RECODYN : Une approche d'ingénierie d'application basée sur la recommandation et la configuration interactive des lignes de produits

Raouia Ben Charrada Triki

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

THESE DE DOCTORAT

DE L'UNIVERSITE PARIS 1 PANTHEON-SORBONNE

Spécialité : Informatique

Raouia Ben Charrada Triki

Pour l'obtention du grade de :

Docteur de l'Université Paris 1 Panthéon-Sorbonne

RECODYN :

Une approche d'ingénierie d'application basée sur la recommandation et la configuration interactive des lignes de produits

Soutenue le 16 Décembre 2016 devant le jury composé de :

Pr. Camille Salinesi Professeur à l'Université Paris 1 Panthéon-Sorbonne, France.	Directeur de Thèse
Pr. Naoufel Kraiem Professeur à l'Université Sultan Qaboos, Oman.	Rapporteur
Dr. Samira Si-Saïd Cherfi Maître de conférences (HDR) au Conservatoire National des Arts et Métiers, France.	Rapporteur
Dr. Saïd Assar Maître de conférences à l'Institut Mines Telecom-Telecom Ecole de Management, France.	Membre du jury
Dr. Raul Mazo Maître de conférences à l'Université Paris 1 Panthéon-Sorbonne, France.	Membre du jury

Une approche d'ingénierie d'application basée sur la recommandation et la configuration interactive des lignes de produits

Résumé

En ingénierie des lignes de produits, la configuration des produits décrit le processus de développement d'un produit selon les besoins des utilisateurs, par réutilisation à partir d'un modèle de ligne de produits. Le problème est qu'il y a un grand nombre de produits dans une LP, au point qu'il est impossible de les spécifier tous explicitement. Ensuite, lorsqu'un utilisateur prend une décision cela peut être contradictoire avec les décisions antérieures. Par conséquent, il est crucial de guider l'utilisateur en combinant la recommandation et la configuration.

Cette thèse présente une revue de la littérature sur les approches de configuration des lignes de produits et les types et techniques de recommandation. Elle montre l'importance de l'utilisation de la recommandation dans le processus de configuration.

Une approche de recommandation, Recodyn, est proposée. Elle permet la configuration interactive en utilisant la recommandation pour guider l'utilisateur à prendre des décisions. Recodyn informe l'utilisateur en temps réel l'utilisateur sur les caractéristiques souhaitables, possibles et inaccessibles en fonction des choix qu'il a déjà effectués, et suggère des décisions portant sur ces caractéristiques et des choix à faire en raisonnant à partir de configurations connues. Recodyn combine la recommandation basée sur la connaissance et celle basée sur le contenu. L'idée clé est de focaliser la recommandation sur un sous-ensemble de caractéristiques pour générer des recommandations satisfaisant les exigences de l'utilisateur et respectant les contraintes de la ligne de produits.

L'évaluation de Recodyn a été effectuée par le développement du logiciel DynaElec implémentant Recodyn. Un cas industriel réel (tableau électrique) a été utilisé pour réaliser cette évaluation. La faisabilité et les performances de Recodyn ont été évalués.

Mots-clés : ligne de produits, configuration, recommandation, caractéristique, similarité, contrainte, exigence.

Centre de Recherche en Informatique

UFR 27
Centre de Recherche en Informatique
Centre Pierre Mendès France
14ème étage
90 rue de Tolbiac
75013

An application engineering approach based on the recommendation and interactive configuration of product lines

Abstract

In Product Line Engineering, product configuration describes the process of developing a product according to user requirements, by reuse from a Product Line model. The problem is that there are so many products in a PL that it is impossible to specify all of them explicitly. Then, when a user makes a decision, this can be contradictory with former decisions. Consequently, it's crucial to guide the user by combining recommendation and configuration.

This thesis presents a literature review about product line configuration approaches, recommendation types and recommendation techniques. It shows the importance of the use of recommendation in the configuration process.

A recommendation approach, Recodyn, is proposed. It allows an interactive configuration by using recommendation to guide the user to take decisions. Recodyn informs the user in real time about desirable/possible/unattainable features consistently with his former choices, and suggests decision that focus on, and choices to make by reasoning on known configurations. Recodyn combines the knowledge-based with the content-based recommendation. The key idea is to focus on a subset of features to recommend configurations satisfying the user requirements with respect to the product line constraints.

The evaluation of Recodyn was fulfilled by developing the DynaElec software that implements Recodyn. An industrial world real case (electric board) was used to carry out the evaluation. Then a proof of concept was achieved, and an evaluation of the Recodyn performance was carried out.

Keywords: product line, configuration, recommendation, feature, similarity, constraint, requirement.

Remerciements

Durant ces années de travail, j'ai pu bénéficier du soutien et des conseils d'un grand nombre de personnes qui m'ont permis d'arriver au bout de cette thèse et que je tiens à remercier.

Je souhaiterais tout d'abord exprimer mes plus vifs remerciements à mon directeur de thèse, Monsieur Camille Salinesi, Professeur à l'Université Paris 1 Panthéon-Sorbonne pour la confiance qu'il m'a accordé en acceptant la direction scientifique de mes recherches. Je lui suis reconnaissante de m'avoir fait bénéficier de sa compétence et de ses pertinents conseils. Je le remercie également pour ses qualités humaines d'écoute et compréhension tout au long de ce travail de recherche.

Je voudrais adresser mes remerciements les plus sincères à Monsieur Naoufel Kraiem, Professeur à l'Université Sultan Qaboos Muscat, et Madame Samira Si-Saïd Cherfi, Maître de conférences, HDR au Conservatoire National des Arts et Métiers (CNAM) Paris, de m'avoir fait l'honneur d'accepter de rapporter cette thèse, et pour leurs précieux commentaires.

Je remercie également Monsieur Saïd Assar, Maître de conférences à Institut Mines Telecom- Telecom Ecole de Management, et Monsieur Raúl Mazo, Maître de conférences à l'Université Paris1 Panthéon Sorbonne d'avoir accepté de faire partie du jury de cette thèse. Je remercie particulièrement Raúl pour ses recommandations et suggestions qui m'ont aidée à améliorer mes recherches durant ces années.

J'adresse également mes remerciements à tous les membres du Centre de Recherche en Informatique (CRI) et en particulier à l'ensemble des thésards pour leur amitié et leurs encouragements.

J'exprime ma gratitude à mes chers parents Faouzi et Noura pour leur soutien lors des moments difficiles durant ces années et pour l'affection qu'ils m'ont manifesté. Merci pour avoir fait de moi celle que je suis aujourd'hui. Merci à mes Sœurs Maroua et Dorra pour leur amour et leurs encouragements.

J'adresse un remerciement particulier à mon cher mari Bassem. Je le remercie profondément pour sa patience durant ces années, de m'avoir soutenue dans les moments les plus difficiles, d'avoir cru en moi et de m'avoir aimée.

Enfin, un remerciement spécial à mon adorable fille Nour pour son calme et sa patience au delà de ses quatre ans.

Raouia

1. Sommaire

2. Liste des Tableaux

3. Liste des Figures

CHAPITRE 1: INTRODUCTION

1.1 Contexte

Depuis longtemps, les produits fabriqués ont été conçus uniquement pour des clients individuels. Plus tard avec le nombre croissant des clients et l'arrivée de l'industrialisation, les processus de fabrication ont été révolutionnés par les lignes de production (FORD[1]). Celles-ci ont permis une production de masse à coût réduit. Cette stratégie génère des produits identiques, ce qui est un avantage de point de vue logistique. Toutefois, le problème est que les produits identiques peuvent ne pas satisfaire tous les clients, en particulier ceux dont les besoins sont différents. Une alternative est alors de fournir des produits personnalisés répondant aux exigences spécifiques de chaque client tout en conservant une partie commune dont la production resterait efficace. Le paradigme de l'ingénierie des Lignes de Produits (LP) répond à cette exigence. Il s'agit de produits personnalisés tout en gardant des coûts faibles de production en combinant (i) la réutilisation d'éléments ou objets appelés artéfacts à partir d'une plateforme commune qui spécifie tous les éléments qui peuvent être utilisés pour développer des produits différents, et (ii) la personnalisation de masse (Pohl et al., 2005) qui consiste à fabriquer des produits spécifiques en grandes quantités. Ainsi, l'ingénierie des LPs est utilisée pour gérer les élements communes et variables dans une LP. Le Software Engineering Institute (SEI) définit une LP comme « *un ensemble de systèmes qui partagent un ensemble de caractéristiques communs, répondant aux besoins spécifiques d'un segment de marché particulier ou d'une mission particulière et qui sont développés à partir d'un ensemble commun d'artéfacts de base de manière prescrite* » (SEI[2]). L'ingénierie des LPs traite la réutilisation d'artéfacts en séparant deux processus : l'ingénierie de domaine et l'ingénierie d'application (Pohl et al., 2005).

L'ingénierie de domaine consiste à « *collecter, organiser et stocker l'expérience passée dans des systèmes ou des parties de systèmes de construction dans un domaine particulier sous la forme d'artéfacts réutilisables, et de fournir un moyen adéquat pour réutiliser ces artéfacts lors de la création d'un nouveau système* » (Czarnecki, 1998). Ainsi, l'objectif de l'ingénierie de domaine est de spécifier les artéfacts réutilisables et leur contexte de réutilisation en définissant les éléments communs et variables de la LP. Elle est, en outre, composée de plusieurs sous-processus: la gestion des produits, l'ingénierie des exigences

[1] http://www.ford.co.uk

[2] http://www.sei.cmu.edu

du domaine, la conception du domaine, la réalisation du domaine et le test du domaine. Les artéfacts réutilisables sont, par exemple, les exigences, la conception, la réalisation, les tests, etc.

L'ingénierie d'application consiste pour sa part en « *le processus d'ingénierie des LPs dans lequel les applications de LP sont construites en réutilisant des artéfacts de domaine et en exploitant la variabilité de la LP* » (Pohl et al., 2005). Au cours de l'ingénierie d'application, les systèmes sont conçus par réutilisation. En effet, ce processus permet de générer des applications spécifiques en exploitant la variabilité de la LP et en réutilisant le domaine d'exigences. Le concept central pour traiter la réutilisation dans l'ingénierie des LPs est de définir des artéfacts communs et variables dans les modèles des LPs.

L'activité de réutilisation des artéfacts du domaine pour concevoir des applications est appelée « *processus de configuration* » (PC). La configuration consiste principalement à générer de nouveaux produits en faisant correspondre les exigences des utilisateurs au domaine des artéfacts. Elle se traduit par la sélection et la combinaison des artéfacts : éléments, exigences, code, etc.

Les deux types d'artéfacts, communs et variables, sont définis comme suit :

- Les artéfacts *communs* sont des parties appartenant à tous les produits de la LP.
- Les artéfacts *variables* font partie de certains produits de la LP.

Ces éléments communs et variables sont généralement représentées dans les LPs à travers des modèles appelés « Modèle de LP » (MLP).

La variabilité est définie comme « *la capacité d'un système ou d'un artéfact à être configuré, personnalisé, étendu ou modifié pour une utilisation dans un contexte spécifique* » (Sinnema & Deelstra, 2006). Elle peut être représentée sous différentes formes telles que :

- La présence ou non d'une caractéristique dans un MLP
- Le nombre d'occurrences d'une caractéristique
- Les groupes de caractéristiques
- Les valeurs d'une caractéristique, etc.

Plusieurs témoignages (Deelstra, 2005) (SEI) montrent que la stratégie d'ingénierie de la LP offre divers avantages, tels que :

- L'amélioration de la productivité et la qualité du produit
- La diminution des coûts des délais de production et de commercialisation

Ces avantages sont particulièrement observés durant le processus de configuration. En effet, au cours de ce processus, l'utilisateur fait son choix parmi une liste d'options, permettant de vérifier ensuite que ses choix satisfont les contraintes et les dépendances de la LP. Ensuite, les impacts sont propagés sur d'autres choix possibles qui n'ont pas encore été réalisés. Toutefois l'utilisateur souhaite exprimer ses besoins et spécifier des exigences plus complexes que le simple choix parmi le panel d'options. En outre, les quantités importantes de produits qu'une LP peut générer porte à confusion. En effet, plus le nombre de caractéristiques augmente, plus le nombre de produits et de leur variabilité augmente. Cependant, plus le nombre de caractéristiques augmente, plus il est difficile de faire des choix et prendre des décisions.

Il est à noter que dans ce contexte, l'utilisateur est toute personne désirant configurer un produit d'une ligne de produit. On peut citer à titre d'exemple, un client désirant acheter un produit sur un site commerçant ou des experts/ingénieurs du domaine auquel appartient la LP développée.

L'idée clé dans cette thèse est d'introduire et intégrer les techniques de recommandation (TR), connues dans la littérature (Adomavicius & Tuzhilin, 2005) (Felfernig & Burke, 2008), dans le processus de configuration des LPs pour faciliter la prise de décisions. Ces techniques permettent aux entreprises de fournir des services de e-commerce et e-marketing offrant l'accès à de grandes quantités de produits. Par exemple, Amazon utilise le système de recommandation musical Amie Street dont le principe est de permettre aux artistes de vendre leur musique. Ces derniers sont notés et commentés par les membres du site pour être recommandés et achetés par les utilisateurs. Toutefois, les TRs classiques ne sont pas suffisantes pour générer des produits complexes personnalisés comme les LPs. En effet, il serait intéressant de penser à combiner la recommandation avec la configuration pour obtenir de façon efficace des poduits personnalisés de masse à faibles coûts. Une nouvelle stratégie est proposée dans cette thèse.

1.2 Problématique

La configuration automatique des produits à partir des MLPs est un problème de contraintes connu dans la littérature. Elle est généralement traitée à l'aide de solveurs sur l'étagère qui génèrent des produits valides en respectant les contraintes développées dans la

LP (Salinesi et al., 2011). Un problème particulier, qui constitue en soi un défi important pour l'industrie, est celui de la configuration interactive de nouveaux produits à partir de MLP. Ce défi se traduit par les enjeux des configurateurs industriels comme le passage à l'échelle (Astesana et al. 2010) et le guidage.

Afin de mieux expliquer l'impact du passage à l'échelle, on peut prendre l'exemple d'un utilisateur qui prend une décision de configuration et qui souhaite en connaitre les conséquences (ex. trouver les produits les moins coûteux et les plus performants). D'un point de vue marketing, il n'est pas acceptable de faire attendre l'utilisateur pendant plusieurs secondes pour savoir si ses exigences sont correctes ou non en termes de configuration. Le fait que la recherche de configuration complète valide est un problème NP-difficile (Astesana et al. 2010), le calcul d'une telle configuration croît quadratiquement.

D'autre part, choisir parmi un large éventail d'options est dans la pratique difficile pour l'utilisateur qui ne sait ni par où commencer, ni quelle solution choisir. En outre, chaque décision peut être en contradiction avec les décisions précédentes, ou avoir un impact négatif sur les décisions en aval. L'utilisateur pourra être déçu de voir que ses choix ne sont pas corrects, avec le risque de se retourner finalement vers un concurrent. Par conséquent, il est crucial de guider l'utilisateur durant un processus garantissant l'existence d'au moins un produit satisfaisant à la fin du processus de configuration. Le deuxième enjeu est donc le guidage. Pour information, un utilisateur qui ne dispose d'aucun guidage a moins d'une chance sur 100 de faire une configuration correcte dans la famille Renault van « Traffic » (Dauron & Astesana, 2010).

Le problème avec les systèmes industriels est que les TRs ne peuvent être simplement appliquées à des systèmes complexes tels que les LPs, les logiciels configurables, ou les systèmes composites comportant un grand nombre d'options. En effet, le problème est qu'il y a généralement un grand nombre de produits dans une LP, au point qu'il est en général impossible de les spécifier tous explicitement. En outre, bien que les TRs aident à prendre des décisions sur de nombreuses options, elles ne tiennent pas compte des contraintes qui les lient. On est donc confronté à une situation où les systèmes de configuration et systèmes de recommandation résolvent deux parties du problème, mais de manière non intégrée. Afin de mettre en valeur les avantages de personnalisation offerts par la stratégie de la LP, qui sont acquis par des raisonnements sur la variabilité plutôt que sur les variantes, il est préconisé d'éviter de focaliser sur un sous-ensemble de produits. L'objectif est donc de

concevoir une stratégie efficace pour des produits complexes personnalisables en combinant recommandation et configuration.

1.3 Questions de recherche

La contribution principale de cette thèse réside en une technique originale combinant deux formes complémentaires de guidage : la recommandation et la configuration. Cette approche informe en temps réel le client (ou l'utilisateur de façon générale) sur les caractéristiques souhaitables, possibles et inaccessibles en fonction des choix qu'il a déjà effectués, et suggère des décisions portant sur ces caractéristiques et des choix à faire en raisonnant à partir de configurations connues (déjà réalisées et/ou souhaitables). L'hypothèse de travail est que les produits personnalisables peuvent être spécifiés à l'aide d'une approche de LP.

Ainsi, la principale question de recherche est :

QR_Principale : Comment combiner recommandation et la configuration des LPs ?

Afin d'apporter les éléments de réponse nécessaire à cette question, il est nécessaire de traiter les trois questions de recherche suivantes.

QR_1 : Quel type de recommandation devrait être utilisé durant le processus de configuration de LP pour fournir un produit répondant aux spécificités du client ?

Afin d'apporter une réponse convaincante à cette question, cette thèse présente une vue d'ensemble des types de recommandation les plus présents dans la littérature (Felfernig & Burke, 2008) et fournit une analyse critique de chacune d'entre elles. Sont en particulier discutées : la recommandation basée sur le contenu (Pazzani & Billsus, 1997), la recommandation collaborative (Linden et al., 2003) et la recommandation basée sur la connaissance (Felfernig & Burke, 2008). Un aspect important de cette analyse est de chercher les moyens de combiner la configuration avec ces types de recommandation (Burke & Ramezani, 2011), (Felfernig et al., 2014).

QR_2 : A quels niveaux du processus de configuration de la LP, la recommandation doit-elle être appliquée ?

Il s'agit de s'interroger sur les principaux niveaux que l'approche de recommandation proposée dans cette thèse devra adresser. Ces différents niveaux de recommandation sont les suivants :

Niveau 1: Les caractéristiques de la LP (Falkner et al., 2011). Il s'agit des caractéristiques et des aspects du produit qui sont visibles pour le client (Kang et al., 1990). Par exemple, la couleur, la taille, la présence d'une fonction, ou la capacité à gérer des données.

Niveau 2: Les valeurs des caractéristiques (Falkner et al., 2011). Il s'agit de recommander les instances possibles des caractéristques. La recommandation des valeurs des caractéristiques permet aux utilisateurs de mieux comprendre les dépendances entre les exigences et les configurations possibles des autres caractéristiques (Cöster et al., 2002). En effet, une telle recommandation permet d'aider l'utilisateur à sélectionner les valeurs de caractéristiques similaires à des exigences spécifiées précédemment.

Niveau 3: La quantité des caractéristiques. C'est le nombre d'apparitions d'une certaine caractéristique dans une configuration (Czarnecki et al., 2005).

Niveau 4: L'ordre de sélection des caractéristiques par l'utilisateur. C'est le classement dans lequel les caractéristiques sont organisées pour configurer un produit.

QR_3 : Quel ordre de sélection des caractéristiques améliore le processus de recommandation ?

Il est important que la configuration du MLP soit autant axée sur le client que sur le produit lui-même. Les clients devraient avoir la possibilité (i) d'intérroger le MLP avec des critères qui ne sont pas nécessairement considérés dans le modèle; (ii) de choisir l'ordre de configuration; et (iii) de faire des choix librement. Toutefois, l'excès de liberté pourrait rendre le processus de configuration lent et fastidieux, et la configuration souhaitée pourrait finalement être perdue (Triki et al., 2014). Le classement des caractéristiques selon leur importance à l'utilisateur est une technique qui a été proposée dans certains travaux afin d'améliorer le processus de recommandation (Falkner et al., 2011) (Pu & Chen, 2008).

1.4 Méthodologie de recherche

Le projet de recherche rapporté dans cette thèse a été mené en utilisant la méthodologie Design Science. March & Smith (March & Smith, 1995) et Hevner et al. (Hevner et al., 2004) ont défini la design science (science de la conception) comme

étant *« la conception et la validation des propositions de solution à des problèmes pratiques »*. Hevner et al. (Hevner et al., 2004) fournissent des lignes directrices afin d'aider les chercheurs à comprendre et à mener une recherche design science. Ces lignes directrices traitent la conception comme un artefact, un problème de pertinence, une évaluation de la conception, des contributions à la recherche, la rigueur de la recherche, un processus de recherche et de communication des résultats de la recherche. En se basant sur ces lignes directrices, Peffers et al. (Peffers et al., 2007) élaborent un modèle pour présenter la recherche sur les systèmes d'information.

