *MoDELS'2013*, pages 69–86, 2013.

[DLZ09] Xiaowen Ding, Bing Liu, and Lei Zhang. Entity discovery and assignment for opinion mining applications. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1125–1134. ACM, 2009.

[DM01] Inderjit S Dhillon and Dharmendra S Modha. Concept decompositions for large sparse text data using clustering. *Machine learning*, 42(1-2):143–175, 2001.

[DRGN07] Deepak Dhungana, Rick Rabiser, Paul Grünbacher, and Thomas Neumayer. Integrated tool support for software product line engineering. In *ASE*, pages 533–534, 2007.

[Dro03] Patrick Drouin. Term extraction using non-technical corpora as a point of leverage. *Terminology*, 9(1):99–115, 2003.

[DS06] Olfa Djebbi and Camille Salinesi. Criteria for comparing requirements variability modeling notations for product lines. In *Comparative Evaluation in Requirements Engineering, 2006. CERE'06. Fourth International Workshop on*, pages 20–35. IEEE, 2006.

[DSB04] Sybren Deelstra, Marco Sinnema, and Jan Bosch. Experiences in software product families: Problems and issues during product derivation. In *Software Product Lines*, pages 165–182. Springer, 2004.

[DSB05] Sybren Deelstra, Marco Sinnema, and Jan Bosch. Product derivation in software product families: a case study. *Journal of Systems and Software*, 74(2):173–194, 2005.

[Dum93] S Dumais. Lsi meets trec: A status report. In *Proceedings of the first Text REtrieval Conference, TREC1*, pages 137–152, 1993.

[Dum95] Susan T Dumais. Latent semantic indexing (lsi): Trec-3 report. *Nist Special Publication SP*, pages 219–219, 1995.

[Dun93] Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational linguistics*, 19(1):61–74, 1993.

[DWVE06] Emile DE, Radboud Winkels, and Tom Van Engers. Automated detection of reference structures in law. *Frontiers in Artificial Intelligence and Applications*, page 41, 2006.

[EE05] Gregor Engels and Martin Erwig. Classsheets: automatic generation of spreadsheet applications from object-oriented specifications. In *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, pages 124–133. ACM, 2005.

[EFM+99] Wolfgang Emmerich, Anthony Finkelstein, Carlo Montangero, Stefano Antonelli, Stephen Armitage, and Richard Stevens. Managing standards compliance. *Software Engineering, IEEE Transactions on*, 25(6):836–851, 1999.

[EN95] Steve Easterbrook and Bashar Nuseibeh. Managing inconsistencies in an evolving specification. In *Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on*, pages 48–55. IEEE, 1995.

[Erw10] Martin Erwig. A language for software variation research. In *ACM SIGPLAN Notices*, volume 46, pages 3–12. ACM, 2010.

[ESK04] Levent Ertöz, Michael Steinbach, and Vipin Kumar. *Finding topics in collections of documents: A shared nearest neighbor approach.* Springer, 2004.

[EW11] Martin Erwig and Eric Walkingshaw. The choice calculus: A representation for software variation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 21(1):6, 2011.

[FA99] Katerina T Frantzi and Sophia Ananiadou. The c-value/nc-value domain-independent method for multi-word term extraction. *Journal of natural language processing*, 6(3):145–179, 1999.

[FHM06] Elena Filatova, Vasileios Hatzivassiloglou, and Kathleen McKeown. Automatic creation of domain templates. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 207–214. Association for Computational Linguistics, 2006.

[Fir03] Donald Firesmith. Specifying good requirements. *Journal of Object Technology*, 2(4):77–87, 2003.

[FKMP13] Mārtiņš Francis, Dimitrios S Kolovos, Nicholas Matragkas, and Richard F Paige. Adding spreadsheets to the mde toolkit. In *Model-Driven Engineering Languages and Systems*, pages 35–51. Springer, 2013.

[FMP08] Thomas Forster, Dirk Muthig, and Daniel Pech. Understanding Decision Models â€“ Visualization and Complexity reduction of Software Variability. In *Proceedings of the 2nd Int. Workshop on Variability Modelling of Software-intensive Systems*, Essen, Germany, January 2008.

[Fox95] Christopher Fox. A domain analysis and reuse environment. 1995.

[Fre98] Dayne Freitag. Toward general-purpose learning for information extraction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 404–408. Association for Computational Linguistics, 1998.

[Fre99] Alex Alves Freitas. On rule interestingness measures. *Knowledge-Based Systems*, 12(5):309–315, 1999.

[FSd13] Alessio Ferrari, Giorgio Oronzo Spagnolo, and Felice dell'Orletta. Mining commonalities and variabilities from natural language documents. In *SPLC*, pages 116–120, 2013.

[GAP09] Sepideh Ghanavati, Daniel Amyot, and Liam Peyton. Compliance analysis based on a goal-oriented requirement language evaluation methodology. In *Requirements Engineering Conference, 2009. RE'09. 17th IEEE International*, pages 133–142. IEEE, 2009.

[GB97] Leah Goldin and Daniel M Berry. Abstfinder, a prototype natural language text abstraction finder for use in requirements elicitation. *Automated Software Engineering*, 4(4):375–412, 1997.

[GF12] David A Grossman and Ophir Frieder. *Information retrieval: Algorithms and heuristics*, volume 15. Springer Science & Business Media, 2012.

[GFA98] Martin L Griss, John Favaro, and Massimo D Alessandro. Integrating feature modeling with the rseb. In *Software Reuse, 1998. Proceedings. Fifth International Conference on*, pages 76–85. IEEE, 1998.

[GKG09] Nikhil Gupta, Praveen Kumar, and Rahul Gupta. Cs 224n final project: Automated extraction of product attributes from reviews. 2009.

[Gom06] Hassan Gomaa. Designing software product lines with uml 2.0: From use cases to pattern-based software architectures. In *Reuse of Off-the-Shelf Components*, pages 440–440. Springer, 2006.

[GPL+06] Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew Fano. Text mining for product attribute extraction. *ACM SIGKDD Explorations Newsletter*, 8(1):41–48, 2006.

[GW89] Donald C Gause and Gerald M Weinberg. *Exploring requirements: quality before design*. Dorset House New York, 1989.

[GW12] Bernhard Ganter and Rudolf Wille. *Formal concept analysis: mathematical foundations*. Springer Science & Business Media, 2012.

[HB08] Jakob Halskov and Caroline Barrière. Web-based extraction of semantic relation instances for terminology work. *Terminology*, 14(1):20–44, 2008.

[HCHM+13a] Negar Hariri, Carlos Castro-Herrera, Mehdi Mirakhorli, Jane Cleland-Huang, and Bamshad Mobasher. Supporting domain analysis through mining and recommending features from online product listings. *IEEE Transactions on Software Engineering*, 99(PrePrints):1, 2013.

[HCHM+13b] Negar Hariri, Carlos Castro-Herrera, Mehdi Mirakhorli, Jane Cleland-Huang, and Bamshad Mobasher. Supporting domain analysis through mining and recommending features from online product listings. *Software Engineering, IEEE Transactions on*, 39(12):1736–1752, 2013.

[HG94] David G Hendry and Thomas RG Green. Creating, comprehending and explaining spreadsheets: a cognitive interpretation of what discretionary users think of the spreadsheet model. *International Journal of Human-Computer Studies*, 40(6):1033–1065, 1994.

[HH99] Robert J Hilderman and Howard J Hamilton. Heuristic measures of interestingness. In *Principles of Data Mining and Knowledge Discovery*, pages 232–241. Springer, 1999.

[HH01] Robert J Hilderman and Howard J Hamilton. Evaluation of interestingness measures for ranking discovered knowledge. In *Advances in Knowledge Discovery and Data Mining*, pages 247–259. Springer, 2001.

[HHKH96] Pei Hsia, Chih-Tung Hsu, David C Kung, and Lawrence B Holder. User-centered system decomposition: Z-based requirements clustering. In *Requirements Engineering, 1996., Proceedings of the Second International Conference on*, pages 126–135. IEEE, 1996.

[HHL09] Mohammad Hamdaqa and Abdelwahab Hamou-Lhadj. Citation analysis: an approach for facilitating the understanding and the analysis of regulatory compliance documents. In *Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on*, pages 278–283. IEEE, 2009.

[HL04] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[HLHE11] Evelyn Nicole Haslinger, Roberto E Lopez-Herrejon, and Alexander Egyed. Reverse engineering feature models from programs' feature sets. In *Reverse Engineering (WCRE), 2011 18th Working Conference on*, pages 308–312. IEEE, 2011.

[HLHE13] Evelyn Nicole Haslinger, Roberto Erick Lopez-Herrejon, and Alexander Egyed. On extracting feature models from sets of valid feature combinations. In *Fundamental Approaches to Software Engineering*, pages 53–67. Springer, 2013.

[Hoo93] Ivy Hooks. Writing good requirements (a requirements working group information report). In *Proc. Third International Symposium of the NCOSE*, volume 2, page 47, 1993.

[HP03] Günter Halmans and Klaus Pohl. Communicating the variability of a software-product family to customers. *Software and Systems Modeling*, 2(1):15–36, 2003.

[HPD12] Felienne Hermans, Martin Pinzger, and Arie van Deursen. Detecting and visualizing inter-worksheet smells in spreadsheets. In *Proceedings of the 34th International Conference on Software Engineering*, pages 441–451. IEEE Press, 2012.

[HPvD10] Felienne Hermans, Martin Pinzger, and Arie van Deursen. Automatically extracting class diagrams from spreadsheets. In *ECOOP 2010–Object-Oriented Programming*, pages 52–75. Springer, 2010.

[HPVD11] Felienne Hermans, Martin Pinzger, and Arie Van Deursen. Supporting professional spreadsheet users by generating leveled dataflow diagrams. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 451–460. ACM, 2011.

[HPvD12] Frederik Hermans, Martin Pinzger, and Arie van Deursen. Detecting code smells in spreadsheet formulas. In *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, pages 409–418. IEEE, 2012.

[HPY00] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOD Record*, volume 29, pages 1–12. ACM, 2000.

[IEE90] IEEE. IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, pages 1–84, 1990.

[JG08] H Ji and R Grishman. Brefining event extraction through unsupervised cross-document inference,[in proc. In *Annu. Meeting Assoc. Comput. Linguist*, pages 254–262, 2008.