La Figure 1 présente le modèle de processus design science, proposé par Peffers et al. (Peffers et al., 2007), appliqué à notre recherche.

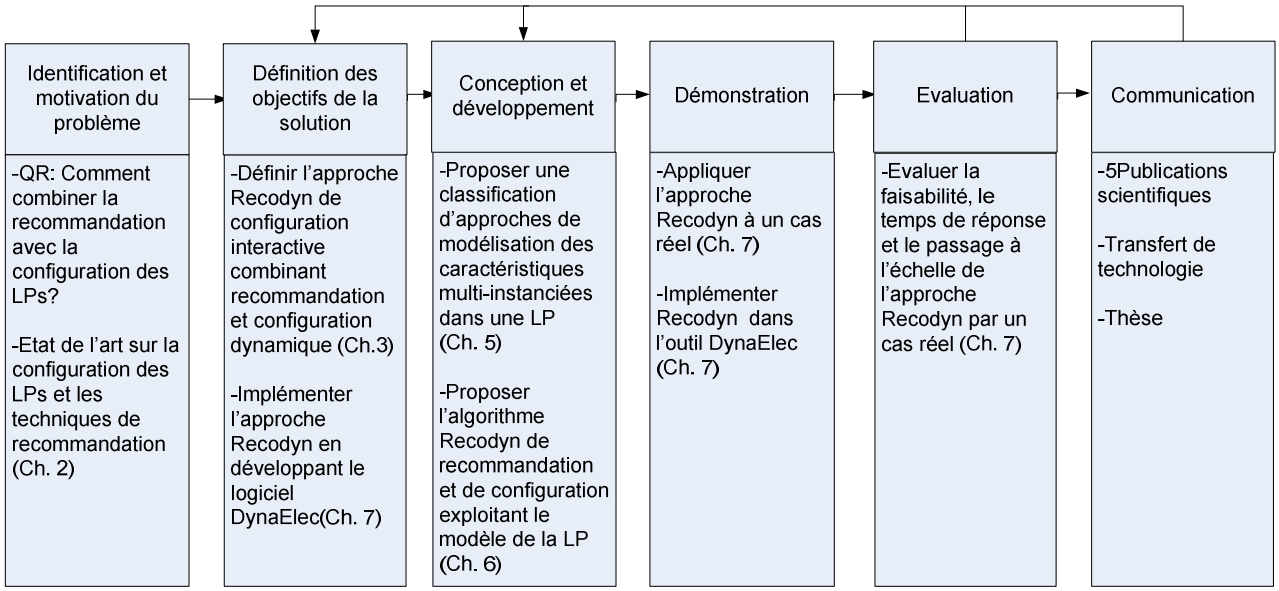


Figure 1 : Modèle du processus design science (Peffers et al., 2007) appliqué à notre recherche

La méthode de recherche choisie vise à valider nos hypothèses de recherche au moyen de prototypes et une étude de cas. Le processus suivant a été mis en oeuvre :

1. Un état de l'art sur la configuration des LPs et les principales TRs a été effectué.
2. La problématique de recherche a été explorée à travers une étude de cas développée dans le contexte d'un projet de collaboration industriel avec la société Rexel.
3. Une analyse critique des TRs et des approches de configuration existantes a été réalisée en se référant aux questions de recherche.

4. Une technique originale de configuration interactive des LPs combinant recommandation et configuration a été proposée.
5. Un algorithme exploitant le MLP a été conçu pour l'approche de configuration interactive proposée, et a été implémenté pour servir à un prototype d'outil.
6. Une évaluation de l'approche de configuration interactive a été menée en utilisant l'étude de cas Rexel.

1.5 Contributions

Cette thèse présente 4 contributions principales:

1. **Un état de l'art sur les techniques de recommandation et configuration des LPs :**
   La revue de la littérature des TRs et de configuration des LPs montre que l'utilisation des approches de configuration uniquement n'est pas suffisante pour aider à la personnalisation des produits, et que la recommandation est nécessaire pour réussir le processus de configuration interactive. En outre, les TRs principales ne sont pas simplement appliquées aux produits configurables. Bien que la littérature offre à la fois des TRs et des techniques de configuration des LPs, une combinaison des deux est une piste originale qui, à notre connaissance, n'avait été explorée dans aucun autre état de l'art auparavant.
2. **L'algorithme Recodyn de recommandation appliqué lors du processus de configurations des LPs :**
   De façon interactive, l'algorithme proposé aide à la configuration des produits en tenant compte des exigences de l'utilisateur. L'approche consiste à décomposer la recommandation en plusieurs étapes portant sur des sous-ensembles de caractéristiques de la LP. Les recommandations sont générées sous la forme de configurations partielles. Le produit est alors crée de manière incrémentale. Une combinaison de la recommandation basée sur les connaissances et la recommandation basée sur le contenu est utilisée pour recommander les caractéristiques, leurs valeurs et leurs quantités. La recommandation basée sur les connaissances prend en compte les exigences des utilisateurs, les contraintes et les dépendances de la LP. La recommandation basée sur le contenu permet de recommander des produits en adéquation avec le profil de l'utilisateur par calcul de similarité. Etant générique, la méthode proposée pourra, par conséquent, s'appliquer à tous types de MLP.
3. **Une mise en œuvre de l'approche :**

Un prototype d'outil a été développé afin de démontrer la faisabilité de l'approche. Ce prototype a été implémenté en utilisant les technologies suivantes : Java, Maven, SQL Server 2008 et JDBC.

4. **Une évaluation de l'approche :**
   Le passage à l'échelle de l'approche a été évalué au moyen d'une analyse qualitative élaborée en utilisant une étude de cas industrielle avec la société de distribution de matériel électronique Rexel.

1.6 L'organisation de la thèse

La suite de cette thèse est structurée comme suit :

Le chapitre 2 présente une revue de la littérature sur la configuration des LPs et les principales TRs. Ce chapitre rapporte les avancements, les limites et les défis discutés dans la littérature.

Le chapitre 3 donne un aperçu de la nouvelle approche de recommandation proposée dans cette thèse. Un exemple d'application est développé pour faire, ensuite, une première présentation générale de l'approche proposée.

Le chapitre 4 présente la méthodologie utilisée pour mener à bien le projet de recherche.

Le chapitre 5 présente l'étude de cas industrielle développée en collaboration avec la société Rexel. Ce chapitre fournit une description détaillée de l'étude de cas, les problèmes rencontrés pour modéliser cette LP et les solutions utilisées pour résoudre ces problèmes en discutant trois stratégies de modélisation des caractéristiques multi-instanciées.

Le chapitre 6 décrit en détails l'approche de recommandation Recodyn pour un guidage interactif de la configuration de la LP. Une synthèse des résultats obtenus par le déploiement de l'algorithme Recodyn est élaborée afin de montrer l'impact de la strategie utilisée pour modéliser la LP.

Le chapitre 7 présente l'application de l'approche Recodyn à l'étude de cas pour effectuer une évaluation qualitative. L'environnement de développement et d'évaluation de Recodyn est détaillé dans ce chapitre, ainsi que les résultats obtenus à partir de cette évaluation.

Le chapitre 8 résume les principaux apports dans cette thèse et évoque les perspectives de recherche les plus pertinentes.

CHAPITRE 2: État de l'art

2.1 Introduction

Ce chapitre présente un état de l'art des approches (a) de configuration des LPs, (b) de recommandation et (c) de configuration utilisant la recommandation. La Section 2 présente la démarche utilisée. Après avoir introduit les principales définitions en Section 3, les différentes approches de configuration sont présentés en Section 4. La recommandation est présentée en Section 5 en mettant en exergue ses différents types de recommandation et les techniques et en identifiant les critères d'analyse utilisés. Ensuite, les méthodes de recommandation utilisées pour la configuration des produits seront présentées à la Section 6. Un tableau récapitulatif clorera le présent chapitre.

2.2 Méthode de recherche

L'objectif principal de cette thèse étant d'intégrer la recommandation au processus de configuration de la LP, la revue de la littérature cherche à développer un panorama complet de tous les types de configuration, et de toutes les TRs existantes, en particulier les approches qui mixent TR et configuration.

Cette revue de la littérature essaye d'apporter des éléments de réponse aux questions suivantes :

Q_1 : Quels sont les types de configuration proposés dans la littérature ?
Q_2: Quels sont les types de recommandation proposés dans la littérature ?
Q_3: Quelles sont les caractéristiques et algorithmes des TRs existantes?
Q_4 : Existe-t-il des TRs utilisées pour la configuration de produits complexes ?

Les études et documents sur lesquels se base cette revue de la littérature ont été identifiés en s'appyant sur les recommandations de Webster & Watson (Webster & Watson, 2002) et Kitchenham (Kitchenham, 2004) qui indiquent comment sélectionner un ensemble de documents candidats dans les journaux, actes de conférences et livres. Environ 50 travaux de recherche ont été sélectionnés dans cet état de l'art. Certains d'entre eux sont liés à l'ingénierie des LPs ; d'autres au domaine de la recommandation.

Les questions Q_1 et Q_2 permettent de définir les cadres de travail des deux domaines auxquels cette thèse s'intéresse en offrant un aperçu sur les types de configuration et de recommandation, et de comprendre les limites de chacun. De plus, des critères sont définis pour analyser les TRs qui implémentent les différents types de recommandation (Q_3). La question Q_4 s'impose au terme de cette revue car il s'agit de

connaitre les travaux de recherche utilisant la recommandation pour des produits complexes configurables pour démontrer l'originalité de celle proposée.

2.3 Définitions

Cette section définit les termes clés nécessaires à la compréhension de cette thèse, à savoir : une caractéristique de produit, une configuration, un processus de configuration, une dérivation, une configuration partielle, une configuration valide, une décision, un configurateur, une recommandation, et la personnalisation.

1. ***Caractéristique de produit :*** En informatique, une caractéristique est une propriété individuelle mesurable d'un phénomène observé (Bishop, 2006). Dans le contexte des LPs, c'est un aspect, une qualité ou un élément relatifs à un produit visible à l'utilisateur (Kang et al., 1990) comme la couleur, le poids, le prix, etc. On notera par F_i une telle caractéristique (« Feature » en anglais).
2. ***Configuration :*** C'est une combinaison de caractéristiques sélectionnées à partir du MLP (Deelstra et al., 2005) respectant les contraintes de la LP et satisfaisant les exigences de l'utilisateur. On la notera (F_i). Elle représente un produit et doit contenir tous les éléments obligatoires de la LP et quelques éléments variables. « La relation entre un MLP et une configuration est comparable à la relation qui existe entre une classe et son instance dans la programmation orientée objet » (Djebbi, 2011).
3. ***Processus de configuration :*** Le processus de configuration est l'activité de dériver un produit valide à partir d'un MLP (Hubaux & Heymans, 2009) de manière à satisfaire les exigences de l'utilisateur et à respecter les contraintes de la LP. Dans un projet réel, le processus de configuration est « *une activité de pilotage d'ingénierie de produit de longue haleine qui peut prendre plusieurs mois et impliquer différents acteurs ayant différentes préoccupations* » (Rabiser et al., 2007). En outre, avec la grande taille des systèmes industriels, les MLP deviennent plus complexes et les dépendances entre les caractéristiques sont difficiles à gérer (Djebbi, 2011), ce qui complique la configuration manuelle de telles LPs. D'où l'automatisation des approches de configuration (Classen et al., 2009) et leur répartition entre les parties prenantes (Czarnecki et al., 2005).
4. ***Dérivation:*** La dérivation est définie comme « *un cas particulier de l'activité de conception où l'artéfact configuré est assemblé à partir d'instances d'un ensemble*

fixe de types de composants bien définis qui peuvent être composés conformément à un ensemble de contraintes » (Sabin & Weigel, 1998).

5. ***Configuration valide :*** C'est une configuration conforme aux contraintes du MLP.
6. ***Configuration partielle :*** Elle représente un produit non-complet de la LP dans le sens où tous les éléments de la LP ne sont pas encore sélectionnés ou complètement traités (Deelstra et al., 2005). Une configuration partielle doit néanmoins être valide.
7. ***Décision :*** Elle consiste à faire un choix parmi d'autres (par exemple choisir ou exclure une caractéristique), selon les exigences de l'utilisateur, tout en raffinant progressivement l'espace de problème (Janota et al., 2010).
8. ***Configurateur :*** Il s'agit d'un outil permettant de déduire les solutions possibles à partir de décisions faites par l'utilisateur et en prenant en compte les contraintes du MLP (Janota et al., 2010).
9. ***Recommandation :*** C'est une forme spécifique de filtrage de l'information visant à présenter l'information pertinente à l'utilisateur selon un ordre jugé le plus pertinent. La recommandation consiste à prédire l'utilité d'un item pour un utilisateur et lui suggère des choix en raisonnant à partir de connaissances diverses.
10. ***Personnalisation :*** La personnalisation est l'action d'adapter quelque chose à une personne selon ses goûts, ses besoins ou ses moyens. Dans le domaine de la recommandation, la personnalisation adapte les recommandations aux goûts et aux besoins de l'utilisateur (Bonnin, 2010).

2.4 Types et approches de configuration

Configurer une LP peut se faire en utilisant différentes approches de configuration et selon deux types : la configuration sans guidage et la configuration avec guidage. La Figure 2 présente l'arborescence relative aux types de configuration et les approches correspondantes.

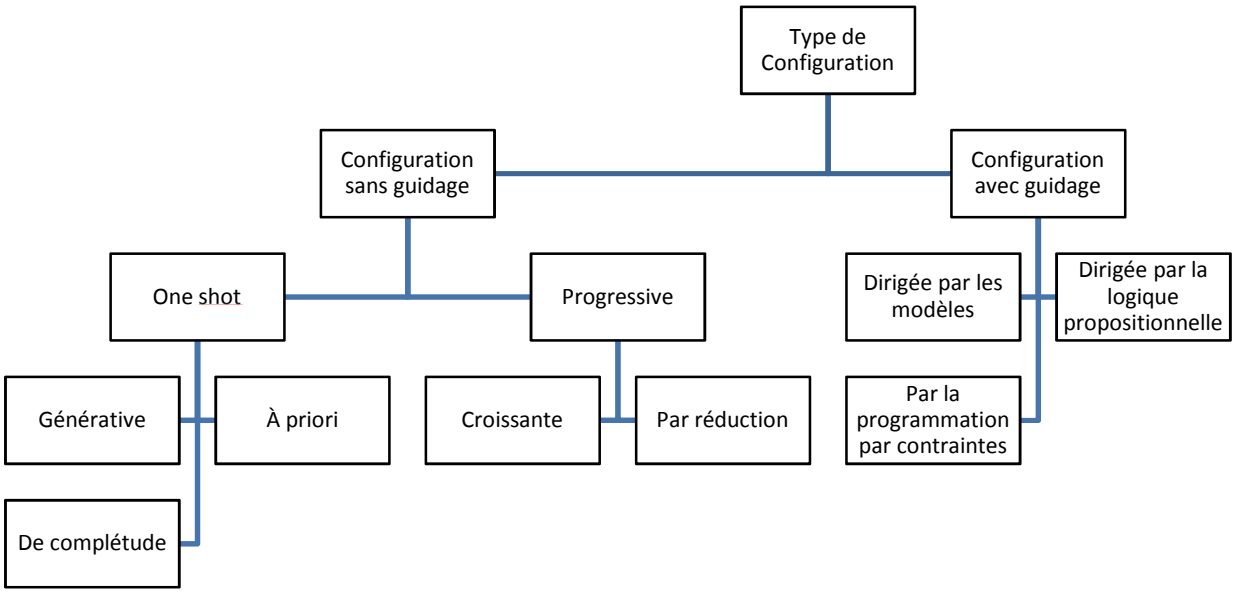


Figure 2 : Arborescence des types de configuration

Les types de configuration et les détails des approches correspondantes sont décrits dans les sections suivantes.

2.4.1 Configuration sans guidage

Dans une configuration non-guidée, le MLP est configuré sans suivre une séquence bien définie. Elle consiste à prendre des décisions en parcourant le MLP de facon aléatoire (Djebbi, 2011). Les processus de configuration appelés « *One Shot* » et « *progressive* » sont des exemples types de configurations non-guidées.

2.4.1.1 Configuration One Shot

Le processus de configuration *One Shot* génère des configurations en bloc sans étape. Il est donc non-itératif, et est adopté quand les configurations sont générées de façon automatique. Il existe trois approches de configuration One shot : *l'approche générative*, *l'approche à priori* et *l'approche de complétude*.

- **L'approche générative**

Dans l'approche générative, l'utilisateur est amené à sélectionner une configuration du MLP de manière descendante. Dans ce cas, le PC consiste à n'effectuer que des opérations de sélection des élements souhaités par l'utilisateur en cherchant à avoir un produit valide à la fin du processus. Plusieurs solutions contenant les éléments séléctionnés peuvent être générées. Toutefois, il ne sera pas possible de les récupérer toutes. Une stratégie naïve de génération de produits est proposée en utilisant un tableau de vérité. Les colonnes de la table de vérité représentent les éléments (ou caractéristiques) de la LP. Il suffit ensuite de remplir la table de vérité avec des valeurs booléennes « Vrai » ou « Faux » en générant exhaustivement toutes les combinaisons. Comme le montre la Figure 3, seules les combinaisons correctes respectant les contraintes de la LP sont finalement retenues dans le tableau. L'utilisateur n'a donc plus qu'à choisir parmi les configurations valides.

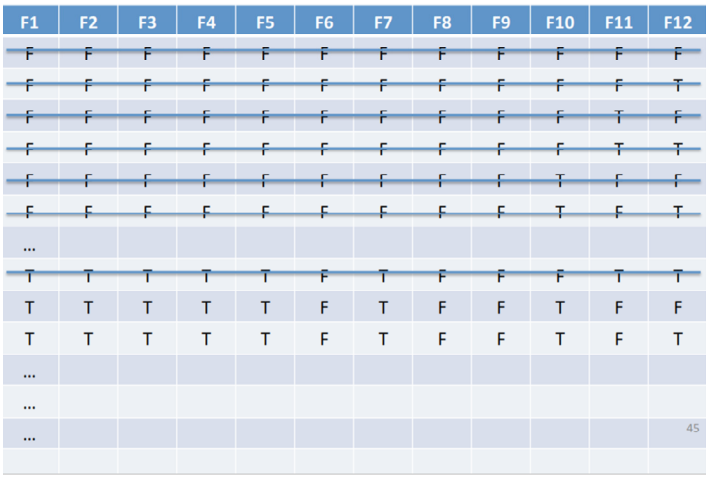


Figure 3 : Exemple de table de vérité énumérant les configurations valides (Salinesi & Mazo, 2010)

Dans cette approche, le nombre de configurations générées (lignes) croit de façon exponentielle en fonction du nombre de caractéristiques (colonnes) : le problème du passage à l'échelle est donc une limite évidente. Pour réduire la croissance exponentielle du nombre de lignes, une stratégie de génération avancée consiste à remplir les lignes de la table de vérité en respectant les contraintes du MLP. Les contraintes prennent la forme d'équations logiques traduisant les dépendances entre les cractéristiques du MLP. La Figure 4 montre un exemple de traduction d'une relation « *requires* » dans les Features Models (FM) (Kang et al., 1990) entre deux caractéristiques en une équation logique et une table de vérité.

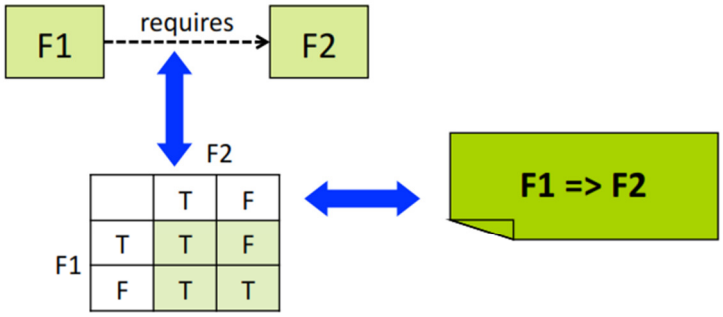


Figure 4 : Exemple de traduction d'une relation « *requires* » (Salinesi & Mazo, 2010)

Bien que la réduction du nombre de configurations générées (lignes) soit important, le problème de passage à l'échelle persiste.

- **L'approche à priori**

L'approche à priori consiste à laisser l'utilisateur définir librement des configurations et tester si elles sont valides. Dans cette approche, l'utilisateur est amené à définir une nouvelle configuration chaque fois que son choix précédent ne respecte pas les contraintes de la LP. Par exemple, l'utilisateur sélectionne la configuration $C_1=\{F_1, F_2, F_3, F_4, F_5\}$ et souhaite savoir si elle est valide. La configuration C_1 est rejetée par le configurateur, car il existe une contrainte qui spécifie que « F_3 *exclut* F_4 ». L'utilisateur doit alors modifier sa configuration et la tester encore une fois, et ce jusqu'à tomber sur une configuration valide.

La limite de cette approche est liée au risque de proposer des configurations non valides : c'est le problème de probabilité (« probabilities issue »).

- **L'approche de complétude**

Elle consiste à compléter une configuration partielle donnée en entrée pour avoir une configuration complète respectant les contraintes de la LP appelée « solution ». Dans ce cas, l'espace de solutions est réduit, comparé à celui de l'approche générative. Cependant, il se peut qu'il n'existe aucune solution contenant la configuration partielle. En revanche, il peut y avoir plusieurs solutions sans pouvoir les générer toutes. Les problèmes de passage à l'échelle et de probabilité persistent donc encore.

2.4.1.2 Configuration progressive

L'approche de configuration *progressive* (Czarnecki et al., 2004) ne génère pas de configuration en bloc mais peut aider l'utilisateur durant le processus de configuration sans pour autant le guider. Il s'agit d'éliminer ou de compléter à partir du MLP des choix de configurations en passant par plusieurs étapes. La configuration est obtenue progressivement par une série d'intéractions entre l'utilisateur et le configurateur. Le caractère progressif du processus de configuration permet soit de compléter une telle configuration ou de revenir en arrière sur les choix faits par l'utilisateur ce qui n'est pas possible avec l'approche One Shot. On distingue deux approches de configuration progressive : *l'approche croissante* et *l'approche par réduction.*

- **L'approche croissante**

L'approche croissante consiste à construire progressivement le produit en sélectionnant les caractéristiques souhaitables à chaque étape. L'idée est de chercher à partir du MLP toutes les configurations contenant une caractéristique *Fi* sélectionnée par l'utilisateur. Une fois une telle configuartion trouvée, une nouvelle caractéristique *Fj* est sélectionnée pour chercher des configurations contenant *Fi* et *Fj*. Ce même processus continue jusqu'à l'obtention d'une configuration *(Fi)* complète. Par exemple, l'utilisateur a sélectionné à chaque étape les caractéristiques optionnelles de la LP qui sont F_5, F_6, F_{10} et F_{11}. En faisant propager les contraintes de la LP, la solution retournée par le configurateur est C_s=*{F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_{10}, F_{11}, F_{12}}*. Toutefois, en retenant les configurations contenant les carcatéristiques sélectionnées, il est possible que la configuration souhaitée soit l'une des configurations rejetées à l'une des étapes précédentes. Par exemple, l'utilisateur souhaite avoir la configuration *Cu={ F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_{10},F_{12}}* qui est déjà rejetée à une étape précédente suite à la propagation des contraintes en sélectionnant F_{11}. C'est le problème d'inaccessibilité.

- **L'approche par réduction**

Contrairement à l'approche croissante, l'approche par réduction consiste à éliminer les caractéristiques non souhaitées. A chaque étape, une caractéristique optionnelle est supprimée réduisant ainsi le nombre de solutions possibles. Il est à noter que le problème d'inaccessibilité persiste encore dans ce cas puisqu'il est possible, à une étape antérieure, de supprimer de l'espace de solutions la configuration souhaitée par l'utilisateur.

En résumé, la configuration sans guidage présente des limites même en utilisant l'approche progressive. En effet, même en cernant les problèmes du passage à l'échelle et de probabilité avec l'approche One Shot, le problème d'inaccessibilité se pose toujours avec l'approche progressive. Ceci est dû au fait que les approches de configuration sans guidage ne sont pas axées sur la spécification explicite des exigences de l'utilisateur.

2.4.2 Configuration avec guidage

La configuration avec guidage consiste à accompagner l'utilisateur et l'orienter vers les solutions correspondant à ses exigences en respectant les contraintes de la LP. Différentes approches sont présentées dans cette section à savoir : l'approche de configuration *dirigée par les modèles*, l'approche de configuration *dirigée par la logique propositionnelle,* l'approche de configuration *par la programmation de contraintes* et l'approche de configuration *utilisant la recommandation*.

2.4.2.1 L'approche de configuration dirigée par les modèles

L'approche de configuration dirigée par les modèles combine deux paradigmes : l'ingénierie dirigée par les modèles et l'ingénierie des LPs utilisées pour configurer une LP en gérant la variabilité par la transformation de modèles. Dans cette approche, la configuration est en effet guidée par un modèle permettant de prendre des décisions sur les élements de la LP qui devraient être sélectionnés.

Ziadi et al. ont formalisé (Ziadi et al., 2003) cette approche de configuration comme étant des transformations successives des modèles UML et ont proposé (Ziadi & Jézéquel, 2005) une technique axée sur la transformantion des modèles UML. Cette activité consiste à transformer le MLP en un modèle de produit (MP) modélisés par un diagramme de classes UML. Pour ce faire, un algorithme de transformation (Ziadi, 2004) décrit en trois étapes a été proposé:

- La sélection des variantes : en fonction des choix de l'utilisateur, les classes variantes et optionnelles sont sélectionnées et introduites dans le MP
- La spécialisation du modèle : toutes les classes optionnelles qui n'ont pas été sélectionnées dans l'étape précédente sont supprimées du MP
- L'optimisation du modèle : le modèle est optimisé pour obtenir le modèle final.

D'autres exemples d'approches de configuration dirigée par les modèles ont été étudié en détails dans la thèse de Djebbi (Djebbi, 2011). On peut citer à titre d'exemple l'approche Kobra (Atkinson et al., 2001) permettant de configurer un produit en se basant sur les choix des utilisateurs décrits par un modèle de décision.

2.4.2.2 L'approche de configuration dirigée par la logique propositionnelle

L'approche de configuration dirigée par la logique propositionnelle consiste à traduire un MLP en formules propositionnelles, et à vérifier la véracité de ces formules pour un modèle de produit valide afin d'en dériver des produits. Une formule propositionnelle est un ensemble de variables booléennes connectées par des opérateurs logiques (¬,Λ,V,→,↔). La véracité de ces formules est analysée par des solveurs de type SAT en déterminant leur satisfaisabilité qui se traduit par l'existance des valeurs logiques pouvant être assignées à des variables donnant à la formule la valeur « Vrai » (Benavides et al., 2006). De manière générale, le solveur permet d'automatiser diverses tâches d'ingénierie comme la vérification de la satisfaisabilité et le calcul des éléments communs (Metzger et al., 2007). Dans le contexte particulier des LPs, les solveurs sont utilisés pour déterminer l'existance de configurations possibles dérivées d'un MLP. Pour plus de détails, le lecteur peut se référer à (Djebbi, 2011).

2.4.2.3 L'approche de configuration par la programmation par contraintes

Afin d'automatiser et guider la configuration des LPs , certains travaux de recherche suggèrent d'utiliser la programmation par contraintes (Benavides et al., 2005) (White et al., 2009) (Salinesi et al., 2010) (Salinesi et al., 2011).

Il s'agit de traduire le MLP en un programme de contraintes appelé « *Constraint Satisfaction Problem* » (CSP) (Apt, 2003) en utilisant le Langage de Contraintes (LC). Le CSP est composé par des contraintes et des opérateurs appliqués à des variables et valeurs (Salinesi et al., 2011). Chaque variable a une valeur à un instant *t* et un domaine définissant l'ensemble de valeurs possibles que peut prendre cette variable. Le domaine peut être booléen, entier, chaine de caractères, intervalle, ou énumération. Les opérateurs peuvent être de deux types (Salinesi et al., 2011):

- Les opérateurs de contraintes : ils sont utilisés pour définir les contraintes lors de la spécification d'une LP telle que l'opérateur symbolique « `at_most (…)` ». Une contrainte peut être symbolique, arithmétique ou booléenne et

contient des opérateurs telle que l'addition (+), la soustraction (-) et la multiplication (*).

- Les opérateurs de résolution : ils sont utilisés pour résoudre les contraintes et par conséquent pour configurer une LP telle que « `find_all(…)` ».

Le CSP est défini par le triplet *(X, D, C)*, où *X* est un ensemble de variables, *D* est un ensemble de domaines, et *C* est un ensemble de contraintes limitant les valeurs que peuvent prendre les variables simultanément.

L'analyse du CSP permet de générer des produits à partir du MLP et déterminer leur validité. La résolution d'un CSP consiste à attribuer des valeurs aux variables de manière à satisfaire toutes les contraintes en utilisant des solveurs. La résolution de contraintes consiste d'abord, à réduire les domaines des variables par des techniques de propagation qui permettront d'éliminer la valeur incompatible dans les domaines. Ensuite, elle cherche des valeurs pour chacune des variables restantes et propage leur impact sur d'autres domaines de variables en réappliquant les mêmes techniques de propagation (Mazo, 2011).

2.5 Types et techniques de recommandation

2.5.1 Types de recommandation

Différents types de recommandation existent dans la littérature : la recommandation collaborative (Resnick et al., 1994), la recommandation basée sur le contenu (Pazzani & Billsus, 1997), la recommandation hybride, la recommandation basée sur la connaissance (Burke, 2000) et la recommandation basée sur les heuristiques (Mazo et al., 2014).

2.5.1.1 Recommandation collaborative

La recommandation collaborative permet de prédire l'utilité d'un produit pour un utilisateur particulier en se basant sur les évaluations de ce produit faites par d'autres utilisateurs. De manière plus formelle, l'utilité $u(c, s)$ d'un produit s pour un utilisateur c est calculée en fonction des utilités $u(c_j, s)$ du produit s estimées par les utilisateur $c_j \in C$ qui lui sont similaires (Adomavicius & Tuzhilin, 2005). Par exemple, afin de recommander des livres à un utilisateur c dans une application de recommandation de livres, le système de recommandation collaborative cherche les utilisateurs ayant des goûts similaires pour les livres. Les livres les plus appréciés par ces utilisateurs seront recommandés à

l'utilisateur *c*. Plusieurs exemples types de systèmes utilisant la recommandation collaborative existent tels que GroupLens (Resnick et al., 1994), Netflix, Amazon.com, etc.

2.5.1.2 Recommandation basée sur le contenu

La recommandation basée sur le contenu prend ses origines dans le Filtrage d'Information (FI) (Salton, 1989). Elle permet de recommander des produits similaires à ceux que l'utilisateur a déjà apprécié. De manière formelle, elle permet de prédire l'utilité $u(c, s)$ d'un produit s en se basant sur les utilités $u(c, s_i)$ assignées par l'utilisateur c aux produits s_i simailaires à s (Adomavicius & Tuzhilin, 2005). Cette similarité est calculée à partir des descriptions des produits et du profil utilisateur en termes de carcatéristiques appelées « métadonnées ». La recommandation basée sur le contenu est alors composée de :

- Descriptions des produits à recommander en fonction de leurs caractéristiques
- Une représentation du profil utilisateur en fonction de ses préférences exprimées avec les caractéristiques des produits, et de son historique de produits appréciés (Pazzani & Billsus, 2007)
- Une méthode pour comparer le profil utilisateur avec les produits, et pour prédire l'utilité *u*

Afin de présenter un produit de manière structurée, des poids sont attribués à ses différentes caractéristiques. Soit F^s_i la caractéristique ayant le numéro *i* dans le produit *s*. Pour chaque caractéristique F^s_i un poids w^s_i est associé indiquant son degré d'importance dans *s*. Ainsi, un vecteur de poids noté w^s est associé à chaque produit. Cette suite de vecteurs génère un espace vectoriel associé au produit. Une des mesures les plus connues dans le FI est TF-IDF (Term Frequency-Inverse Document Frequency) (Salton, 1989) qui permet de calculer la valeur des poids avec la formule **([1])** :

$$W_{i,s} = Tf_{i,s} \times \log(\frac{n}{DF_i}) \quad \textbf{[1]}$$

Avec :

- n: le nombre de produits dans la collection
- $Tf_{i,s}$: le nombre d'occurrences de la caractéristique F_i du produit s
- DF_i: le nombre de produits contenant au moins une fois la caractéristique F_i

Nous identifions, ainsi, un ensemble appelé « *produit(S)»* **([2])** définissant le produit.

[2] $produit(S) = \{(F_i^s, w_i^s)\}, i = 1..n$

Le profil utilisateur peut être construit de manière explicite à partir de questionnaires sur les caractéristiques que l'utilisateur aime ou n'aime pas, ou implicitement à partir d'évaluations ou d'actions précédentes sur les produits comme l'achat, la recherche, etc.

Plusieurs méthodes (Pazzani & Billsus, 2007) d'apprentissage du modèle utilisateur utilisent les caractéristiques des produits. Ces méthodes seront présentées à la section 2.5.2.

Reprenons l'exemple précédent concernant l'application de recommandation de livres. Le système de recommandation basée sur le contenu analyse les caractéristiques communes des livres telles que le nom de l'écrivain, le genre, le sujet, la collection, etc. appréciées par l'utilisateur pour recommander ceux qui sont similaires aux préférences de l'utilisateur.

2.5.1.3 Recommandation basée sur la connaissance

L'idée clé de la recommandation basée sur la connaissance repose sur le principe défini par la phrase suivante : « *Dites-moi ce qui me convient en fonction de mes besoins* » (Jannach et al., 2010).

En utilisant les différents modèles définis à partir des caractéristiques des produits, le configurateur propose ce qui pourrait convenir à l'utilisateur (Burke, 2000) (Rao & Talwar, 2008). Ce qui caractérise ce type de recommandation est qu'il met l'accent sur les besoins des utilisateurs et la manière de les satisfaire (Felfernig & Burke, 2008). Les approches de recommandation basée sur la connaissance peuvent être classées en deux types selon (Felfernig & Burke, 2008) : la recommandation basée sur le cas (Burke, 2000) (Mirzadeh et al., 2005) et la recommandation basée sur les contraintes (Felfernig et al., 2007). Mazo et al (Mazo et al., 2014) considère que l'approche de recommandation basée sur les heuristiques fait également partie de la recommandation basée sur les connaissances.

Dans les trois approches, le processus de recommandation commence par la spécification des besoins de l'utilisateur pour lui recommander des solutions. Toutefois, l'utilisateur est amené à modifier ses besoins au cas où aucune solution n'a été trouvée.

La recommandation basée sur le cas utilise un raisonnement par réutilisation de cas (problème, situation) passés similaires au cas actuel pour trouver des solutions (Aamodt & Plaza, 1994) (Mirzadeh et al., 2005).

Dans la recommandation basée sur les contraintes, les solutions sont générées en respectant un ensemble de contraintes ou « règles de recommandation » (Felfernig et al.,

2014) spécifiées explicitement par l'utilisateur. Les exigences, les propriétés des produits et les contraintes forment « *une base de connaissances du système de recommandation* » (Felfernig & Burke, 2008). Dans ce contexte, Felfernig et Burke définissent une activité de recommandation comme un CSP (U, P, $CR \cup COMP \cup FILT \cup PROD$) avec :

- U : un ensemble fini de variables représentant les exigences potentielles de l'utilisateur.
- P : un ensemble de variables définissant les propriétés de base d'un assortiment de produits.
- CR : un ensemble d'exigences potentielles.
- $COMP$: un ensemble de contraintes de (in)compatibilité limitant l'ensemble d'exigences possibles.
- $FILT$: un ensemble de contraintes de filtrage définissant les relations entre les exigences de l'utilisateur et les produits.
- $PROD$: un ensemble de produits proposés sous forme de contraintes de produits limitant les instanciations possibles dans P.

La solution pour un tel CSP est une affectation complète aux variables (U, P) de manière qu'elle soit cohérente avec les contraintes $\in$ ($CR \cup COMP \cup FILT \cup PROD$).

Il est à noter que la recommandation basée sur les contraintes proposée par (Felfernig & Burke, 2008), peut être assimilée à la configuration des LPs par la programmation par contraintes. Elle est également considérée comme une recommandation dynamique proposée pour recommander des valeurs de caractéristiques (Falkner et al., 2011).

La recommandation basée sur les heuristiques (Mazo et al., 2014) permet d'améliorer le processus de configuration du MLP. Ce type de recommandation est basée sur un ensemble de connaissances implémentées sous forme d'heuristiques pour recommander les éléments à configurer. Six heuristiques de recommandation ont été présentées par Mazo et al. (Mazo et al., 2014) afin d'améliorer l'interactivité de configuration des LPs en l'adaptant à des situations d'ingénierie courantes. En particulier, ces heuristiques sont destinées à éviter des prises de décisions inutiles ou inefficaces. Ils décrivent les principes, les avantages et l'application de chaque méthode heuristique à l'aide de programmation par contraintes. Les heuristiques sont les suivantes :

- *Heuristique 1 :* les variables avec le plus petit domaine sont choisies en premier
- *Heuristique 2 :* les variables ayant le plus de contraintes sont choisies en premier
- *Heuristique 3 :* les variables présentes dans la plupart des produits sont choisies en premier
- *Heuristique 4 :* la configuration des variables est automatiquement complétée quand il n'y a aucun choix
- *Heuristique 5 :* les variables requises par la dernière variable configurée sont choisies en premier
- *Heuristique 6 :* les variables divisant l'espace de problème en deux sont choisies en premier

Contrairement à l'approche proposée par (Felfernig & Burke, 2008) utilisant la recommandation basée sur les contraintes, cette approche ne consiste pas à recommander un produit mais à recommander les caractéristiques d'un produit en utilisant les heuristiques et sans utiliser la similarité avec les configurations passées.

2.5.1.4 Recommandation hybride

La recommandation hybride combine deux ou plusieurs types de recommandation pour offrir une meilleure performance du système de recommandation et surmonter les limites individuelles de chaque type (Burke, 2002). A titre d'exemple, la combinaison de la recommandation collaborative et celle basée sur le contenu sont souvent utilisées dans la littérature.

Différentes méthodes d'hybridation ont été présentées par Burke (Burke, 2002) (Burke, 2007) :

- *Hybridation pondérée :* Il s'agit d'utiliser plusieurs systèmes de recommandation pour générer des solutions en faisant une union ou une intersection des résultats fournis par chaque système. Les scores des éléments recommandés et pondérés par chaque système sont combinés pour les recommander à l'utilisateur.
- *Hybridation par commutation :* Cette méthode ne combine pas les résultats des systèmes de recommandation mais en sélectionne un de façon dynamique lors

du processus de recommandation selon un critère qui doit être déterminé, ce qui introduit une complexité supplémentaire dans le processus de recommandation.

- *Hybridation mixe :* Elle consiste à combiner différentes solutions générées simultanément par plusieurs systèmes de recommandation.
- *Hybridation par combinaison de fonctionnalités :* Il s'agit d'exploiter des données habituellement utilisées pour un type de recommandation (ex. recommandation collaborative) en faveur d'un autre type de recommandation (ex. recommandation basée sur le contenu).
- *Hybridation en cascade :* Ce type d'hybridation est hiérarchique. Les solutions générées par un système de recommandation sont affinées pour être utilisées par le système de recommandation suivant.
- *Hybridation par augmentation de fonctionnalité :* Au lieu d'utiliser les fonctionnalités déduites des systèmes de recommandation existants, cette méthode génère une nouvelle fonctionnalité en utilisant *la logique* adoptée par les différents types de ces systèmes.
- *Hybridation méta-niveau :* Cette méthode d'hybridation utilise en entrée un modèle généré par un système de recommandation antécédent. Contrairement à l'hybridation par augmentation, cette méthode demande en entrée un remplacement total du modèle généré du système antécédent.

2.5.2 Principales Techniques de Recommandation

Différentes TRs existent dans la littérature. Elles concernent particulièrement la recommandation collaborative et celle basée sur le contenu. Les principales techniques sont présentées dans cette section en suivant cette classification.