[JK95] John S Justeson and Slava M Katz. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural language engineering*, 1(01):9–27, 1995.

[JKW08] Mikoláš Janota, Victoria Kuzina, and Andrzej Wąsowski. Model construction with external constraints: An interactive journey from semantics to syntax. In *Model Driven Engineering Languages and Systems*, pages 431–445. Springer, 2008.

[JMF99] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[Joh06] Isabel John. Capturing product line information from legacy user documentation. In *Software Product Lines*, pages 127–159. Springer, 2006.

[Jon86] Cliff B Jones. *Systematic software development using VDM*, volume 2. Prentice-Hall Englewood Cliffs, NJ, 1986.

[JS10] David Jurgens and Keith Stevens. The s-space package: an open source package for word space models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 30–35. Association for Computational Linguistics, 2010.

[JZ05] Jianxin Jiao and Yiyang Zhang. Product portfolio identification based on association rule mining. *Computer-Aided Design*, 37(2):149–172, 2005.

[Kam05a] Erik Kamsties. Understanding ambiguity in requirements engineering. In *Engineering and Managing Software Requirements*, pages 245–266. Springer, 2005.

[Kam05b] Erik Kamsties. Understanding ambiguity in requirements engineering. In *Engineering and Managing Software Requirements*, pages 245–266. Springer, 2005.

[Kar95] Joachim Karlsson. *Towards a strategy for software requirements selection.* Department of Computer and Information science, Linköping University, 1995.

[KCH+90] Kyo C Kang, Sholom G Cohen, James A Hess, William E Novak, and A Spencer Peterson. Feature-oriented domain analysis (foda) feasibility study. Technical report, DTIC Document, 1990.

[KIM+04] Nozomi Kobayashi, Kentaro Inui, Yuji Matsumoto, Kenji Tateishi, and Toshikazu Fukushima. Collecting evaluative expressions for opinion extraction. In *Natural Language Processing–IJCNLP 2004*, pages 596–605. Springer, 2004.

[KKL+98] Kyo C Kang, Sajoong Kim, Jaejoon Lee, Kijoo Kim, Euiseob Shin, and Moonhang Huh. Form: A feature-; oriented reuse method with domain-; specific reference architectures. *Annals of Software Engineering*, 5(1):143–168, 1998.

[KKM08] Ioannis Korkontzelos, Ioannis P Klapaftis, and Suresh Manandhar. Reviewing and evaluating automatic term recognition techniques. In *Advances in Natural Language Processing*, pages 248–259. Springer, 2008.

[KLD02] K.C. Kang, Jaejoon Lee, and P. Donohoe. Feature-oriented product line engineering. *Software, IEEE*, 19(4):58–65, 2002.

[KO10] Niels Kasch and Tim Oates. Mining script-like structures from the web. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 34–42. Association for Computational Linguistics, 2010.

[Kow98] Gerald Kowalski. Information retrieval systems: theory and implementation. *Computers and Mathematics with Applications*, 5(35):133, 1998.

[KR97] Joachim Karlsson and Kevin Ryan. A cost-value approach for prioritizing requirements. *Software, IEEE*, 14(5):67–74, 1997.

[KU96] Kyo Kageura and Bin Umino. Methods of automatic term recognition: A review. *Terminology*, 3(2):259–289, 1996.

[KZB+08] Nadzeya Kiyavitskaya, Nicola Zeni, Travis D Breaux, Annie I Antón, James R Cordy, Luisa Mich, and John Mylopoulos. Automating the extraction of rights and obligations for regulatory compliance. In *Conceptual Modeling-ER 2008*, pages 154–168. Springer, 2008.

[Lam05] Giuseppe Lami. Quars: A tool for analyzing requirements. Technical report, DTIC Document, 2005.

[LCHD07] Paula Laurent, Jane Cleland-Huang, and Chuan Duan. Towards automated requirements triage. In *Requirements Engineering Conference, 2007. RE'07. 15th IEEE International*, pages 131–140. IEEE, 2007.

[LHC05] Bing Liu, Minqing Hu, and Junsheng Cheng. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the 14th international conference on World Wide Web*, pages 342–351. ACM, 2005.

[LHGB+12] Roberto Erick Lopez-Herrejon, José A Galindo, David Benavides, Sergio Segura, and Alexander Egyed. Reverse engineering feature models with evolutionary algorithms: An exploratory study. In *Search Based Software Engineering*, pages 168–182. Springer, 2012.

[LHLG+14] Roberto E. Lopez-Herrejon, Lukas Linsbauer, José A. Galindo, José A. Parejo, David Benavides, Sergio Segura, and Alexander Egyed. An assessment of search-based techniques for reverse engineering feature models. *Journal of Systems and Software*, 2014.

[LHLG+15] Roberto E Lopez-Herrejon, Lukas Linsbauer, José A Galindo, José A Parejo, David Benavides, Sergio Segura, and Alexander Egyed. An assessment of search-based techniques for reverse engineering feature models. *Journal of Systems and Software*, 103:353–369, 2015.

[LHM99] Bing Liu, Wynne Hsu, and Yiming Ma. Pruning and summarizing the discovered associations. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 125–134. ACM, 1999.

[Liu07] Bing Liu. *Web data mining: exploring hyperlinks, contents, and usage data*. Springer Science & Business Media, 2007.