2.5.2.1 Recommandation collaborative

Deux types d'approches de recommandation collaborative existent : les approches basés mémoire et les approches basées modèles (Breese et al., 1998). Les techniques les plus communes des approches basées mémoire sont : la technique des k-plus proches voisins et la technique probabiliste.

2.5.2.1.1 Technique des k-plus proches voisins

Cette approche est aussi appelée l'approche des k-plus proches voisins (kNN pour k-nearest neighbors en anglais) (Resnick et al., 1994) (Sarwar et al., 2001) (Linden et al., 2003). En se basant sur la notion du « *voisinage* », cette approche permet de former une collection d'utilisateurs qui sont similaires à un utilisateur *c* à partir de laquelle une suite d'informations quantifiées (ex. évaluations) relatives aux produits est extraite. Ces données permettent de calculer la prédiction d'évaluation d'un produit par l'utilisateur c. Pour ce faire, deux étapes existent pour recommander des produits à un utilisateur :

- L'étape de formation du voisinage permettant de définir la collection d'utilisateurs « *voisins* » est basée sur le calcul de similarité entre utilisateurs.
- L'étape de recommandation permet de prédire l'évaluation d'un ensemble de produits et de sélectionner ceux à recommander.

Il est à noter que la notion du voisinage peut être aussi appliquée aux produits pour définir une collection de produits similaires au produit *s*.

L'exemple ci-dessous est proposé pour mieux expliquer cette approche et ses deux étapes.

Traitement d'exemple

L'exemple de la Figure 5 présente un tableau de correspondance entre trois utilisateurs et quatre produits évalués. Chaque utilisateur attribue une note de 0 à 10 (du mauvais à excellent). Le tiret (-) représente l'absence d'évaluation.

	Produit1	Produit2	Produit3	Produit4
Lucie	5	-	2	8
Pierre	4	9	-	5
Sophie	-	8	3	7

Figure 5: Exemple d'une matrice utilisateur-produit

Deux variantes existent pour appliquer l'approche kNN: celle basée sur les utilisateurs et celle basée sur les produits.

Méthode basée utilisateur

La méthode basée utilisateur (Resnick et al., 1994) (Shardanand & Maes, 1995) permet de prédire l'évaluation $p_{c,s}$ d'un produit *s* pour l'utilisateur actuel *c* en se basant sur les évaluations précédentes données au produit *s* par les k plus proches utilisateurs **([3])**.

[3] $p_{c,s}= \overline{v}_c + k \sum_{i=1}^{n} sim(c, c_i)(v_{i,s} - \overline{v}_i)$

Avec :

- n : le nombre d'utilisateurs considérés
- $v_{i,s}$: l'évaluation du produit s par l'utilisateur c_i
- $\overline{v_i}$, $\overline{v_c}$: la moyenne des évaluations données par l'utilisateur c_i (resp. l'utilisateur c)

$$\overline{v_i} = \frac{1}{|s_i|}\sum_{j \epsilon s_i} v_{i,j}$$

- $sim(c, c_i)$: le score de similarité entre l'utilisateur c et c_i
- k : un coefficient de normalisation telle que la somme des mesures de similarité soit égale à 1

Le score de similarité $sim(c, c_i)$ est une mesure de distance entre les évaluations des produits faites par les deux utilisateurs c et c_i. Plusieurs méthodes peuvent être utilisées pour calculer $sim(c, c_i)$. Les plus populaires parmi celles-ci sont la méthode du cosinus **([4])** (Breese et al., 1998) (Sarwar et al., 2001) et celle du coefficient de Pearson **([5])** (Resnick et al., 1994).

La méthode du cosinus considère les deux utilisateurs c et c_i comme deux vecteurs dans un espace de dimension $d = |S_{c,c_i}|$ (S_{c,c_i} est l'ensemble des produits évalués simultanément par c et c_i). La similarité entre les deux vecteurs est mesurée en calculant le cosinus de l'angle entre eux :

[4] $$sim(c, c_i) = \cos(\overrightarrow{c}, \overrightarrow{c_i}) = \sum_{j \in S_{c,c_i}} \frac{v_{c,j}}{\sqrt{\sum_{k \in S_c} v_{c,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k \in S_{c_i}} v_{i,k}^2}}$$

La méthode du coefficient de Pearson calcule la similarité en utilisant la formule suivante :

[5] $$sim(c, c_i) = \frac{\sum_{j \in S_{c,c_i}} (v_{c,j} - \overline{v_c})(v_{i,j} - \overline{v_i})}{\sqrt{\sum_{j \in S_{c,c_i}} (v_{c,j} - \overline{v_c})^2 \sum_{j \in S_{c,c_i}} (v_{i,j} - \overline{v_i})^2}}$$

Avec :

- S_c, S_{c_i}: l'ensemble des produits évalués par l'utilisateur c (resp. c_i)
- S_{c,c_i} : l'ensemble des produits évalués à la fois par c et c_i
- $v_{c,j}$: l'évaluation du produit j par l'utilisateur actuel c
- $\overline{v_c}$: la moyenne des évaluations données par l'utilisateur actuel c

- $v_{i,j}$: l'évaluation donnée par l'utilisateur c_i
- $\overline{v_i}$: la moyenne des évaluations de l'utilisateur c_i

Pour l'exemple de la Figure 5, la mesure de similarité calculée par **[4]** est entre:

- *Lucie* et *Pierre* : $sim\ (Lucie, Pierre) = \frac{(5*4)+(8*5)}{\sqrt{5^2+2^2+8^2}\sqrt{4^2+9^2+5^2}} = 0.56$
- *Lucie* et *Sophie* : $sim\ (Lucie, Sophie) = 0.58$
- *Pierre* et *Sophie*: $sim\ (Pierre, Sophie) = 0.87$

Les utlisateurs *Pierre* et *Sophie* sont les plus proches voisins.

Sophie n'ayant attribué aucune valeur au *produit1*, la prédiction relative à ce produit est pour elle :

$$p_{Sophie,produit\ 1} = \frac{(8+3+7)}{3} + \left[0.87\left(4 - \frac{18}{3}\right)\right] = 4.24 \approx 4$$

Ce résultat est attendu puisque *Pierre* et *Sophie* sont les plus proches voisins.

Méthode basée produit

Cette méthode (Sarwar et al., 2001) compare les produits en se basant sur un modèle d'évaluations données par les utilisateurs. Il s'agit, en effet, de déterminer les k plus proches produits à *s* en mesurant leur similarité.

Le calcul de similarité entre deux produits se fait de la même manière que celui entre deux utilisateurs en utilisant soit la méthode du cosinus soit le coéfficient de Pearson.

La prédiction d'évaluation est calculée suivant la formule de pondération **([6])**.

[6] $p_{c,s} = \overline{v^s} + \frac{\sum_{i=1}^{m} sim(s,s_i)\left(v^{c,s_i} - \overline{v^{s_i}}\right)}{\sum_{i=1}^{n} |sim(s,s_i)|}$

Avec :

- m *:* le nombre de produits considérés
- $\overline{v^s}$, $\overline{v^{s_i}}$: la moyenne des évaluations donnée au produit *s* actuel (resp. s_i)
- $sim(s, s_i)$: la similarité entre les produits *s* et s_i
- v^{c,s_i} : l'évaluation du produit s_i donnée par l'utilisateur *c*

Pour l'exemple de la Figure 5, la mesure de similarité calculée par **([5])** et relative au *produit1* est:

- *sim (prod1, prod2)*= $\frac{(4*9)}{\sqrt{5^2+4^2}\sqrt{9^2+8^2}} = 0.46$
- *sim (prod1, prod3)*= 0.43
- *sim (prod1, prod4)*= 0.79

Le *produit 4* est le plus proche voisin du *produit 1*. La prédiction de l'évaluation de ce produit pour *Sophie* est:

$$p_{Sophie,produit\ 1} = \frac{5+4}{2} + \frac{0.79\left(7 - \frac{8+5+7}{3}\right)}{0.79} = 4.83$$

Les valeurs de la prédiction $p_{Sophie,produit\ 1}$ calculées par les deux variantes de l'approche kNN sont proches.

Bien que les résultats numériques obtenus d'un exemple aléatoire sont cohérents, les approches kNN présentent, d'une façon générale, deux problèmes majeurs (Koren, 2010):

- *Le passage à l'échelle :* le calcul d'une matrice de voisinage est une fonction quadratique du nombre d'objets à comparer avec une complexité $\boldsymbol{O(n^2 \times m \times k)}$ dans le cas de l'approche basée utilisateur et une complexité $\boldsymbol{O(k)}$ pour le calcul d'une prédiction d'évaluation (Meyer, 2012), où $\boldsymbol{n}$ est le nombre d'utilisateurs, $\boldsymbol{m}$ le nombre de produits et $\boldsymbol{k}$ le nombre d'évaluations. Cette forte complexité combinatoire empêche le passage à l'échelle pour un grand nombre d'utilisateurs et produits.
- *Prédictions peu pertinentes :* généralement le nombre de produits communément évalués par les utilisateurs est faible ce qui entraine une matrice de faible densité (« *sparse* ») et par conséquent des prédictions peu pertinentes.

2.5.2.1.2 Technique probabiliste

Cette approche est basée sur l'apprentissage de modèle en liant les utilisateurs, les produits et leurs évaluations. Tenant compte du profil utilisateur et de ses anciennes évaluations, des nouvelles évaluations $p_{c,s}$ sont prédites en utilisant une approche probabiliste de recommandation collaborative (Breese et al., 1998) **([7])**:

[7] $p_{c,s} = E(v_{c,s}) = \sum_{i=0}^{n} i \times Pr\left((v_{c,s} = i) \middle| v_{c,s'}, s' \in S_c\right)$

Cette expression de probabilité consiste à calculer la probabilité qu'un utilisateur *c* attribue une évaluation au produit *s* à partir de ses évaluations précédentes. Les valeurs des évaluations sont des entiers compris entre 0 et *n*. Pour ce faire, Breese et al (Breese et al.,

1998) proposent l'utilisation de deux modèles : *le modèle de clusters* et *le modèle à base de réseaux bayésiens*.

Le modèle de clusters consiste à grouper dans une même classe les utilisateurs partageant les mêmes préférences. Ce modèle relie les probabilités des classes et des évaluations des utilisateurs sous la forme d'un modèle bayésien naïf (Chickering et al., 1997). Le nombre de classes et les paramètres du modèle sont calculés à partir de ces données. La probabilité **([8])** d'observer un individu d'une classe particulière se calcule en se référant à la règle de Bayes et à l'hypothèse naïve de Bayes supposant que les caractéristiques sont indépendantes.

[8] $Pr(C = c, v_1, \dots, v_n) = Pr(C = c) \prod_{i=1}^{n} Pr(v_i | C = c)$

L'objectif est de maximiser la probabilité à postériori de la classe de tout produit pour classer les données fournies (Meyer, 2012).

Le deuxième modèle à base de réseaux bayésiens représente chaque produit comme un nœud. L'état de chaque nœud correspond aux valeurs d'évaluations possibles, et les évaluations manquantes sont associées à une valeur dénommée « *pas d'évaluation* ». Ce modèle est construit à partir des données en appliquant un algorithme d'apprentissage (Chickering et al., 1997) qui cherche sur plusieurs structures de modèle des dépendances pour chaque produit. Dans le réseau résultant de l'apprentissage, chaque produit a un ensemble de produits « *parent* » qui sont les meilleurs prédicteurs de ses évaluations. Le résultat est alors sous forme d'arbres de décision représentant chaque table de probabilité pour chaque nœud (Oufaida & Nouali, 2009).

D'autre techniques d'apprentissage ont été proposées pour construire les modèles telles que l'utilisation des réseaux de neurones artificiels couplés avec une méthode d'analyse de données comme le Singular Decomposition Value (SVD) (Billsus & Pazzani, 1998).

Billsus & Pazzani (Billsus & Pazzani, 1998) et Breese et al. (Breese et al.,1998) affirment que les approches basées modèles sont plus performantes que les approches basées utilisateurs en terme de précision de recommandations fournies et de temps de réponse. Cette conclusion est confirmée par Sarwar et al. (Sarwar et al., 2001) et notamment à travers les contributions de Netflix (Bell et al., 2007). Les approches basées modèle ont l'avantage de mieux traiter le problème de manque d'évaluations en regroupant les utilisateurs ou les produits en classes. Toutefois, quand le nombre d'utilisateurs et/ou produits croît, ces modèles deviennent non efficaces (Yu et al., 2004) d'où le problème du

passage à l'échelle persiste donc encore. D'autre part, une classe pourrait contenir un utilisateur unique ce qui entraine des recommandations peu pertinentes.

2.5.2.1.3 Avantages et limites de la recommandation collaborative

Le principal avantage de la recommandation collaborative est qu'elle effectue une sélection basée sur les opinions des utilisateurs concernant les produits ce qui reflète leurs goûts, leurs préférences et d'autres facteurs de pertinence utiles aux utilisateurs. En effet, en évaluant positivement un produit, l'utilisateur affirme que ce produit l'intéresse et qu'il est de bonne qualité à la fois.

Toutefois, le nombre d'évaluations attribuées par les utilisateurs est toujours faible par rapport au nombre de prédictions d'évaluations ce qui engendre des matrices utilisateur/ produit *peu dense*. En outre, la faible probabilité que deux produits donnés soient évalués en commun par un couple donné d'utilisateurs implique des recommandations *peu pertinentes*. D'autre part, la recommandation collaborative souffre du problème appelé « *démarrage à froid* » qui se manifeste par l'absence d'évaluations dans les profils des nouveaux utilisateurs ce qui engendre une difficulté de prédiction.

2.5.2.2 Recommandation basée sur le contenu

Cette section présente les principales méthodes utilisées pour l'apprentissage du modèle utilisateur dans une recommandation basée sur le contenu. La première approche est une méthode heuristique qui est kNN et la deuxième est une méthode basée modèle. D'autres approches ont été présentées par Pazzani et Billsus (Pazzani & Billsus, 2007).

2.5.2.2.1 Technique des k-plus proches voisins

La recommandation basée sur le contenu utilise également la technique des k-plus proches voisins pour classer les produits en utilisant la formule du cosinus ***([4])***. Cette similarité est obtenue en calculant le rapport entre le profil de l'utilisateur et celui du produit *s*, en attribuant des poids aux différentes caractéristiques des produits existantes dans le profil utilisateur. Pour chaque utilisateur, un vecteur de poids noté w^c est associé. Si « *profil_Utilisateur(C)* » **([2'])** est l'ensemble définissant le profil utilisateur, la formule **([4])** devient **([4'])**.

[2'] $profil\_Utilisateur(C) = \{(F_i^c, w_i^c)\}, i = 1..n$

[4'] $sim(c, s) = \cos(\vec{c}, \vec{s}) = \frac{c.s}{\|c\| \cdot \|s\|} = \sum_{i=1}^{k} \frac{w_i^s}{\sqrt{\sum_{i=1}^{k} w^2{}_i^s}} \frac{w_i^c}{\sqrt{\sum_{i=1}^{k} w^2{}_i^c}}$

Avec :

- w_i^s : le poids de la caractéristique F_i dans le profil produit
- w_i^c : le poids de la caractéristique F_i dans le profil utilisateur

Par exemple, si un utilisateur est passionné par les livres écrit par « Dan Brown », il lui sera recommandé les livres dont l'écrivain est « Dan Brown ».

Cette technique a été appliquée à plusieurs applications de classification de texte (Yang, 1999) (Cohen & Hirsh, 1998), et malgré sa simplicité ses performances sont compétitives avec d'autres techniques plus complexes (Pazzani & Billsus, 2007).

2.5.2.2.2 Techniques probabilistes de la classification naïve bayésienne

Outre la méthode du calcul de similarité, il existe plusieurs techniques basées sur l'apprentisage automatique de modèle en utilisant des techniques telles que la classification naïve bayésienne (Pazzani & Billsus, 1997), les réseaux de neurones (Pazzani & Billsus, 1997) et les arbres de décisions (Zhang et al., 2002). Cette section prend la classification naïve bayésienne comme exemple de méthode pour son utilisation fréquente dans la littérature.

La classification bayésienne consiste à *apprendre* un modèle liant les données du profil utilisateur et les caractéristiques des produits pour prédire l'utilité d'un produit à un autre utilisateur. Il s'agit, en effet, de répartir les produits en classes et d'estimer la probabilité qu'un produit appartienne à une certaine classe C_i étant donnés les caractéristiques de ce produit. Cette probabilité se calcule en se référant à la règle de Bayes et à la condition d'indépendance (hypothèse naïve de Bayes) donnant la formule **([8])** dans la recommandation collaborative. Dans ce contexte de recommandation basée sur le contenu, Pazzani et Billsus (Pazzani & Billsus, 1997) utilisent la classification naïve bayésienne (Duda et al., 2001) pour classifier des pages web en estimant la probabilité qu'une page p_j appartienne à une certaine classe C_i (pertinente ou non-pertinente) en prenant en compte les mots-clés $k_{1,j}$,…, $k_{n,j}$ de p_j et la condition d'indépendance des mots-clés. Cette probabilité **([8'])** est calculée comme suit :

[8'] $p(C_i|k_{1,j}, \dots, k_{n,j}) = p(C_i) \prod_{x=1}^{n} p(k_{x,j}|C_i)$

Bien que la condition d'indépendance ne soit pas toujours appliquée, des résultats expérimentaux démontrent que la classification naïve bayésienne donne des résultats de classification très précis (Pazzani & Billsus, 1997) (Pazzani & Billsus, 2007).

2.5.2.2.3 Avantages et limites de la recommandation basée sur le contenu

Contrairement à la recommandation collaborative, celle basée sur le contenu permet de générer des recommandations des nouveaux produits même en l'absence d'évaluations. Elle permet également de fournir des recommandations pour des utilisateurs ayant des goûts différents en se basant uniquement sur les produits présentant des caractéristiques similaires.

Toutefois, le problème du nouvel utilisateur persiste car un utilisateur n'ayant jamais évalué un produit ne pourra pas avoir des recommandations pertinentes. En outre, ce type de recommandation ne permet pas de distinguer deux produits différents représentés par le même sous ensemble de caractéristiques (Adomavicius & Tuzhilin, 2005). La recommandation porte uniquement sur les produits similaires à ceux que l'utilisateur a apprécié précédemment. Ce type de recommandation ne permet alors pas de proposer d'autres produits qui pourront intéresser l'utilisateur sans être forcément similaires : c'est *l'effet entonnoir*.

2.6 Techniques de configuration utilisant la recommandation

Guider le processus de configuration en utilisant la recommandation permet de réduire les problèmes de probabilité, inaccessibilté et passage à l'échelle causés par une configuration non guidée lors de la sélection des caractéristiques. Différentes méthodes existent pour aider à la sélection des caractéristiques telles que la sélection par inclusion ou exclusion, le classement des caractéristiques, la sélection collaborative et les sélections basées sur la popularité, l'entropie et l'utilité.

- *La sélection par inclusion ou exclusion* (Falkner et al., 2011)*:*
  Une fois que l'utilisateur a spécifié ses exigences en sélectionnant une caractéristique F_i, cette méthode inclut ou exclut toutes les caractéristiques ayant des relations « particulières » avec F_i. En effet, toutes les sous-caractéristiques obligatoires de F_i et celles ayant une relation de dépendance d'implication avec F_i telle que la dépendance « *requires* » dans les FMs seront incluses. Par analogie avec la sélection par inclusion citée précédemment, la

sélection par exclusion exclut toute caractéristique ayant une relation d'exclusion (comme « *excludes* » en FM) et/ou à choix alternatif (comme « *XOR* ») avec Fi.

Il est à noter que cette méthode de sélection peut être assimilée à l'approche de configuration des LPs dirigée par les modèles.

- *Le classement « Ranking » des caractéristiques :* Il s'agit de classer les caractéristiques selon leur importance pour l'utilisateur (Falkner et al., 2011), permettant d'optimiser la sélection des caractéristiques pour favoriser les plus pertinentes.
- *La sélection collaborative :* Cette méthode permet de sélectionner les caractéristiques pertinentes en utilisant les principes de la recommandation collaborative (Herlocker et al., 2004). En effet, les caractéristiques pertinentes vis-à-vis de l'utilisateur sont recommandées en se basant sur les évaluations des autres utilisateurs similaires. L'approche de recommandation collaborative et ses techniques sont détaillées dans la suite de ce chapitre.
- *La sélection basée sur la popularité :* Il s'agit d'aider l'utilisateur à sélectionner les caractéristiques les plus pertinentes en se basant sur leur popularité. Cette métrique (popularité) est obtenue en calculant le rapport entre le nombre de fois de sélection de la caractéristique F_i et le nombre total de sélection de toutes les caractéristiques. Elle traduit l'utilisation fréquente d'une caractéristique par les utilisateurs (Mirzadeh & Ricci, 2007). Bien que la popularité permette de sélectionner les caractéristiques les plus pertinentes, elle ne permet pas de minimiser le nombre d'interactions entre l'utilisateur et le configurateur (Mirzadeh & Ricci, 2007).
- *La sélection basée sur l'entropie :* L'entropie (valeur de l'information) d'une caractéristique est une mesure de la quantité d'information fournie par cette caractéristique destinée à identifier un produit (Witten & Frank, 2000). La caractéristique ayant l'entropie la plus élevée a plus d'information permettant d'identifier un produit. En l'occurrence, l'utilisation de l'entropie permet d'identifier les caractéristiques ayant le plus d'informations dans le but de minimiser le nombre d'interactions entre l'utilisateur et le configurateur. Toutefois, l'entropie ne prend pas en compte la pertinence des caractéristiques.

- *La sélection basée sur l'utilité :* Cette méthode permet de surmonter les limites de la sélection basée sur la popularité et celle basée sur l'entropie en combinant les deux. La sélection basée sur l'utilité consiste, alors, à chercher les caractéristiques ayant le plus d'information disponible et qui sont les plus pertinentes (Mirzadeh & Ricci, 2007).

Outre l'utilisation de la recommandation des caractéristiques pour guider la configuration, la recommandation des valeurs des caractéristiques fournit des indications sur leurs instanciations possibles (Falkner et al., 2011). Différentes approches de recommandation des valeurs de caractéristiques existent et sont énumérées par Falkner et al. (Falkner et al., 2011): la recommandation statique, la recommandation dynamique, la recommandation par similarité et la recommandation par les approches probabilistes.

- *La recommandation statique :* Elle ne tient pas compte le contexte du l'utilisateur en attribuant par défaut des valeurs aux caractéristiques et ceci indépendamment des valeurs d'autres caractéristiques.
- *La recommandation dynamique :* Contrairement à la recommandation statique, la recommandation dynamique attribue des valeurs aux caractéristiques en prenant en compte les valeurs d'autres caractéristiques comme la recommandation basée sur des règles (ou contraintes).
- *La recommandation par similarité :* Elle recommande des valeurs en se basant sur la similarité avec des configurations passées en utilisant par exemple la technique des k-plus proches voisins (kNN) (Cöster et al., 2002). Cette technique est détaillée dans la section 2.5.3.
- *La recommandation par les approches probabilistes :* Cette approche utilise la classification naïve bayésienne pour estimer la probabilité d'avoir une certaine valeur d'une caractéristique (Cöster et al., 2002) (Tiihonen & Felfernig, 2010). Cette technique de classification est détaillée dans la section 2.5.3.

2.7 Conclusion

Ce chapitre a présenté un état de l'art en répondant aux questions soulevées dans la méthode de recherche. Cette revue de la littérature distingue deux types de configuration de LPs : la configuration guidée et celle non guidée. Pour chaque type, diverses approches ont été présentées. Cette thèse s'intéresse à la configuration guidée et en particulier à la

configuration utilisant la recommandation et celle utilisant la programmation par contraintes. D'autre part, les différents types de recommandation et leurs principales techniques ont été présentées. Les techniques ont été classifiées selon le type de recommandation et leurs natures (méthode basée mémoire, méthode basée modèle, etc.). Le Tableau 1 présente un croisement entre les types de recommandation utilisés pour la configuration des produits complexes. Ainsi, les lignes du tabelau présentent les différents types de recommandations : collaborative, basée sur le contenu, basée sur la connaissance, hybridation entre la recommandation collaborative et celle basée sur la connaissance et l'hybridation entre la recommandation basée sur le contenu et celle basée sur la caonnaissance. Il est à noter que l'hybridation entre la recommandation collaborative et celle basée sur le contenu n'est pas présentée dans le tableau car elle a été utilisée pour les produits simples (Balabanovic & Shoham, 1997). Les colonnes du tableau représentent les deux approches de configuration avec recommandation : la configuration avec recommandation de caractéristiques et la configuration avec recommandation des valeurs des caractéristiques. La configuration utilisant la programmation de contraintes n'est pas présentée en colonne car elle est assimilée à la recommandation basée sur la connaissance.

Tableau 1 : Tableau récapitulatif des types de recommandations utilisées pour la configuration des produits complexes

Configuration avec recommandation / **Type de recommandation**	**Configuration avec recommandation de caractéristiques**	**Configuration avec recommandation des valeurs de caractéristiques**
(1) collaborative	(Herlocker et al., 2004)	(Cöster et al., 2002)
(2) basée sur le contenu		
(3) basée sur la connaissance	(Mazo et al., 2014)	(Felfernig & Burke, 2008) (Tiihonen & Felfernig, 2010)
(4) hybride : (1) + (3)		(Tran & Cohen, 2000) (Burke, 2002)
(5) hybride : (2) + (3)		

La littérature et le tableau montrent que la recommandation collaborative a été largement explorée et que la recommandation basée sur la connaissance est adaptée pour la

configuration des produits complexes. Toutefois, la recommandation basée sur la connaissance n'est, entre autres, que la configuration par la programmation de contraintes dans le contexte des LPs ce qui n'est pas suffisant pour guider l'utilisateur. Il est, alors, intéressant d'utiliser une approche hybride combinant la recommandation basée sur la connaissance avec un autre type de recommandation afin de permettre une configuration dynamique et interactive de la LP et un guidage supplémentaire à l'utilisateur.

Dans ce contexte, cette thèse s'intéresse à combiner la recommandation basée sur la connaissance et la recommandation basée sur le contenu. Le choix de la recommandation basée sur le contenu repose sur le fait que dans le cadre de produits complexes tels que les LPs, il y a suffisamment de connaissances et de descriptions relatives aux produits ce qui facilitera l'utilisation de la recommandation basée sur le contenu. De plus, la recommandation collaborative nécessite un nombre important d'évaluations des utilisateurs pour bien fonctionner. Toutefois, il n'y a pas de garantie de trouver des évaluations pour les LPs qui dépendent du domaine traité (voitures, appartements, etc.) (Felfernig et al., 2007). En se référant à l'étude faite sur les systèmes hybrides par Burke (Burke, 2007), l'approche proposée, Recodyn, effectue une hybridation mixée des deux types de recommandation.

Le chapitre suivant donne un aperçu de Recodyn en présentant les grandes lignes de fonctionnement de la démarche.

CHAPITRE 3: Aperçu de l'approche Recodyn

3.1 Introduction

Ce chapitre présente un aperçu de l'approche de recommandation Recodyn proposée dans cette thèse pour la configuration interactive de systèmes complexes. Le cœur de cette approche réside en un algorithme qui permet de configurer des systèmes dont la complexité est dûe à la variabilité spécifiée au moyen de MLP. L'algorithme génère des recommandations en interagissant avec l'utilisateur qui a la possibilité de spécifier ses exigences au fur et à mesure des étapes du processus de configuration.

3.2 Aperçu de Recodyn

L'algorithme prend comme entrée une spécification du système et des exigences, et génère des recommandations. L'algorithme est itératif, c'est-à-dire que le système final est élaboré progressivement, par une succession d'intéractions avec l'utilisateur. A chaque fois que l'utilisateur exprime une exigence, l'algorithme propose de nouvelles recommandations qui concernent des cacractéristiques du système qui n'ont pas encore été considérées. Le système étant spécifié comme une LP dont les caractéristiques varient en respectant une série de contraintes. Les recommandations concernent, donc, le choix de caractéristiques optionnelles.

A chaque étape, les recommandations générées résident en des configurations partielles. L'idée est donc de focaliser chaque recommandation sur un sous-ensemble de caractéristiques de la LP. A chaque itération du processus de recommandation, seules certaines caractéristiques sont considérées par l'algorithme qui n'en recommande que quelques unes. Le sous-ensemble concerné est appelé « *paquet de caractéristiques* ». Ces paquets de caractéristiques peuvent être soit définis statiquement (donc à l'avance) et fournis en entrée, soit définis dynamiquement au fur et à mesure de la configuration. L'inconvénient de la première stratégie est le risque de s'appuyer sur des paquets de caractéristiques qui pourraient en réalité être incompatibles avec les caractéristiques déjà choisies. La solution à ce problème consiste à définir les paquets de caractéristiques par des experts-métier qui maitrisent les spécifications de la LP et par le processus de recommandation proposée qui ne propose que des choix compatibles. La deuxième stratégie permet de calculer les paquets de caractéristiques en fonction des exigences de l'utilisateur ou à chaque fois que l'utilisateur prend des décisions sur des caractéristiques pendant le processus de configuration. La deuxième stratégie nécessite une mise en œuvre très complexe. D'autre part, la configuration d'une LP respecte toujours les contraintes et

dépendances entre ses caractéristiques ce qui limite les possibilités des choix de paquets. Cette stratégie pourrait donner des résultats qui s'approchent de ceux d'une stratégie statique définissant les paquets de caractéristiques par des experts-métiers. Une stratégie statique est alors moins couteuse qu'une stratégie dynamique.

En plus d'effectuer les recommandations de caractéristiques, l'algorithme se charge de définir les paquets de caractéristiques auxquels les recommandations s'appliquent en garantissant que seules les caractéristiques éligibles sont réunies dans les paquets. Les exigences de l'utilisateur prennent la forme de collections de caractéristiques choisies. Dans sa thèse, Djebbi (Djebbi, 2011) montre que dans le contexte de LP, les exigences peuvent en réalité être spécifiées sous forme de contraintes qui peuvent être beaucoup plus complexes. Le choix de la simplicité effectué ici correspond aux attentes des utilisteurs de systèmes de recommandation, comme cela est montré au chapitre 6. Elles sont envoyées au système de recommandation avec le MLP qui spécifie toutes les combinaisons possibles et interdites de caractéristiques. Le système calcule les recommandations sous la forme d'une liste ordonnée de configurations partielles, de la plus recommandée à la moins recommandée. Seules les caractéristiques incluses dans le paquet considéré à l'entrée de cette étape sont recomandées à l'utilisateur lors d'une étape de configuration, et seules les configurations correctes sont recommandées. L'utilisateur sélectionne une configuration partielle parmi celles proposées dans la liste, et l'adapte s'il le souhaite en fonction de ses exigences supplémentaires. La configuration sélectionnée est donc systématiquement vérifiée par rapport aux contraintes de la LP afin de s'assurer que l'étape suivante commence à partir d'un choix correct. Le processus se répète de la même façon jusqu'à ce que l'utilisateur décide de l'arrêter en sélectionnant une configuration complète. L'utilisateur peut aussi modifier ses choix à tout moment faisant ainsi un retour en arrière. Cette possibilité est prise en compte dans la conception de l'algorithme qui réagit dynamiquement en s'assurant que les nouveaux choix restent compatibles, et en recalculant les recommandations. La Figure 6 présente un aperçu de l'approche Recodyn en spécifiant les entrées et sorties du système. Les entrées sont la configuration partielle de l'utilisateur, les paquets de caractéristiques et le MLP. La sortie est la liste des configurations partielles recommandées.

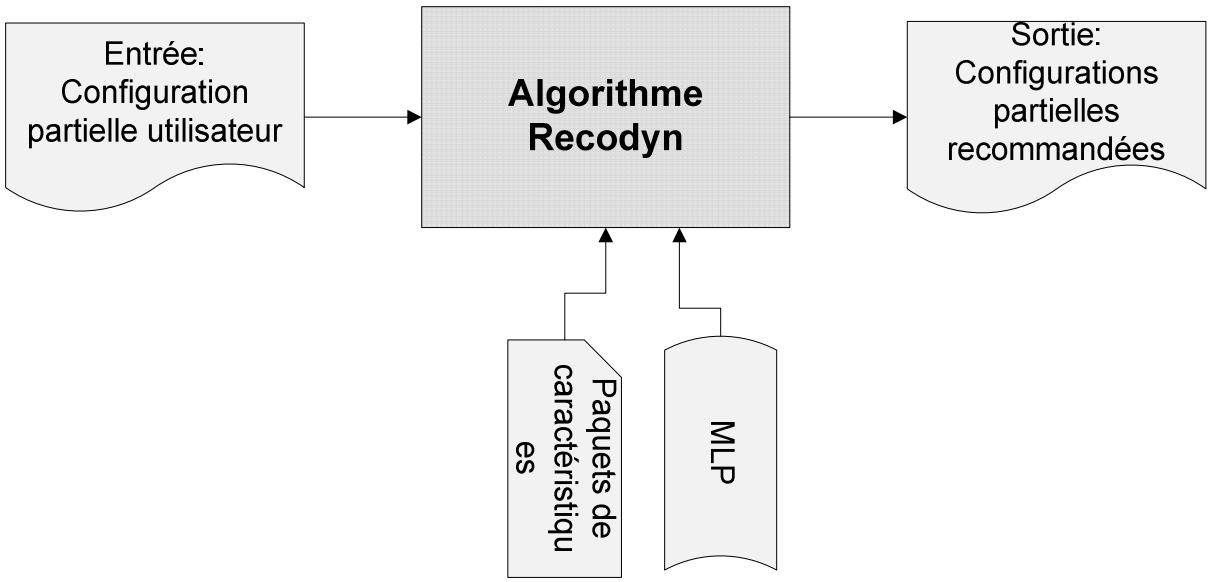


Figure 6 **: L'approche Recodyn**

Utiliser une stratégie de configuration partielle évite les questions soulevées par la configuration à priori. En effet, lorsque les exigences sont spécifiées, il y a des chances pour qu'elles ne soient pas compatibles avec les contraintes de la LP. Par ailleurs, se concentrer sur des paquets de caractéristiques plutôt que sur tout l'ensemble de toutes les caractéristiques permet d'atténuer le risque d'exigences « incorrectes » au sens où elles ne permettraient même pas de configurer un produit en respectant les contraintes de la LP, ce qui impliquerait un retour en arrière préjudiciable. Dans la stratégie de configuration partielle, l'utilisateur n'a pas à se prononcer sur l'ensemble des caractéristiques. Seul un sous-ensemble de caractéristiques est considéré à la fois, puis les décisions sont automatiquement propagées sur les autres caractéristiques via les contraintes. En plus de diminuer le risque de configurations incorrectes, cela permet également de propager les exigences sur le reste des caractéristiques de la LP. En d'autres termes, l'utilisateur a la possibilité de voir directement comment ses décisions induisent des exigences relatives à des caractéristiques sur lesquelles il ne s'est pas encore prononcé explicitement.

À l'extrême, il est possible d'élaborer le produit caractéristique par caractéristique. Ce n'est cependant pas toujours intéressant, en particulier lorsque la spécification de la LP comporte de nombreuses caractéristiques. Malgré tout, cette approche pourrait être employée dans le cas des caractéristiques « abstraites» qui sont pertinentes et déterminantes.

A l'inverse, le paquet pourrait correspondre à l'ensemble de toutes les caractéristiques. Dans ce cas, l'utilisateur doit prendre des décisions sur toutes les carcatéristiques à la fois ce qui entrainerait les risques d'exigences incorretes et leurs conséquences mentionnées précédemment.

3.3 Exemple d'application

L'exemple de la LP Laptop/Notebook Dell choisi pour illustrer l'application de Recodyn provient du référentiel SPLOT[3]. SPLOT (Software Product Lines Online Tools) est une plateforme offrant des outils en ligne pour manipuler les LPs. Il permet de modifier, deboguer, analyser, configurer, partager et télécharger des FMs instantanément.

Le MLP présenté en Figure 7 est spécifié au moyen de la notation FODA (Kang et al., 1990). Le modèle a été choisi pour sa petite taille et sa simplicité et parcequ'il s'agit d'un exemple connu.

FODA (Feature Oriented Domain Analysis) est une notation utilisée pour l'analyse de domaine et axée sur les caractéristiques des sytèmes. L'objectif de FODA est d'identifier les caractéristiques importantes décrivant les aspects communs et variables d'un système à l'autre dans une famille de produits.

[3] http://www.splot-research.org

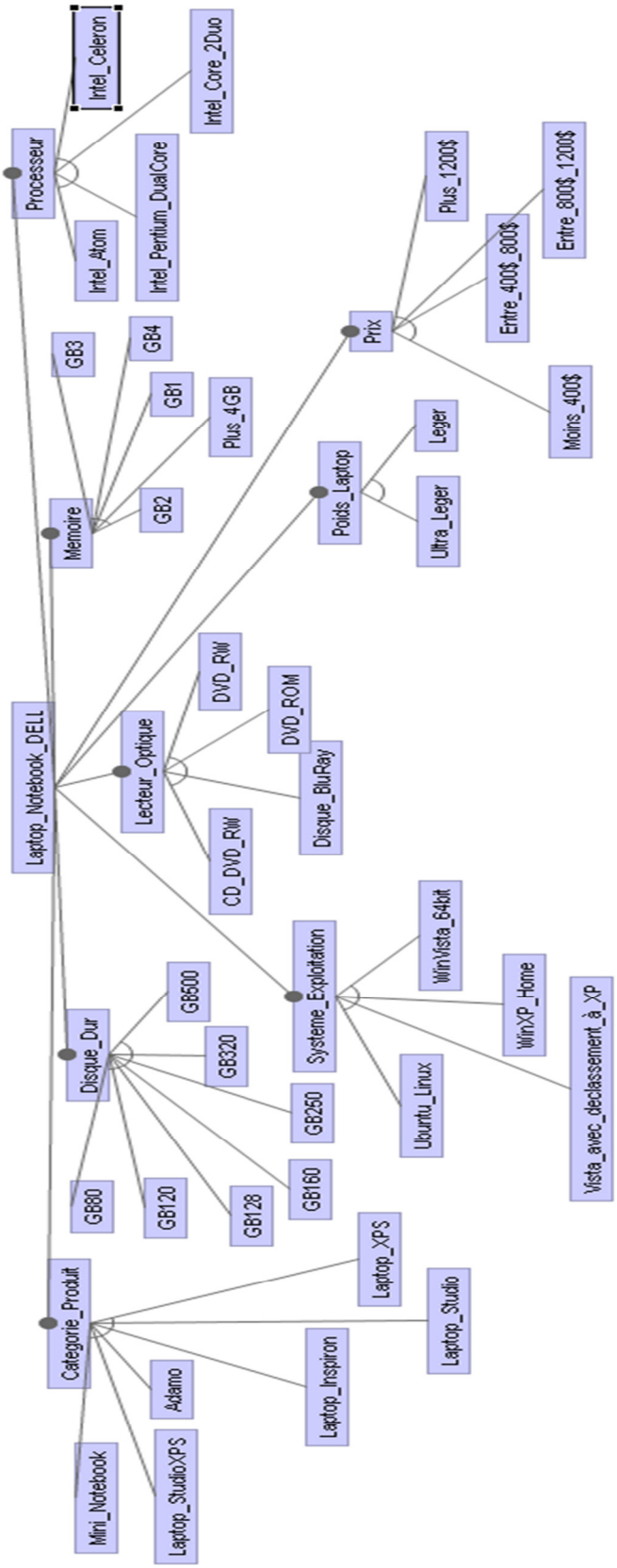


Figure 7: Le modèle de caractéristiques des ordinateurs Laptop/Notebook Dell

Chaque ordinateur Laptop/Notebook Dell possède 8 caractéristiques obligatoires qui sont « *Categorie_Produit* », « *Disque_Dur* »,« *Système_Exploitation* », « *Lecteur_Optique* »,« *Memoire* », « *Processeur* », « *Poids_Laptop* » et « *Prix* ». Chacune de ces caractéristiques obligatoires possède des caractéristiques-filles qui sont liées avec la cardinalité de groupe [1..1]. Par exemple, « *Ultra_Leger* » et « *Leger*» sont liées avec la cardinalité de groupe [1..1], ce qui signifie qu'une seule option parmi les deux peut être sélectionnée dans une configuration d'ordinateur Laptop/Notebook Dell.

La Figure 8 présente un exemple de contraintes de la LP Laptop/Notebook Dell. Toutes les contraintes sont de type « *exclut* ».