[LJC+11] Cane Wing-ki Leung, Jing Jiang, Kian Ming A Chai, Hai Leong Chieu, and Loo-Nin Teow. Unsupervised information extraction with distributional prior knowledge. 2011.

[LK95] Pericles Loucopoulos and Vassilios Karakostas. *System requirements engineering*. McGraw-Hill, Inc., 1995.

[LLX+09] Ping Luo, Fen Lin, Yuhong Xiong, Yong Zhao, and Zhongzhi Shi. Towards combining web classification and web information extraction: a case study. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1235–1244. ACM, 2009.

[LMSM] Alberto Lora-Michiels, Camille Salinesi, and Raúl Mazo. The baxter return of experience on the use of association rules to construct its product line model. In *Journée SPL, Lignes de produits logiciels et usines logicielles*.

[LMSM10] Alberto Lora-Michiels, Camille Salinesi, and Raúl Mazo. A method based on association rules to construct product line model. In *4th International Workshop on Variability Modelling of Software-intensive Systems (VaMos)*, page 50, 2010.

[LZ05] Zhenmin Li and Yuanyuan Zhou. Pr-miner: automatically extracting implicit programming rules and detecting violations in large software code. In *ACM SIGSOFT Software Engineering Notes*, volume 30, pages 306–315. ACM, 2005.

[MA99] Diana Maynard and Sophia Ananiadou. Term extraction using a similarity-based approach. *Recent advances in computational terminology*, pages 261–278, 1999.

[MA+09] Jeremy C Maxwell, Annie Antón, et al. Developing production rule models to aid in acquiring requirements from legal texts. In *Requirements Engineering Conference, 2009. RE'09. 17th IEEE International*, pages 101–110. IEEE, 2009.

[MA10] Jeremy C Maxwell and Annie I Antón. The production rule framework: developing a canonical set of software requirements for compliance with law. In *proceedings of the 1st ACM International Health Informatics Symposium*, pages 629–636. ACM, 2010.

[Mae02] Alexander Maedche. *Ontology learning for the semantic web*. Springer Science & Business Media, 2002.

[MAS+11] Jeremy C Maxwell, Annie Antón, Peter Swire, et al. A legal cross-references taxonomy for identifying conflicting software requirements. In *Requirements Engineering Conference (RE), 2011 19th IEEE International*, pages 197–206. IEEE, 2011.

[MC07] Mstislav Maslennikov and Tat-Seng Chua. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the Association of Computational Linguistics (ACL)*, 2007.

[MDS02] Zvika Marx, Ido Dagan, and Eli Shamir. Crosscomponent clustering for template induction. In *Workshop on Text Learning (TextML-2002), Sydney, Australia*, 2002.

[Mea06] Nancy R Mead. Requirements prioritization introduction. *Software Eng. Inst. web pub., Carnegie Mellon Univ*, 2006.

[MGL+06] Michael J May, Carl Gunter, Insup Lee, et al. Privacy apis: Access control techniques to analyze and verify legal privacy policies. In *Computer Security Foundations Workshop, 2006. 19th IEEE*, pages 13–pp. IEEE, 2006.

[MKS06] Seung Ki Moon, Soundar RT Kumara, and Timothy W Simpson. Data mining and fuzzy clustering to support product family design. In *ASME 2006 International Design Engineering Technical Conferences*

and Computers and Information in Engineering Conference, pages 317–325. American Society of Mechanical Engineers, 2006.

[MOA+09] Aaron K Massey, Paul N Otto, Annie Anton, et al. Prioritizing legal requirements. In *Requirements Engineering and Law (RELAW), 2009 Second International Workshop on*, pages 27–32. IEEE, 2009.

[Moi00] Frank Moisiadis. Prioritising scenario evolution. In *Requirements Engineering, 2000. Proceedings. 4th International Conference on*, pages 85–94. IEEE, 2000.

[MP14] Andreas Metzger and Klaus Pohl. Software product line engineering and variability management: achievements and challenges. In *Proceedings of the on Future of Software Engineering*, pages 70–84. ACM, 2014.

[MPH+07] Andreas Metzger, Klaus Pohl, Patrick Heymans, Pierre-Yves Schobbens, and Germain Saval. Disambiguating the documentation of variability in software product lines: A separation of concerns, formalization and automated analysis. In *Requirements Engineering Conference, 2007. RE'07. 15th IEEE International*, pages 243–253. IEEE, 2007.

[MR10] Douglas C Montgomery and George C Runger. *Applied statistics and probability for engineers*. John Wiley & Sons, 2010.

[NBKC14] Sarah Nadi, Thorsten Berger, Christian Kästner, and Krzysztof Czarnecki. Mining configuration constraints: Static analyses and empirical results. In *Proceedings of the 36th International Conference on Software Engineering (ICSE)*, 6 2014.

[NE08a] Nan Niu and Steve Easterbrook. Extracting and modeling product line functional requirements. In *International Requirements Engineering, 2008. RE'08. 16th IEEE*, pages 155–164. IEEE, 2008.

[NE08b] Nan Niu and Steve Easterbrook. On-demand cluster analysis for product line functional requirements. In *Software Product Line Conference, 2008. SPLC'08. 12th International*, pages 87–96. IEEE, 2008.