$\neg(Mini\_Notebook \wedge Leger)$
$\neg(Mini\_Notebook \wedge Intel\_Core\_2Duo)$
$\neg(Mini\_Notebook \wedge GB128)$
$\neg(Laptop\_Inspiron \wedge Intel\_Atom)$
$\neg(Laptop\_Inspiron \wedge Plus\_4GB)$
$\neg(Laptop\_Inspiron \wedge Plus\_1200\$)$
$\neg(Laptop\_Studio \wedge GB120)$
$\neg(Laptop\_Studio \wedge Intel\_Celeron)$
$\neg(Laptop\_Studio \wedge Plus\_4GB)$
$\neg(Adamo \wedge Intel\_Core\_2Duo)$
$\neg(Adamo \wedge WinXP\_Home)$
$\neg(Adamo \wedge CD\_DVD\_RW)$
$\neg(Laptop\_XPS \wedge GB160)$
$\neg(Laptop\_XPS \wedge DVD\_RW)$
$\neg(Laptop\_XPS \wedge Intel\_Atom)$
$\neg(Laptop\_XPS \wedge Moins\_400\$)$

Figure 8 : Extrait des contraintes de la LP Laptop/Notebook Dell

Par exemple, un ordinateur Laptop/Notebook Dell avec une catégorie de produit « *Mini_Notebook* » ne peut pas avoir un laptop de poids « *Leger* ».

Pour appliquer l'approche Recodyn à la LP Laptop/Notebook Dell, il faut définir les paquets de caractéristiques. Pour des raisons de simplicité et pour les raisons de l'exemple, ceux-ci seront prédéfinis en amont. Les 3 paquets de carcatéristiques suivants sont proposés:

P_1= {Categorie_Produit, Système_Exploitation}

P_2= {Disque_Dur, Lecteur_Optique, Memoire, Processeur}

P_3= {Prix, Poids_Laptop}

Le choix des paquets de caractéristiques est fait en classant les caractéristiques par ordre d'importance d'un utilisateur témoin.

Une fois les paquets de caractéristiques définis, l'utilisateur spécifie ses exigences en sélectionnant une configuration partielle formée par les caractéristique du paquet P_1. Dans ce cas, l'utilisateur spécifie la catégorie du produit et le système d'exploitation souhaités,

par exemple il choisit la catégorie « *Mini_Notebook* » et le système d'exploitation « *Ubuntu_Linux* » . Le système vérifie la consistence de cette configuration : si elle ne respecte pas les contraintes de la LP, il demande à l'utilisateur de la modifier, sinon il calcule les recommandations sur P_2. Les recommandations fournies à l'utilisateur sont des configurations partielles correctes répondant aux exigences spécifiées sur P_1 et respectant les contraintes permettant de combiner les valeurs des caractéristiques considérés dans P_1 et P_2. Par exemple, le système recommande la configuration $C_{2.1}$*={160GB, Disque_BluRay, 2GB, Intel_Atom}* et ne recommande pas la configuration $C_{2.2}$*={160GB, Disque_BluRay, 2GB, Intel_Core_2Duo}* car il y a la contrainte suivante :

$$\neg(Mini\_Notebook \wedge Intel\_Core\_2Duo)$$

L'utilisateur fait son choix sur le paquet P_2 en sélectionnant une recommandation ou en l'adaptant selon ses exigences. Avant de passer à la recommandation sur P3, le système vérifie la consistence de la nouvelle configuration. Par exemple, l'utilisateur choisit de sélectionner la recommandation $C_{2.1}$. La nouvelle configuration de l'utilisateur devient C_u*={Mini_Notebook, Ubuntu_Linux, 160GB, Disque_BluRay, 2GB, Intel_Atom}.* Le système cherche ensuite à calculer des recommandations sur P_3, et l'utilisateur sélectionne la recommandation proposée $C_{3.1}$*={Entre_400$_800$, Ultra_Leger}.* L'utilisateur obtient alors une configuration complète C_u*={Mini_Notebook, Ubuntu_Linux, 160GB, Disque_BluRay, 2GB, Intel_Atom, Entre_400$_800$, Ultra_Leger}.*

Le Tableau 2 présente un récapitulatif des intéractions effectuées entre l'utilisateur et le système Recodyn.

Tableau 2 : Récapitulatif des interactions entre l'utilisateur et Recodyn

Paquets de caractéristiques	Utilisateur	Recodyn
P_1	Spécifier les exigences	
P_1		Vérifier les contraintes
P_1	Modifier les exigences en cas d'incorrection	
P_1		Vérifier les contraintes
P_2		Recommander des configurations partielles
P_2	Sélectionner une configuration partielle	
P_2		Vérifier les contraintes
P_2	Modifier les choix en cas d'incorrection	
P_2		Vérifier les contraintes
P_3		Recommander des configurations partielles
P_3	Sélectionner une configuration partielle	
P_3		Vérifier les contraintes
P_3	Modifier les choix en cas d'incorrection	
P_3		Vérifier les contraintes
P_1, P_2, P_3		Renvoyer la configuration finale complète

3.4 Conclusion

Ce chapitre a présenté un aperçu de l'approche Recodyn et un exemple d'application qui est la LP Laptop/Notebook Dell. Cet exemple sera utilisé plus tard pour démontrer le fonctionnement de l'approche Recodyn. Le processus de configuration et recommandation est décrit de manière plus précise dans les chapitres suivants.

CHAPITRE 4: Méthode de Recherche

4.1 Démarche scientifique

Ce chapitre présente la méthode utilisée pour mener à bien le projet de recherche. Partant d'une collaboration avec l'entreprise Rexel, il a fallu développer un algorithme qui serait utilisé dans le cadre de l'activité commerciale pour la vente de tableaux électriques. S'agissant d'un projet réel, il a été décidé que l'étude serait réalisée à l'aide de la méthode recherche-action (O'Brien, 2001). La recherche-action est un type particulier d'étude de cas (Wieringa, 2014). Sa spécificité est que contrairement à l'étude de cas qui a un rôle d'observation, elle a un rôle d'action. La recherche-action consiste à utiliser un artefact expérimental sur un problème réel afin d'aider un acteur dans un contexte réel, tout en tirant des leçons d'ordre scientifique (Wieringa, 2014).

La conception de la méthode recherche-action mise en œuvre pour cette thèse a été inspirée de (Wieringa, 2014), (Runeson et al., 2012) et (Wohlin et al., 2012).

Pour les études de cas menées selon une approche itérative, Anderson et Runeson (Andersson & Runeson, 2007) ont défini le modèle de processus en spirale présenté en Figure 9. Le modèle montre comment ajuster progressivement l'objectif, le champ d'application des conclusions et définir les responsabilités des chercheurs et celles du personnel de la compagnie concernée de manière itérative. Le modèle de processus en spirale distingue trois phases : la phase exploratoire, la phase confirmative et la phase explicative.

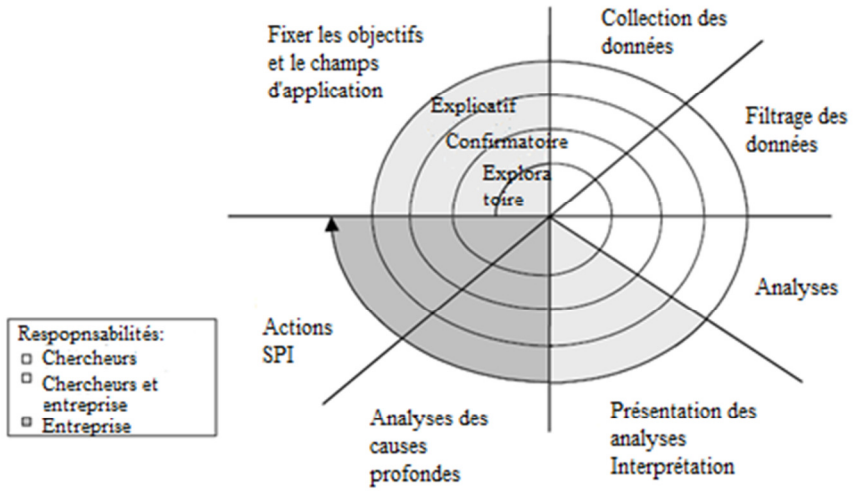


Figure 9 : Le processus d'étude de cas en spirale (Andersson et Runeson, 2007)

Dans le contexte de la méthode recherche-action, le cas tableau électrique a été utilisé à des fins exploratoires en s'inspirant du processus d'étude de cas en spirale. L'étude a été

conduite à la demande de l'entreprise Rexel qui commercialise du matériel électrique et des produits électroniques dans le monde entier aux professionnels de la construction, de l'entretien, de la rénovation et du développement de tous types de bâtiments et d'infrastructures qu'il s'agisse de bâtiments à usage professionnel ou de particuliers. La finalité du projet était de développer un système de recommandation d'un type nouveau à l'intention des services commerciaux de la société. Les actions du département « Recherche & Développement » de Rexel l'avaient amenée à conclure sur l'inadéquation des techniques classiques de recommandation : celles-ci produisaient des préconisations qui en faisaient pas sens de point de vue métier. Le prototype de l'algorithme proposé dans cette thèse ayant attiré l'attention de Rexel, la société a souhaité entamer une collaboration. Afin de tester les concepts, il a été convenu de se concentrer sur les tableaux électriques. En effet, ceux-ci sont à la fois représentatifs des produits vendus par Rexel, tout en étant d'un niveau de complexité suffisant pour démontrer le passage à l'échelle et son aptitude à traiter des situations réelles. Pour ce faire, il a été nécessaire de modéliser la LP tableaux électriques. Le cycle exploratoire nous a aidés à redéfinir l'approche, reformuler les objectifs et résoudre un certain nombre de verrous tels que l'impact du langage de modélisation choisi pour les spécifications sur les fonctions de recommandation.

4.2 La recherche-action

La recherche-action est définie comme « *l'intervention dans une situation sociale afin d'améliorer cette situation et d'apprendre de celle-ci* » (Wieringa & Morali, 2012) (Susman & Evered, 1978). Elle vise à améliorer la pratique en résolvant des problèmes du monde réel et est menée afin d'étudier les phénomènes contemporains dans leur contexte naturel (Koshy et al., 2011). Cette méthode a été choisie pour sa capacité à valider et évaluer notre recherche à partir d'une expérience sur un cas réel et renseigner sur ce qui s'est passé durant l'expérience. Comme nous étions dans une situation où Rexel avait réellement besoin d'aide pour traiter le problème de la spécification et de la configuration d'une LP non-triviale, nous avons trouvé la méthode de recherche-action appropriée au projet.

Susman (Susman, 1983) a développé un modèle de recherche-action détaillé, et décrit les différentes étapes à effectuer dans chaque cycle du processus de recherche-action. Le modèle de recherche-action consiste en un certain nombre de cycles, contenant chacun cinq phases comme l'illustre la Figure 10.

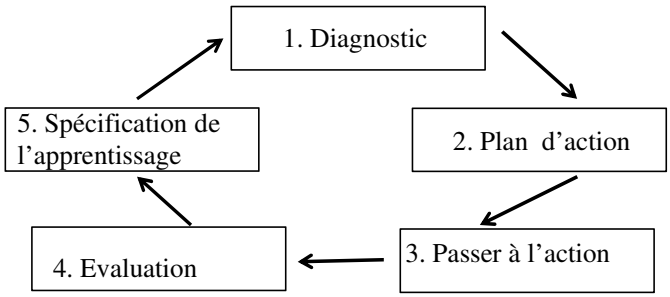


Figure 10 : Modèle de recherche-action (Susman, 1983)

Les cinq phases du cycle de recherche-action peuvent être décrites comme suit :

1. **Diagnostic :** cerner le problème et collecter les données nécessaires pour effectuer un diagnostic détaillé.
2. **Plan d'action :** définir les différentes solutions possibles susceptibles de résoudre le problème défini dans la première étape.
3. **Passer à l'action :** choisir une solution à mettre en œuvre.
4. **Évaluation :** analyser les données correspondant aux résultats du plan d'action choisi.
5. **Spécification de l'apprentissage :** interpréter les résultats de ce plan d'action en apprenant de celui-ci en fonction de la réussite de la solution ou non. Le problème est ensuite réévalué et un nouveau cycle commence. Plusieurs cycles peuvent être établis jusqu'à ce que le problème soit résolu et que tous les intervenants soient satisfaits.

La spécificité de la recherche-action est qu'elle peut partir d'un existant et d'un plan d'action ; et améliorer les pratiques des intervenants grâce à des expériences faites à chaque cycle de la méthode.

4.3 Processus de spécification d'une étude de cas

Cette section apporte les éléments de réponse nécessaires évoqués par les principales étapes du processus de spécification du tableau électrique. Pour ce faire, un rappel des éléments de conception de Runeson et al. (Runeson et al., 2012) est présenté en premier lieu. La conception et planification du tableau électrique sont présentés en second lieu.

4.3.1 La spécification du tableau électrique

L'élaboration des spécifications du tableau électrique a été menée en suivant le processus de recherche d'étude de cas de Runeson et al. (Runeson et al., 2012. Ce processus comprend cinq grandes étapes :

1. **Conception de l'étude de cas** - les objectifs sont définis et l'étude de cas est planifiée.
2. **Préparation de la collecte des données** – les procédures et protocoles de collecte de données sont définis.
3. **Collecte d'éléments de preuve** - les procédures de collecte de données sont exécutées sur le cas étudié.
4. **Analyse des données collectées** –les procédures d'analyse de données sont appliquées aux données.
5. **Rapports** - l'étude et ses conclusions sont consolidées dans les bons formats pour la présentation des rapports.

(Runeson et al., 2012) déclare que la stratégie d'étude de cas est « une stratégie de conception flexible ce qui engendre un nombre important d'itérations au fil des étapes. Par conséquent, les étapes du processus sont très générales et forment seulement un cadre de travail qui ne font que présenter des lignes directrices . »

En outre, (Runeson et al., 2012) définit un ensemble d'éléments qui doivent être pris en compte dans la conception d'une étude de cas. Les quatorze axes de réflexion sont :

1. **Raison.** Pourquoi cette étude est-elle faite ?
2. **Objectif.** Qu'est-ce qui doit être réalisé avec l'étude ?
3. **Le cas.** Dans l'ensemble, qu'est-ce qui est étudié ?
4. **Unités d'analyse.** Plus en détail, qu'est-ce qui est étudié ?
5. **Théorie.** Quel est le cadre de référence théorique ?
6. **Questions de recherche.** Quelles sont les connaissances demandées ou qu'on s'attend à découvrir ?
7. **Propositions.** Quelles relations particulières (causales) sont à étudier ?
8. **Définir les concepts et mesures.** Comment les entités et attributs sont-ils définis et mesurés ?
9. **Méthodes de collecte de données.** Comment les données sont-elles collectées ?

10. **Méthodes d'analyse de données.** Comment les données sont-elles analysées ?
11. **Stratégie de sélection des cas.** Comment les cas (et les unités d'analyses) sont-ils identifiés et sélectionnés ?
12. **Stratégie de sélection des données.** Comment les données seront-elles identifiées et sélectionnées ? Par exemple, qui sera interrogé ?
13. **Stratégie de réplication.** Est-ce que l'étude est destinée à répliquer littéralement une étude antérieure, ou répliquer une étude antérieure de manière théorique ; ou n'y a-t-il pas d'intention de répliquer ?
14. **Assurance de la qualité, la validité et la fiabilité.** Comment les données collectées seront-elles vérifiées pour la qualité ? Comment l'analyse sera-t-elle vérifiée pour la qualité ?

Ces différentes questions sont abordées en détails dans les sections et sous-sections ci-dessous dans le cas du tableau électrique.

4.3.2 Le cas tableau électrique

Afin de spécifier le tableau électrique, les phases de conception et de planification sont élaborées, et la méthode de recherche est décrite.

4.3.2.1 Conception du tableau électrique

1. Raison. Pourquoi cette étude est-elle faite ?

L'entreprise Rexel est un fournisseur de matériel électronique. Elle vend des milliers de produits électroniques en se basant sur des catalogues présentant les produits de manière extensive, c'est à dire un par un. Un ensemble de règles, contraintes et normes régissent les relations et combinaisons des produits électriques. Les clients de Rexel sont des électriciens qui achètent du matériel pour leurs différents chantiers. Chaque client a des habitudes particulières ainsi que des préoccupations logistiques à partir desquelles ses exigences sont définies. Étant donné le large choix de produits, la tâche de recherche et le choix du produit qui correspond aux besoins d'un client est difficile et lente. En fait, le client de Rexel peut se perdre, et perdre du temps en recherchant le produit désiré. En outre, les experts de Rexel ont observé que les clients étaient souvent à la recherche de produits similaires à ce qu'ils ont déjà commandé auparavant. Le département « Recherche & Développement » de Rexel a reconnu qu'il est important de représenter les produits de Rexel dans une représentation unique et intensive des produits, et de définir un processus

adapté pour sélectionner le produit souhaité. Le but était de faciliter la recherche de produits, et proposer à leurs clients uniquement des produits qui soient cohérents avec les projets des clients, conformes aux normes et standards, et respectent les exigences des clients.

Les travaux de recherche effectués durant ce cas ne sont pas simplement de l'observation car les chercheurs ont été impliqués dans la réalisation du cas lui-même. D'où l'utilisation de la méthode recherche-action.

2. Objectif. Qu'est-ce qui doit être réalisé avec l'étude ?

Le projet a été mené avec deux objectifs principaux : tout d'abord, il s'agissait d'introduire un algorithme de recommandation dans le système de vente Rexel (représenté en b.); le second but découle du premier, il s'agissait de trouver une nouvelle représentation pour le tableau électrique (représenté en a.).

a. **Aider à la modélisation du tableau électrique :** Il s'agissait de trouver un moyen de représenter de manière intensive toutes les configurations du tableau électrique, au lieu de les représenter une par une dans un catalogue. Différents problèmes sont considérés dans ce cas tels que les problèmes de compatibilité entre les différentes pièces du tableau électrique et entre les différents constructeurs de pièces, la difficulté liée aux différents types d'habitats, etc. Le problème clef était alors de trouver la représentation la plus appropriée modélisant correctement le tableau électrique avec toutes ses caractéristiques et contraintes.
b. **Guider le processus de recommandation et configuration du tableau électrique :** Le deuxième objectif consiste à configurer des produits, en générant des recommandations, selon les exigences du client et respectant les contraintes de la LP.

3. Le cas et les unités d'analyse. Qu'est-ce qui est étudié ?

Le cas étudié se déroule dans le contexte où le client achète des produits électriques. Ces produits font partie de systèmes complexes au sens où ils sont composés de constituants interfacés mais qui n'ont pas été nécessairement conçus ni produits par la même société et qui présentent eux aussi une structure complexe. Il s'agit d'étudier le produit « *tableau électrique* » utilisé pour tous types d'habitat, tels qu'un appartement ou

une maison et en prenant en compte la norme NF C15-100. Le tableau électrique se compose d'un ensemble d'équipements électriques. Il regroupe les circuits électriques alimentant chaque pièce et les équipements de sécurité et de protection. La Figure 11 présente le modèle du contexte de l'expérience menée avec Rexel, du cas tableau électrique et des unités d'analyse.

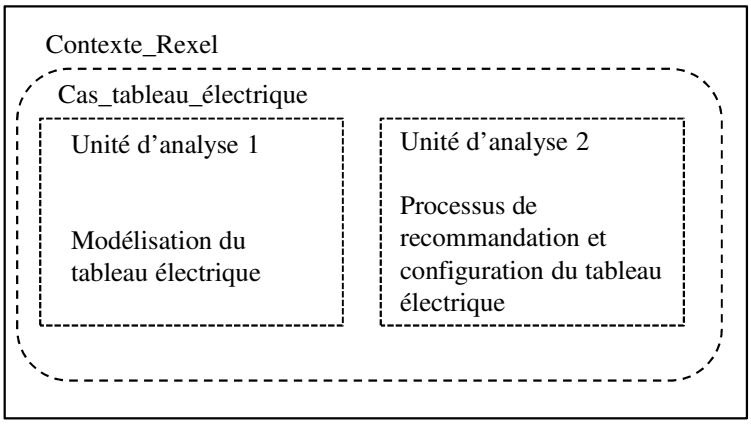


Figure 11 : Modèle du cas unique intégré correspondant au cas Rexel inspiré de (Runeson et al., 2012)

Comme mentionné précédemment, il y a deux objectifs à atteindre qui sont (i) la modélisation du tableau électrique, et (ii) le processus de recommandation et configuration du tableau électrique. Ce cas est de type unique et intégrant deux unités d'analyse.