[NE09] Nan Niu and Steve Easterbrook. Concept analysis for product line requirements. In *Proceedings of the 8th ACM international conference on Aspect-oriented software development*, pages 137–148. ACM, 2009.

[NM03] Hiroshi Nakagawa and Tatsunori Mori. Automatic term recognition based on statistics of compound nouns and their components. *Terminology*, 9(2):201–219, 2003.

[NSAB14] Sana Ben Nasr, Nicolas Sannier, Mathieu Acher, and Benoit Baudry. Moving toward product line engineering in a nuclear industry consortium. In *Proceedings of the 18th International Software Product Line Conference-Volume 1*, pages 294–303. ACM, 2014.

[NvdBWR09] JAR Noppen, PM van den Broek, Nathan Weston, and Awais Rashid. Modelling imperfect product line requirements with fuzzy feature diagrams. 2009.

[OA+07] Paul N Otto, Annie Antón, et al. Addressing legal requirements in requirements engineering. In *Requirements Engineering Conference, 2007. RE'07. 15th IEEE International*, pages 5–14. IEEE, 2007.

[Pan08] Raymond R Panko. Thinking is bad: Implications of human error research for spreadsheet research and practice. *arXiv preprint arXiv:0801.3114*, 2008.

[PBM03] Monica Palmirani, Raffaella Brighi, and Matteo Massini. Automated extraction of normative references in legal texts. In *Proceedings of the 9th international conference on Artificial intelligence and law*, pages 105–106. ACM, 2003.

[PBvdL05a] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques.* Springer-Verlag, 2005.

[PBvdL05b] Klaus Pohl, Günter Böckle, and Frank J van der Linden. *Software product line engineering: foundations, principles and techniques.* Springer Science & Business Media, 2005.

[PE07] Ana-Maria Popescu and Orena Etzioni. Extracting product features and opinions from reviews. In *Natural language processing and text mining*, pages 9–28. Springer, 2007.

[PGK+07] Katharina Probst, Rayid Ghani, Marko Krema, Andrew E Fano, and Yan Liu. Semi-supervised learning of attribute-value pairs from product descriptions. In *IJCAI*, volume 7, pages 2838–2843, 2007.

[Poh94] Klaus Pohl. The three dimensions of requirements engineering: a framework and its applications. *Information systems*, 19(3):243–258, 1994.

[Poh10] Klaus Pohl. *Requirements Engineering - Fundamentals, Principles, and Techniques.* Springer, 2010.

[PPZ05] Maria Teresa Pazienza, Marco Pennacchiotti, and Fabio Massimo Zanzotto. Terminology extraction: an analysis of linguistic and statistical approaches. In *Knowledge Mining*, pages 255–279. Springer, 2005.

[PR07] Siddharth Patwardhan and Ellen Riloff. Effective information extraction with semantic affinity patterns and relevant regions. In *EMNLP-CoNLL*, volume 7, pages 717–727, 2007.

[PR09] Siddharth Patwardhan and Ellen Riloff. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 151–160. Association for Computational Linguistics, 2009.

[Pre05] Roger S Pressman. *Software engineering: a practitioner's approach.* Palgrave Macmillan, 2005.

[PRWB04] Andizej Pieczyriski, Silva Robak, and Anna Walaszek-Babiszewska. Features with fuzzy probability. In *Engineering of Computer-Based Systems, 2004. Proceedings. 11th IEEE International Conference and Workshop on the*, pages 323–328. IEEE, 2004.

[PVG+01] Anselmo Peñas, Felisa Verdejo, Julio Gonzalo, et al. Corpus-based terminology extraction applied to information access. In *Proceedings of Corpus Linguistics*, volume 2001, 2001.

[PWSBC10] Rajwinder Kaur Panesar-Walawege, Mehrdad Sabetzadeh, Lionel Briand, and Thierry Coq. Characterizing the chain of evidence for software safety cases: A conceptual model based on the iec 61508 standard. In *Software Testing, Verification and Validation (ICST), 2010 Third International Conference on*, pages 335–344. IEEE, 2010.

[PY13] Jakub Piskorski and Roman Yangarber. Information extraction: Past, present and future. In *Multi-source, multilingual information extraction and summarization*, pages 23–49. Springer, 2013.

[RF94] William N Robinson and Stephen Fickas. Supporting multi-perspective requirements engineering. In *Requirements Engineering, 1994., Proceedings of the First International Conference on*, pages 206–215. IEEE, 1994.

[RHW06] WENRA Reactor Harmonization Working Group RHWG. Harmonisation of Reactor Safety in WENRA Countries. Technical report, WENRA, 2006.

[Rie03] Matthias Riebisch. Towards a more precise definition of feature models. *Modelling Variability for Object-Oriented Product Lines*, pages 64–76, 2003.

[RKJ+92] Lisa Rau, George Krupka, Paul Jacobs, Ira Sider, and Lois Childs. Ge nltoolset: Muc-4 test results and analysis. In *Proceedings of the 4th conference on Message understanding*, pages 94–99. Association for Computational Linguistics, 1992.

[RP92] Colette Rolland and Christophe Proix. A natural language approach for requirements engineering. In *Advanced information systems engineering*, pages 257–277. Springer, 1992.

[RP04] Silva Robak and Andrzej Pieczyński. Application of fuzzy weighted feature diagrams to model variability in software families. In *Artificial Intelligence and Soft Computing-ICAISC 2004*, pages 370–375. Springer, 2004.

[RPK11] Uwe Ryssel, Joern Ploennigs, and Klaus Kabitzsch. Extraction of feature models from formal contexts. In *FOSD'11*, pages 1–8, 2011.

[RR01] John F Roddick and Sally Rice. What's interesting about cricket?: on thresholds and anticipation in discovered rules. *ACM SIGKDD Explorations Newsletter*, 3(1):1–5, 2001.

[RS98] Ellen Riloff and Mark Schmelzenbach. An empirical approach to conceptual case frame acquisition. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 49–56, 1998.

[RS10] Pradeep Rai and Shubha Singh. A survey of clustering techniques. *International Journal of Computer Applications*, 7(12):1–5, 2010.

[RWP05] Ellen Riloff, Janyce Wiebe, and William Phillips. Exploiting subjectivity classification to improve information extraction. In *Proceedings of the National Conference On Artificial Intelligence*, volume 20, page 1106. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[SA92] J Michael Spivey and JR Abrial. *The Z notation*. Prentice Hall Hemel Hempstead, 1992.

[SAB13] Nicolas Sannier, Mathieu Acher, and Benoit Baudry. From comparison matrix to variability model: The wikipedia case study. In *ASE'13*, pages 580–585. IEEE, 2013.

[Sag90] Juan C Sager. *A practical course in terminology processing*. John Benjamins Publishing, 1990.

[Sah99] Sigal Sahar. Interestingness via what is not interesting. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 332–336. ACM, 1999.

[SB88] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[SB12a] Nicolas Sannier and Benoit Baudry. Defining and retrieving themes in nuclear regulations. In *RELAW*, pages 33–41, 2012.

[SB12b] Nicolas Sannier and Benoit Baudry. Toward multilevel textual requirements traceability using model-driven engineering and information retrieval. In *Model-Driven Requirements Engineering Workshop (MoDRE), 2012 IEEE*, pages 29–38. IEEE, 2012.

[SB14a] Nicolas Sannier and Benoit Baudry. Increment: A mixed mde-ir approach for regulatory requirements modeling and analysis. In *REFSQ'2014*, pages 135–151, 2014.

[SB14b] Nicolas Sannier and Benoit Baudry. Increment: A mixed mde-ir approach for regulatory requirements modeling and analysis. In *Requirements Engineering: Foundation for Software Quality*, pages 135–151. Springer, 2014.

[SGN11] Pete Sawyer, Vincenzo Gervasi, and Bashar Nuseibeh. Unknown knowns: Tacit knowledge in requirements engineering. In *Requirements Engineering Conference (RE), 2011 19th IEEE International*, pages 329–329. IEEE, 2011.

[SHTB07] Pierre-Yves Schobbens, Patrick Heymans, Jean-Christophe Trigaux, and Yves Bontemps. Generic semantics of feature diagrams. *Computer Networks*, 51(2):456–479, 2007.

[SJ04] Klaus Schmid and Isabel John. A customizable approach to full lifecycle variability management. *Sci. Comput. Program.*, 53:259–284, December 2004.

[SLB+11a] S. She, R. Lotufo, T. Berger, A. Wasowski, and K. Czarnecki. Reverse engineering feature models. In *ICSE'11*, pages 461–470. ACM, 2011.

[SLB+11b] Steven She, Rafael Lotufo, Thorsten Berger, Andrzej Wøsowski, and Krzysztof Czarnecki. Reverse engineering feature models. In *Software Engineering (ICSE), 2011 33rd International Conference on*, pages 461–470. IEEE, 2011.

[SMPS09] Alberto Siena, John Mylopoulos, Anna Perini, and Angelo Susi. Designing law-compliant software requirements. In *Conceptual Modeling-ER 2009*, pages 472–486. Springer, 2009.

[SRC05] Pete Sawyer, Paul Rayson, and Ken Cosh. Shallow knowledge as an aid to deep understanding in early phase requirements engineering. *Software Engineering, IEEE Transactions on*, 31(11):969–981, 2005.

[SRG11] Klaus Schmid, Rick Rabiser, and Paul Grünbacher. A comparison of decision modeling approaches in product lines. In *Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems*, pages 119–126. ACM, 2011.

[SRS+93] Ian Sommerville, Tom Rodden, Pete Sawyer, Richard Bentley, and Michael Twidale. Integrating ethnography into the requirements engineering process. In *Requirements Engineering, 1993., Proceedings of IEEE International Symposium on*, pages 165–173. IEEE, 1993.

[SS06a] Yusuke Shinyama and Satoshi Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the main con-*

ference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, pages 304–311. Association for Computational Linguistics, 2006.

[SS06b] Andrew Stone and Pete Sawyer. Identifying tacit knowledge-based requirements. *IEE Proceedings-Software*, 153(6):211–218, 2006.

[SSG03] Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. An improved extraction pattern representation model for automatic ie pattern acquisition. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 224–231. Association for Computational Linguistics, 2003.

[ST95] Abraham Silberschatz and Alexander Tuzhilin. On subjective measures of interestingness in knowledge discovery. In *KDD*, volume 95, pages 275–281, 1995.