- **Unité d'analyse 1 : la modélisation du tableau électrique**

La première unité concerne l'activité de modélisation de la LP. En pratique, dans un contexte industriel, on observe souvent que certaines des caractéristiques sont présentes plusieurs fois dans le même produit. Ce phénomène est représenté au moyen du concept de la multi-instanciation. Trois catégories de modélisation ont été élaborées durant ce travail de recherche afin de représenter la multi-instanciation dans une LP. Elles reposent sur: (a) les langages de modélisation sans cardinalités, tels que FODA et OVM, (b) les langages de modélisation où le concept de multi-instanciation est intégré, comme UML et les ontologies, et (c) les langages de modélisation spécifiant des contraintes sur des instances non-distinctes, comme le modèle de caractéristiques avec cardinalités et TVL.

- **Unité d'analyse 2 : le processus de recommandation et configuration du tableau électrique**

La deuxième unité d'analyse concerne le processus de recommandation et configuration au tableau électrique. Ce processus est basé sur la combinaison entre la technique de recommandation et la configuration de LP. Le MLP et la spécification explicite des besoins du client représentent une base de connaissances nécessaire pour configurer un produit. Toutefois, ceci reste insuffisant, car le client cherche aussi des produits qui soient similaires à d'autres qu'il aurait achetés précédemment. Par conséquent, il est essentiel de fournir une solution qui fournisse des recommandations par similarité avec des achats antérieurs répondant aux besoins courants des clients, et respectant les contraintes des LPs.

Suite à l'existence de deux unités d'analyse, la méthode recherche-action doit être appliquée à chacune d'entre elles. Les détails relatifs aux cinq phases appropriées à chacune des unités d'analyse sont présentés ci-dessous.

- ***La méthode recherche-action appliquée à la modélisation du tableau électrique***

Trois cycles d'exploration des trois catégories de modélisation de la multi-instanciation sont réalisés pour effectuer le traitement de validation sur la première unité d'analyse du cas tableau électrique. Chaque cycle correspond à une catégorie de langages. La description des différentes phases du cycle est donnée ci-dessous.

Phase 1 : Diagnostic

Définir le problème clé lors de la modélisation du tableau électrique Rexel qui était lié au concept de multi-instanciation. Une caractéristique est multi-instanciée quand elle est présente dans un produit plusieurs fois. Tout d'abord, il a fallu étudier la LP tableau électrique, puis il s'est agi d'identifier les caractéristiques multi- instanciées et leurs contraintes correspondantes. Cette phase a nécessité plusieurs réunions avec l'équipe de Rexel pour transmettre les connaissances nécessaires à la réalisation du projet.

Phase 2 : Plan d'action

Les langages de spécification des LPs ont été classés en trois catégories permettant de modéliser la multi-instanciation de manières très différentes. Les trois catégories de langages sont : (a) la catégorie de langages sans cardinalités, (b) la catégorie de langages ayant des concepts intégrés pour spécifier la multi-instanciation, et (c) la catégorie de langages de modélisation spécifiant des contraintes sur des ensembles d'instances non-distinctes.

Des exemples de langages de chaque catégorie sont spécifiés. Le choix des trois catégories s'appuie sur une classification des langages existants dans la littérature modélisant la multi-instanciation différemment. Après chaque modélisation du tableau électrique avec les différents langages des trois catégories, les limites et avantages de chacune des catégories de spécification de la multi-instanciation sont explicités.

Phase 3 : Passer à l'action

La première solution appliquée pour modéliser la multi-instanciation est l'utilisation de la catégorie de langages sans cardinalités. Cette catégorie comprend des langages comme FODA et OVM avec lesquels le tableau électrique a été modélisé. Des moyens ont été proposés pour spécifier la multi-instanciation afin de détourner l'absence des cardinalités dans ce type de langages.

Phase 4 : Evaluation

Une évaluation de la première catégorie choisie est faite par tous les intervenants (les chercheurs et les experts de Rexel), pour vérifier que les caractéristiques multi-instanciées et leurs contraintes ont bien été spécifiées.

Phase 5 : Spécification de l'apprentissage

Les résultats de ce cycle sont interprétés en se basant sur les résultats de la phase d'évaluation et en déterminant les avantages et les limites de l'approche de modélisation étudiée afin de spécifier la catégorie la plus adéquate à modéliser la multi-instanciation.

À la fin de ce cycle, le problème est réévalué, et un deuxième cycle démarre en utilisant la deuxième catégorie de langages de modélisation ayant des concepts intégrés pour spécifier la multi-instanciation. Les résultats de ce cycle sont ensuite discutés et le problème est à nouveau réévalué. Enfin, le troisième cycle est lancé pour tester la catégorie des langages de modélisation spécifiant des contraintes sur des ensembles d'instances non-distinctes. Les résultats obtenus à la fin de ce cycle sont évalués à leur tour.

La description des diverses phases et la succession des cycles montre que cette méthode est collaborative. En effet, le processus de recherche et ses objectifs sont effectués en collaboration avec les experts de Rexel. Ceci est une autre caractéristique qui nous a amené à choisir la méthode recherche-action pour mener cette étude.

- ***La méthode recherche-action appliquée au processus de recommandation et configuration du tableau électrique***

Deux cycles ont été effectués pour implémenter l'algorithme de recommandation appliqué au tableau électrique. Les cinq phases de chaque cycle se sont déroulées comme suit :

Phase 1 : Diagnostic

Etude du besoin relatif à la recommandation et l'amélioration du processus de configuration du tableau électrique. Prise de connaissance des nombreuses caractéristiques multi-instanciées et des contraintes complexes de ce dernier.

Phase 2 : Plan d'action

Elaboration d'un algorithme combinant la recommandation basée sur le contenu et la recommandation basée sur la connaissance. Spécifications des exigences lors les deux premières réunions avec l'équipe de Rexel. Détermination des différents niveaux du processus de configuration auxquels la recommandation sera appliquée, ainsi que de la stratégie d'ordre de sélection des caractéristiques.

Phase 3 : Passer à l'action

Application de l'algorithme de recommandation au tableau électrique avec une implémentation spécifique à ses caractéristiques et contraintes. La stratégie choisie pour la sélection des caractéristiques a été en première intention la stratégie statique où les paquets de caractéristiques étaient prédéfinis par les experts de Rexel. De plus, il a été décidé d'appliquer la recommandation sur les valeurs des caractéristiques et leur quantité.

Phase 4 : Evaluation

Evaluation de la première version de l'algorithme par les experts de Rexel et des chercheurs visant à vérifier si l'implémentation de l'algorithme satisfait les exigences de l'entreprise.

Phase 5 : Spécification de l'apprentissage

Interprétation des résultats de ce cycle en déterminant les limites d'une telle implémentation.

À la fin de ce cycle, le problème est réévalué, et un deuxième cycle démarre en implémentant une deuxième version de l'algorithme. Cette seconde version est générique (et non pas spécifique au tableau électrique) qui peut être appliquée à toute LP en considérant le concept de multi-instanciation. En effet, la deuxième version génère des recommandation en prenant en compte les différentes instanciations d'une caractéristique

dans une même configuration. Les résultats de ce cycle sont ensuite discutés et le problème est réévalué à nouveau.

4. ***Questions de recherche. Quelles sont les connaissances demandées ou qu'on s'attend à découvrir ?***

Comme il y a deux objectifs à atteindre à travers le cas du tableau électrique, une série de questions de recherche a été identifiée pour chaque objectif. La première série de questions de recherche concerne la manière de modéliser le tableau électrique. La seconde s'intéresse à la manière d'appliquer la recommandation durant le processus de configuration du tableau électrique.

- **Questions de recherche relatives à la modélisation du tableau électrique**

QR1. Quelle catégorie de langages de modélisation utiliser pour le cas tableau électrique ?

QR2. Quelles informations devraient être prises en considération pour choisir la bonne catégorie de modélisation ?

QR3. Etant donnée la catégorie de modélisation sélectionnée, quel langage de modélisation utiliser pour le tableau électrique ?

- **Questions de recherche relative au guidage du processus de recommandation et de configuration**

QR4. Quelle technique de recommandation utiliser durant le processus de configuration afin de fournir un produit répondant aux besoins du client et adapté à la multi-instanciation ?

QR5. Quels types de données doivent être utilisés pour la recommandation ?

QR6. Quels sont les différents niveaux concernés par la recommandation dans le contexte de configuration des LPs ?

QR7. Quelle stratégie devrait être utilisée pour définir l'ordre dans lequel les caractéristiques sont étudiées ?

4.3.2.2 La planification du tableau électrique

1. ***Méthodes de collecte de données. Comment les données sont-elles collectées ?***

Des groupes de discussion et une visite du site de Rexel ont abouti à une collecte de données et d'informations sur le tableau électrique. Lors de la visite, les experts-métier de la société ont présenté les divers équipements d'un tableau électrique et leurs fonctionnalités. Les groupes de discussion étaient organisés une fois tous les deux mois pendant un an et étaient composés d'experts-métier de Rexel et de chercheurs de l'Université Paris 1 Panthéon-Sorbonne. Les réunions par Skype ont été faites environ une fois par semaine avec le chef de projet pour discuter et faire le point sur les progrès de l'étude.

Au cours des réunions, les experts Rexel ont présenté le fonctionnement d'un tableau électrique, les équipements qu'il contient et leurs rôles, la norme NFC 15-100, les éléments étudiés dans un chantier, les habitudes d'achat des clients, ainsi que les exigences à satisfaire durant cette étude. Ensuite, une discussion concernant les TRs à appliquer durant la configuration du tableau électrique a été menée.

Pour une meilleure compréhension de la norme NFC 15-100, le chef de projet a proposé un document préparé par le fournisseur Schneider Electric. Chaque réunion a été enregistrée (enregistrement sonore) et rapportée sous la forme d'un compte rendu.

La base de données utilisée pour ce cas contient plusieurs configurations du tableau électrique, des profils clients et des données relatives aux achats effectués par un échantillon de clients. Le solveur GNU Prolog a été employé pour générer des configurations supplémentaires, car le nombre de configurations fournies par l'entreprise n'était pas suffisant pour les besoins de l'étude.

2. Méthodes d'analyse de données. Comment les données sont-elles analysées ?

L'analyse des données a été réalisée dans le cadre d'une étude qualitative. Une fois, les groupes de discussion et les réunions par Skype transcrits en comptes rendus, les données collectées ont été classées selon leur type. Il a fallu par exemple distinguer les configurations complètes des incomplètes. Ces données ont été introduites dans la base de données de l'outil développé DynaElec (cf. chapitre 7) pour tester l'approche Recodyn (cf. chapitre 6). DynaElec a permis d'expérimenter l'approche Recodyn sur le cas tableau électrique, d'évaluer ses performances en termes de temps de réponse et a montré la possibilité du passage à l'échelle.

3. Stratégie de sélection du cas

Le cas du tableau électrique a été choisi en concertation avec la société Rexel parce qu'il est à la fois simple que les autres cas de la société, complet, typique du domaine d'activité de Rexel, et des données et expertises étaient disponibles.

Les unités d'analyse 1 et 2 ont été identifiées et sélectionnées lors de la réunion d'équipe. Elles découlent des objectifs spécifiés par l'étude.

4.4 Conclusion

Ce chapitre a présenté la méthode de recherche-action qui a été utilisée pour **valider** l'approche Recodyn appliquée au cas réel tableau électrique utilisé dans un premier temps dans un **but exploratoire**. C'est cette démarche de recherche-action qui nous a conduit à envisager la modélisation de la multi-instanciation, alors que cette problématique n'était pas prévue initialement dans l'approche Recodyn. L'analyse des données et les menaces de validité du cas tableau électrique sont respectivement présentées dans les chapitres 7 et 5. Le prochain chapitre décrit en détails le tableau électrique et présente la modélisation qui en a été faite pour les besoins de la recommandation.

CHAPITRE 5: Modélisation des produits complexes : le cas tableau électrique

5.1 Introduction

La première problématique dans le cas Rexel a été de chercher la meilleure façon de modéliser le tableau électrique, ce qui a notamment posé la question de la représentation du concept de multi-instanciation. Trois catégories de langages de modélisation ont été distinguées. Dans chacune de ces catégories de langages, la multi-instanciation est représentée d'une manière spécifique et qui a un impact sur la technique, l'algorithme, et les résultats de la recommandation. Une comparaison est effectuée entre les trois catégories afin de trouver laquelle est à même de permettre de modéliser « *au mieux* » la multi-instanciation.

5.2 Le concept de la multi-instanciation dans l'ingénierie des LPs

En génie logiciel, l'instanciation est l'un des concepts fondamentaux de la programmation orientée objet. Stefik & Bobrow (Stefik & Bobrow, 1985) définit l'instanciation comme le processus de création d'un nouvel objet dont la structure est définie par une classe. On parle de « multi-instantiation » lorsqu'on utilise plusieurs fois la même classe pour définir des objets ayant les mêmes structures mais avec des spécialisations différentes p.ex. pour offrir des vues différentes du même « phénomène » (Rolland, 1996).

En ingénierie des LPs, on parle de multi-instanciation lorsqu'un artefact (p.ex. caractéristique, variante, etc.) est employé à plusieurs reprises dans la définition d'un produit particulier (Cordy et al., 2013) (Czarnecki et al., 2012). Par exemple, dans le modèle de caractéristiques avec cardinalités, une caractéristique ayant une cardinalité individuelle supérieure à « un » peut être instanciée plusieurs fois (Czarnecki et al., 2005).

5.3 Le cas tableau électrique

5.3.1 Contexte

Avec plus de 2500 références de produits à chaque point de vente, la société Rexel gère la vente à l'aide de catalogues dans lesquels chaque produit est présenté de manière entièrement individuelle. Les produits sont présentés dans le catalogue sans que ne soit mentionné ni leur type d'utilisation ni la conformité de leur utilisation avec la norme (p.ex. la norme NF C15-100). En outre, les relations, telles que les dépendances et les incompatibilités entre les différents produits du catalogue (ou hors catalogue) ne sont pas définis explicitement.

Etant donné le large choix de produits, trouver et choisir le produit qui correspond aux besoins d'un client est fastidieux et lent. Une observation faite par les experts-métier de Rexel est que les clients ont du mal à faire le bon choix parmi les produits du catalogue en un temps raisonnable. Néanmoins, les clients sont généralement à la recherche de produits similaires à ceux qu'ils ont précédemment commandés. Pour toutes ces raisons, Rexel a décidé (i) d'opter pour la représentation de ses produits avec des modèles de domaine permettant une spécification intégrée des dépendances et contraintes et de leur application dans les projet, et (ii) de définir un processus de configuration guidant le choix des produits en fonction du contexte, habitudes du client, etc. L'objectif est de faciliter la recherche des produits et d'aider les utilisateurs (commerciaux, vendeurs, clients) à définir des combinaisons de produits cohérentes et correctes.

5.3.2 Description du tableau électrique

Le tableau électrique est un produit qui peut être présent dans n'importe quel type d'habitat, bâtiment professionnel, hôpital, etc. Il regroupe un ensemble d'éléments électriques permettant la distribution, le contrôle et la protection des circuits électriques alimentant chaque pièce. Il garantit également la sécurité des personnes et de toute l'installation électrique. La configuration d'un tableau électrique est régie par un ensemble de règles et contraintes établies par la norme NFC15-100 que la loi impose de respecter. Cette norme dresse les règles de conception, de construction et de maintenance des installations électriques en France. Bien qu'il existe plusieurs LPs industrielles et réelles chez Rexel, le choix du tableau électrique repose sur le fait qu'il représente une LP complète et moins complexe que l'installation domotique ou les tableaux électriques de sites industriels. La Figure 12 montre un exemple de tableau électrique.

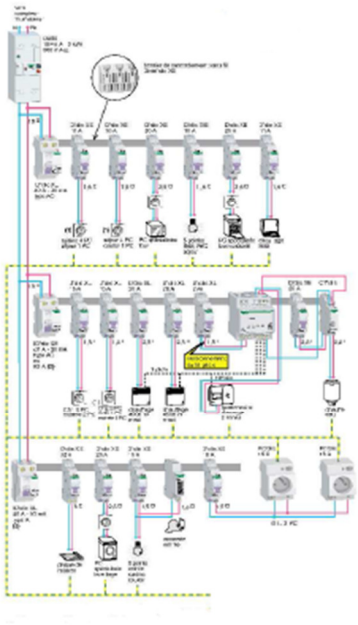


Figure 12 : Exemple d'un tableau électrique[4]

Un logement est composé d'une (ou plusieurs) pièce(s) de type chambre, cuisine, salon, WC, dont les installations électriques sont reliées à un tableau électrique. Ce dernier comporte des composants tels qu'une réhausse, une porte, un peigne vertical, des peignes horizontaux – permettant l'alimentation horizontale des rangées du tableau électrique -, des circuits, différents équipements nécessaires pour la commande d'éclairage, la commande de chauffage et la programmation des circuits. Il est aussi composé d'éléments garantissant la protection des personnes et des biens tels que la parafoudre qui protége les installations électriques contre la surtension dûe à la foudre ; et l'interrupteur différentiel protégeant les personnes contre les risques d'électrocution. Il existe en outre un interrupteur différentiel par peigne horizontal.

Le peigne horizontal alimente également une collection d'éléments électriques de protection tels que les disjoncteurs et les disjoncteurs différentiels. Le rôle du disjoncteur est de protéger le circuit contre un court-circuit, tandis que le disjoncteur différentiel protège à la fois les circuits et les personnes. Chaque peigne horizontal comprend toujours au moins un circuit. Plusieurs types de circuits peuvent être distingués comme le circuit d'éclairage, le circuit de prise PC16A et les circuits spécialisés.

Les circuits spécialisés sont utilisés pour les circuits dédiés aux machine à laver, lave-vaisselle, sécheuse, four, congélateur, ventilation mécanique contrôlée (VMC), plaque de

[4]Le guide de la Norme NF C15-100- Schneider Electric

cuisson, chauffe-eau et volets roulants. Ainsi, il existe un interrupteur différentiel ou un disjoncteur pour chaque peigne horizontal afin de protéger les circuits correspondants.

Le tableau électrique doit tenir compte des contraintes définies par la norme NF C15-100, qui établit des règles pour les différents équipements électriques. Typiquement, une commande d'éclairage inclut un télérupteur ou un contacteur et une minuterie. La commande de chauffage comprend généralement un temporisateur ou un gestionnaire de fil pilote. Pour la programmation de circuits, le tableau contient un interrupteur horaire ou un télévariateur.

Après avoir étudié les problèmes et les exigences de Rexel, nous avons formalisé les deux exigences suivantes en vue de conduire la modélisation de la LP.

- *Exigence5.3.2.1:* Modéliser les produits, présentés un par un dans les catalogues, dans un MLP.
- *Exigence5.3.2.2:* Modéliser les caractéristiques multi-instanciées et les contraintes associées.

La Figure 13 présente les caractéristiques multi-instanciées du tableau électrique.

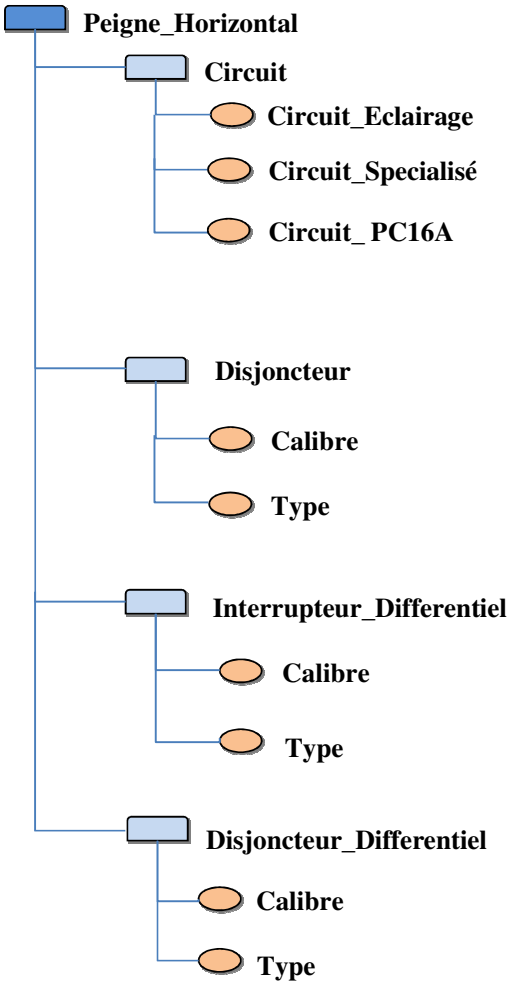


Figure 13 : Les caractéristiques multi-instanciées

Comme le montre la Figure 13, dans un tableau électrique, il y a plusieurs peignes horizontaux. Un seul interrupteur différentiel est branché par peigne horizontal alors que plusieurs interrupteurs différentiels existent par tableau électrique. Sur un peigne horizontal, différents disjoncteurs et disjoncteurs différentiels sont branchés protégeant ainsi plusieurs circuits électriques : circuit spécialisé, circuit d'éclairage ou circuit PC16A. Chacun des équipements branchés sur le peigne horizontal a divers types et calibres. Plusieurs contraintes s'appliquent à ces caractéristiques multi-instanciables qui doivent être respectées.