[STA06] Mihai Surdeanu, Jordi Turmo, and Alicia Ageno. A hybrid approach for the acquisition of information extraction patterns. In *EACL-2006 Workshop on Adaptive Text Extraction and Mining (ATEM 2006)*, pages 48–55, 2006.

[SVGB05] Mikael Svahnberg, Jilles Van Gurp, and Jan Bosch. A taxonomy of variability realization techniques. *Software: Practice and Experience*, 35(8):705–754, 2005.

[T+99] Ah-Hwee Tan et al. Text mining: The state of the art and the challenges. In *Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases*, volume 8, page 65, 1999.

[TB07] Anil Kumar Thurimella and Bernd Bruegge. Evolution in product line requirements engineering: A rationale management approach. In *Requirements Engineering Conference, 2007. RE'07. 15th IEEE International*, pages 254–257. IEEE, 2007.

[TBK09] Thomas Thüm, Don Batory, and Christian Kästner. Reasoning about edits to feature models. In *ICSE'09*, pages 254–264. ACM, 2009.

[TGTG05] Patrick Tessier, Sébastien Gérard, François Terrier, and Jean-Marc Geib. Using variation propagation for model-driven management of a system family. In *Software Product Lines*, pages 222–233. Springer, 2005.

[TSK06] Pang-Ning Tan, Micahel Steinbach, and Vipin Kumar. Introduction todata mining. *University of Minnesota: Addisson-Wesley*, 2006.

[TTC09] Huifeng Tang, Songbo Tan, and Xueqi Cheng. A survey on sentiment detection of reviews. *Expert Systems with Applications*, 36(7):10760–10773, 2009.

[Tur02] Peter D Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics, 2002.

[UMN+04] Naohiko Uramoto, Hirofumi Matsuzawa, Tohru Nagano, Akiko Murakami, Hironori Takeuchi, and Koichi Takeda. A text-mining system for knowledge discovery from biomedical documents. *IBM Systems Journal*, 43(3):516–533, 2004.

[vEB03] Tom M van Engers and Margherita R Boekenoogen. Improving legal quality: an application report. In *Proceedings of the 9th international conference on Artificial intelligence and law*, pages 284–292. ACM, 2003.

[VLDL98] Axel Van Lamsweerde, Robert Darimont, and Emmanuel Letier. Managing conflicts in goal-driven requirements engineering. *Software Engineering, IEEE Transactions on*, 24(11):908–926, 1998.

[W+99] David M Weiss et al. Software product-line engineering: a family-based software development process. 1999.

[WCR09] Nathan Weston, Ruzanna Chitchyan, and Awais Rashid. A framework for constructing semantically composable feature models from natural language requirements. In *Proceedings of the 13th International Software Product Line Conference*, pages 211–220. Carnegie Mellon University, 2009.

[WF74] Robert A Wagner and Michael J Fischer. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173, 1974.

[Wie99] Karl E Wiegers. Writing quality requirements. *Software Development*, 7(5):44–48, 1999.

[Yau92] Alan T Yaung. Design and implementation of a requirements clustering analyzer for software system decomposition. In *Proceedings of the*

1992 ACM/SIGAPP symposium on Applied computing: technological challenges of the 1990's, pages 1048–1054. ACM, 1992.

[YC79] Edward Yourdon and Larry L Constantine. *Structured design: Fundamentals of a discipline of computer program and systems design*. Prentice-Hall, Inc., 1979.

[YCN+10] Chunyu Yang, Yong Cao, Zaiqing Nie, Jie Zhou, and Ji-Rong Wen. Closing the loop in webpage understanding. *Knowledge and Data Engineering, IEEE Transactions on*, 22(5):639–650, 2010.

[YGTH00] Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 940–946. Association for Computational Linguistics, 2000.

[You11] Jessica D Young. Commitment analysis to operationalize software requirements from privacy policies. *Requirements Engineering*, 16(1):33–46, 2011.

[YZZ+12] Li Yi, Wei Zhang, Haiyan Zhao, Zhi Jin, and Hong Mei. Mining binary constraints in the construction of feature models. In *RE'12*, pages 141–150. IEEE, 2012.

[Zah71] Charles T Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *Computers, IEEE Transactions on*, 100(1):68–86, 1971.

[ZEMK97] Oren Zamir, Oren Etzioni, Omid Madani, and Richard M Karp. Fast and intuitive clustering of web documents. In *KDD*, volume 97, pages 287–290, 1997.

[ZJ97] Pamela Zave and Michael Jackson. Four dark corners of requirements engineering. *ACM transactions on Software Engineering and Methodology (TOSEM)*, 6(1):1–30, 1997.

[ZJ06] Tewfik Ziadi and Jean-Marc Jézéquel. Software product line engineering with the uml: Deriving products. In *Software Product Lines*, pages 557–588. Springer, 2006.

[ZK02] Ying Zhao and George Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 515–524. ACM, 2002.

[ZK07] Paul Zhang and Lavanya Koppaka. Semantics-based legal citation network. In *Proceedings of the 11th international conference on Artificial intelligence and law*, pages 123–130. ACM, 2007.

[ZLLOS10] Lei Zhang, Bing Liu, Suk Hwan Lim, and Eamonn O'Brien-Strain. Extracting and ranking product features in opinion documents. In *Proceedings of the 23rd international conference on computational linguistics: Posters*, pages 1462–1470. Association for Computational Linguistics, 2010.

[ZNZW08] Jun Zhu, Zaiqing Nie, Bo Zhang, and Ji-Rong Wen. Dynamic hierarchical markov random fields for integrated web data extraction. *The Journal of Machine Learning Research*, 9:1583–1614, 2008.

[ZSWG09] Shuyi Zheng, Ruihua Song, Ji-Rong Wen, and C Lee Giles. Efficient record-level wrapper induction. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 47–56. ACM, 2009.

List of Figures

1.1 The product line engineering framework [PBvdL05b] 10
1.2 A family of mobile phones described with a feature model [BSRC10] . . 14
1.3 PCM of Wikipedia about Portable Media Players 17
1.4 The PCM Metamodel [BSA+14] 18

2.1 Text Mining Framework [T+99] 26

3.1 Overview of the nuclear regulatory landscape [SB12a] 46
3.2 A Metamodel for Structuring Requirements Collections [SB12b] 48
3.3 Overview of the General Approach 54
3.4 Mapping between standards BM and standards FM 56
3.5 Feature Model Synthesis 61
3.6 Mapping between the standards FM and the design rules FM 67
3.7 Mapping between Design Rules FM and I&C Architecture 69
3.8 I&C Architecture PL (Sequoia) 70
3.9 Evaluation Overview 72
3.10 Properties of Requirements FM and Design rules FM 73
3.11 Quality of Clusters 75

4.1 Automatic Extraction of PCM from Textual Product Descriptions . . . 80
4.2 Computing Technical Specifications PCM 81
4.3 Complementarity Aspect of Product Overviews and Technical Specifications 84
4.4 Approach Overview 84
4.5 The editor of *MatrixMiner* in action 94
4.6 Features: Random vs Supervised Scoping 100
4.7 Cell Values: Random vs Supervised Scoping 101
4.8 Complementarity of Overview PCMs and Specification PCMs (RQ1.3) . 105

4.9 Overview of the environment during PCMs evaluation (by column) . . . 107
4.10 Quality of Features and Cell Values 109
4.11 Overlap between PCMs and the Specifications 111

5.1 Lessons Learned 117

6.1 Context-independent methodology for mining and modeling variability from informal documentation 126

List of Tables

1.1 Valid product configurations of mobile phone SPL 16

3.1 Mining variability in *Independence* topic 57
3.2 Mining variability in *Safety Classification* topic 57
3.3 Identification of features from IEC and IEEE standards and design rules 58
3.4 Case Study Data Description 71

4.1 Overview dataset 96
4.2 Properties of Synthesized PCMs: Random vs Supervised Scoping 102

5.1 Case Studies Comparison 118

Abstract

A Product Line (PL) is a collection of closely related products that together address a particular market segment or fulfil a particular mission. In product line engineering, domain analysis is the process of analyzing these products to identify their common and variable features. This process is generally carried out by experts on the basis of existing informal documentation. When performed manually, this activity is both time-consuming and error-prone. Numerous approaches have been proposed to mine variability and support domain analysis, but few of them adopt automated techniques for the construction of variability models from unstructured and ambiguous documents. In this thesis, our general contribution is to address mining and modeling variability from informal documentation. We adopt Natural Language Processing (NLP) and data mining techniques to identify features, commonalities, differences and features dependencies among the products in the PL. We investigate the applicability of this idea by instantiating it in two different contexts: (1) reverse engineering *Feature Models (FMs)* from regulatory requirements in nuclear domain and (2) synthesizing *Product Comparison Matrices (PCMs)* from informal product descriptions.

The first case study aims at capturing variability from textual regulations in nuclear domain. We propose an approach to extract variability from safety requirements as well as mapping variable requirements and variable architecture elements to derive a complying architecture. We adopt NLP and data mining techniques based on semantic analysis, requirements clustering and association rules to assist experts when constructing feature models from these regulations. The evaluation shows that our approach is able to retrieve 69% of correct clusters without any user intervention. We notice that structural dependencies show a high predictive capacity: 95% of the mandatory relationships and 60% of optional relationships are found. We also observe that the totality of requires and exclude relationships are extracted.

The second case study is about the extraction of variability from informal product descriptions. Our proposed approach relies on contrastive analysis technology to mine *domain specific terms* from text, information extraction, terms clustering and information clustering. Overall, our empirical study shows that the resulting PCMs exhibit numerous quantitative and comparable information: 12.5% of quantified features, 15.6% of descriptive features and only 13% of empty cells. The user study shows that our automatic approach retrieves 43% of correct features and 68% of correct values in one step and without any user intervention. We also show that regarding a significant portion of features (56%) and values (71%), we have as much or more information in the generated PCMs than in the specifications.

The main lesson learnt from the two case studies is that three key factors affect the choice of techniques to apply when mining and exploiting variability knowledge from informal documentation. These factors are: the context, the nature of variability and the nature of text. Specifically, formalizing variability depends on the nature of the input text and the context while the choice of NLP and data mining techniques, employed when mining variability, is influenced by the choice of the formalism and the kind of text.