5.4 Modélisation du tableau électrique

Trois catégories de langages sont proposées pour représenter la multi-instanciation dans les LPs:

1. Une catégorie de langages de modélisation sans cardinalité, comme par exemple le modèle de caractéristiques sans cardinalité (FODA) (Kang et al., 1990) qui représente les caractéristiques d'un domaine particulier et les dépendances qui les relient; ou le modèle de variabilité orthogonal (OVM) (Pohl et al., 2005) qui fournit une vue en coupe transversale de la variabilité relative à tous les artefacts de développement logiciel.
2. Une catégorie de langages ayant des concepts intégrés, tels que les cardinalités, l'héritage et l'instanciation, pour spécifier la multi-instanciation. On compte parmi ces langages les diagrammes de classes UML proposés par Clauss (Clauss, 2001) et les ontologies (Zaid et al., 2008).
3. Une catégorie de langages de modélisation spécifiant des contraintes sur des ensembles d'instances non-distinctes. C'est ce que proposent par exemple le modèle de caractéristiques avec cardinalités (Czarnecki et al., 2005) et le langage textuel de variabilité (TVL) (Classen et al., 2011). Le modèle de caractéristiques avec cardinalités est une extension du langage FODA dans lequel les cardinalités individuelles (pour les caractéristiques) et les cardinalités de groupe (pour les groupes de caractéristiques) peuvent être spécifiés.

Une comparaison des trois catégories de langages de modélisation est effectuée en se basant sur un ensemble de critères : la *simplicité*, l'*efficacité* de la modélisation des caractéristiques multi-instanciées, la *capacité à spécifier des contraintes* complexes sur des caractéristiques multi-instanciées, le *passage à l'échelle* des langages de modélisation des différentes catégories, et le *processus de configuration*.

- La simplicité : Ce critère concerne le degré de simplicité d'utilisation des aspects du langage pour modéliser la multi-instanciation.
- L'efficacité de la modélisation des caractéristiques multi-instanciées : Les trois catégories de langages permettent la modélisation de la multi-instanciation différemment. Toutefois, l'efficacité se traduit par l'aptitude d'un langage à représenter les caractéristiques multi-instanciée dans un MLP de façon optimale tout en respectant les spécifications fonctionnelles du système à modéliser.
- La capacité à spécifier des contraintes complexes sur des caractéristiques multi-instanciées : Les contraintes relatives aux caractéristiques multi-

instanciées sont complexes. La capacité d'un langage est évaluée si elles sont correctement spécifiées par celui-ci.

- Le passage à l'échelle : Dans le contexte de modélisation de la multi-instanciation, on dit qu'un langage permet la modélisation des systèmes à grandes échelles si et seulement s'il permet de modéliser des caractéristiques multi-instanciées à plusieurs niveaux hiérarchiques dans le cas où toute sous-caractéristique est multi-instanciée.
- Le processus de configuration : Certains aspects (structure, interprétation de contraintes, etc.) de langages peuvent être contraignants durant la configuration d'un MLP.

5.4.1 Catégorie des langages de modélisation sans cardinalités

Cette section décrit la première approche de modélisation du concept de multi-instanciation n'utilisant pas les cardinalités. Cette approche offre deux stratégies: (a) dans la première, toutes les instances possibles dans le modèle de LP sont explicitement spécifiées; et (b) dans la deuxième, les attributs sont utilisés pour spécifier le nombre d'occurrences d'une caractéristique.

5.4.1.1 Stratégie de spécification explicite de toutes les instances

Dans cette stratégie, la multi-instanciation est modélisée en spécifiant explicitement toutes les instances possibles d'une caractéristique dans le MLP. La Figure 14 présente le modèle FODA de la LP tableau électrique.

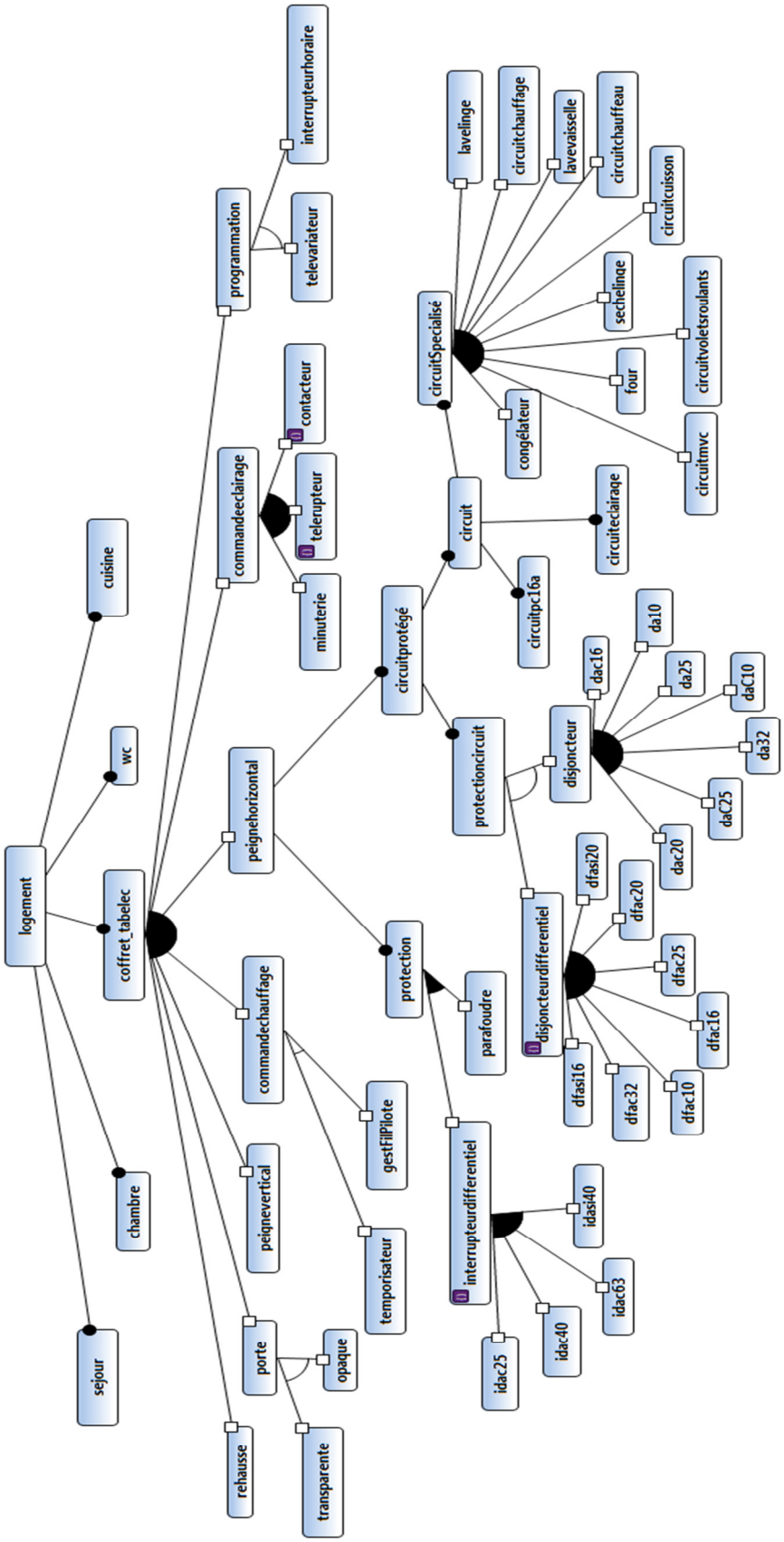


Figure 14 : Modèle FODA du tableau électrique

Le modèle FODA ci-dessus présente tous les éléments qui peuvent être inclus dans le tableau électrique d'un logement, tels que l'« *interrupteurdifférentiel* », le « *disjoncteur* », etc. Ces éléments sont présentés comme des caractéristiques de différents types : *obligatoire*, *optionnel,* groupe-*ou*, groupe-*et*, ou groupe-*alternatif*. Dans le MLP du tableau électrique, certaines caractéristiques sont présentées comme abstraites, d'autres comme concrètes. Par exemple, puisque sur un peigne horizontal, il est possible de brancher un parafoudre, des interrupteurs différentiels, des disjoncteurs, des disjoncteurs différentiels et tous types de circuits tel que le circuit d'éclairage. Le peigne horizontal pourrait être représenté comme la caractéristique-parent, les éléments branchés étant représentés par un groupe appelées caractéristiques-filles. Néanmoins, cette représentation ne respecte pas les relations entre le groupe des caractéristiques-filles. En fait, ces caractéristiques du tableau électrique jouent des rôles si différents qu'elles ne peuvent être regroupées dans un seul groupe de caractéristiques. Par exemple, les circuits d'éclairage et PC 16A sont deux types de circuits alors que les disjoncteurs et disjoncteurs différentiels assurent la protection des circuits. Comme chaque circuit nécessite une protection, les deux groupes de caractéristiques doivent être représentés séparément dans le MLP. Cette situation est traitée en appliquant une technique de refactoring (Thüm et al., 2009) utilisant la caractéristique abstraite comme « *circuitprotégé* ».

Comme l'illustre la Figure 14, une configuration du tableau électrique peut inclure plusieurs exemplaires de *« peigneHorizontal »* contenant chacun des sous-éléments comme « *interrupteurdifférentiel »*. De nombreuses caractéristiques peuvent être ainsi multi-instanciées dans une même configuration du tableau électrique, ce qui complexifie le processus de configuration. La démarche consiste alors à modéliser les caractéristiques multi-instanciables en spécifiant explicitement toutes leurs instances possibles dans le MLP. Par exemple, la caractéristique « *interrupteurdifférentiel* » est définie par un « *calibre* » et un « *type* ». Un interrupteur différentiel peut avoir 3 calibres différents et 3 différents types. Chaque instance d'interrupteur avec un certain type et un certain calibre apparait donc individuellement dans le MLP : un « *interrupteurdifférentiel »* de type « *A* » et de calibre « *40* », un « *interrupteurdifférentiel »* de type « *AC* » et de calibre « *25* », etc.

Chaque instance peut être présente plusieurs fois dans une même configuration, tel est l'exemple de la contrainte spécifiée par Rexel pour le tableau électrique suivante : « *Pour un logement de surface supérieure à 100m2, au moins 3 interrupteurs différentiels de courant assigné 40A et de type AC doivent être prévus* ». Ainsi, un grand nombre d'instances de chaque caractéristique multi-instanciée existent dans le MLP. Par

conséquent, il n'est pas possible de les présenter toutes dans le MLP. Ceci est la première limite du langage FODA dans la prise en compte de la multi-instanciation. La contrainte concernant la caractéristique multi-instanciée « *interrupteurdifférentiel* » peut être spécifiée avec cette stratégie de modélisation, et peut être satisfaite lors de la configuration de la LP. Toutefois, il y a des contraintes qu'il est complexe de spécifier avec cette stratégie. Le Tableau 3 présente quelques exemples de contraintes non spécifiées avec leurs justifications.

Tableau 3 : Exemples de contraintes non spécifiées avec FODA

Contraintes	Justifications
« La dimension du peigne vertical correspond au nombre de rangées à alimenter dans le coffret. »	FODA n'utilise pas les attributs Pour spécifier la dimension du peigne vertical.
« Il est recommandé de protéger les circuits de prises de courant PC 16A et d'éclairage d'une même pièce par des disjoncteurs différentiels différents. »	Spécifier exactement pour quelle instance de pièce (chambre, séjour, salle de bains ou cuisine) la contrainte doit être appliquée est compliqué et porte à confusion.
« Un peigne horizontal permet de connecter de nombreux circuits; chaque circuit est protégé par un disjoncteur ou un disjoncteur différentiel. »	FODA n'utilise pas les cardinalités pour spécifier que plusieurs disjoncteurs et/ou disjoncteurs différentiels sont branchés sur le peigne horizontal.
« Lorsque des circuits de chauffage et de chauffe-eau électriques, dont la somme des puissances est supérieure à 8kVA, sont placés en aval d'un même interrupteur différentiel, il faut remplacer un interrupteur différentiel 40A de type AC par un interrupteur différentiel 63A de type AC. »	FODA n'utilise ni attributs ni cardinalités pour spécifier cette contrainte.

Cependant, cette stratégie permet de modéliser des caractéristiques multi-instanciées à différents niveaux hiérarchiques de l'arborescence comme le montre l'extrait du MLP du tableau électrique de la Figure 15.

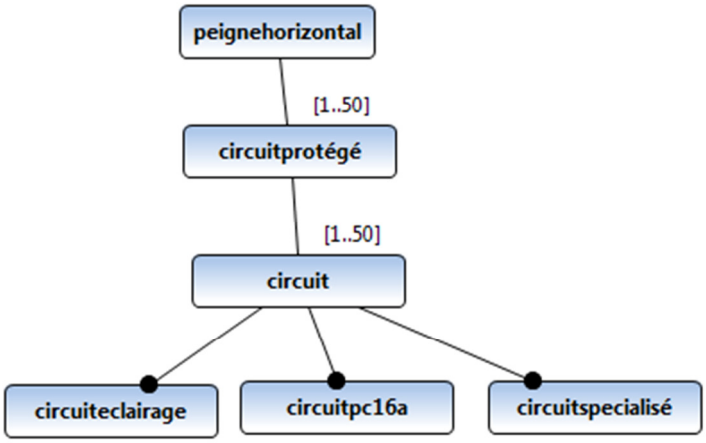


Figure 15 : Le sous arbre peigne horizontal extrait du MLP du tableau électrique

Le « *peignehorizontal*» est une caractéristique multi-instanciée ayant comme caractéristique-fille également multi-instanciée « *circuit* » qui à son tour a des caractéristiques-filles multi-instanciées « *circuiteclairage* », « *circuitpc16a* » et « *circuitspécialisé* ».

Ainsi, la stratégie de modélisation, spécifiant explicitement toutes les instances possibles, permet de résoudre le problème de spécification de la multi-instanciation en partie puisque certaines contraintes ne peuvent être spécifiées. En outre, la structure arborescente est contraignante parce que l'accès direct à une caractéristique n'est pas possible et qu'il faut respecter la hiérarchie. Par exemple dans le modèle FODA, pour sélectionner un « *circuiteclairage* », il faut passer par « *circuit* », puis par « *circuitprotégé* » et enfin par « *peignehorizontal* » (cf. Figure 15). Cette hiérarchie entre les différentes caractéristiques n'est pas prise en compte dans le langage orthogonal OVM.

5.4.1.2 Stratégie utilisant des attributs pour spécifier le nombre d'instances d'une caractéristique

Une autre façon de modéliser la multi-instanciation est d'utiliser des contraintes sur des variables. Les variables sont dans ce cas utilisées pour compter le nombre de fois où une caractéristique est instanciée. Pour ce faire, la variable concernée est spécifiée comme un attribut de la caractéristique (ou variante) dont elle est censée comptabiliser le nombre d'instances. La Figure 16 présente le tableau électrique modélisé avec OVM.

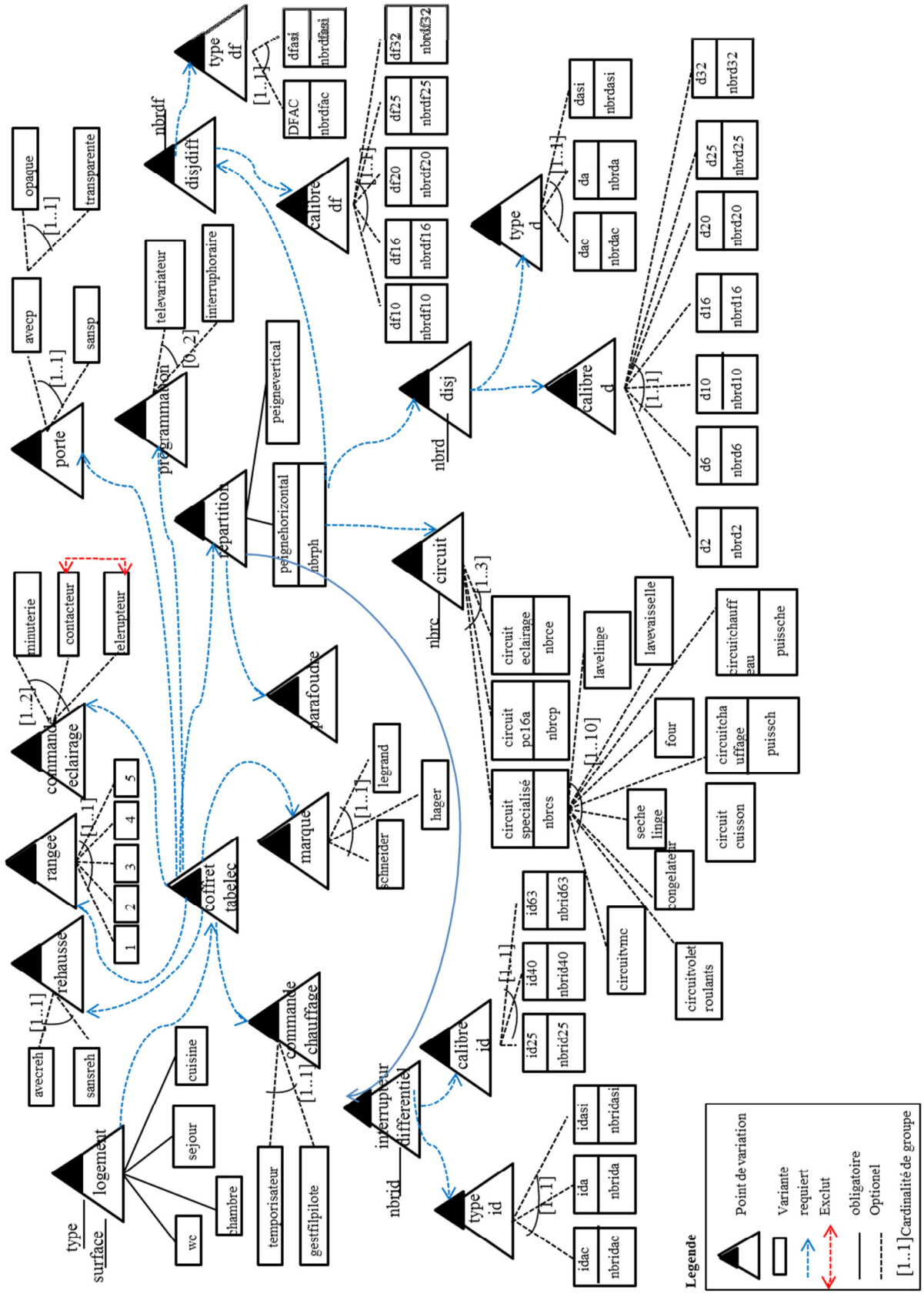


Figure 16 : Modèle OVM du tableau électrique

L'idée est de représenter chaque élément du tableau électrique comme un point de variation qui à son tour offre des variantes. Les variantes représentent les instances possibles de chaque point de variation (Pohl et al., 2005). Par exemple, il peut y avoir de 1 à 5 rangées dans un tableau électrique. Par conséquent, la « *porte* » est représentée par un point de variation, et les différents types de porte sont représentés par des variantes. Puisque les variantes sont optionnelles mais qu'une porte a nécessairement un type, elles sont reliées par la cardinalité de groupe à choix alternatif représentée par la cardinalité *[1..1]*. Tous les éléments du tableau électrique représentés par des points de variation sont reliés par la dépendance « *requiert* ».

Pour chaque point de variation et variante, un attribut est utilisé pour définir le nombre d'instances correspondantes. La Figure 17 présente un extrait du modèle OVM du tableau électrique.

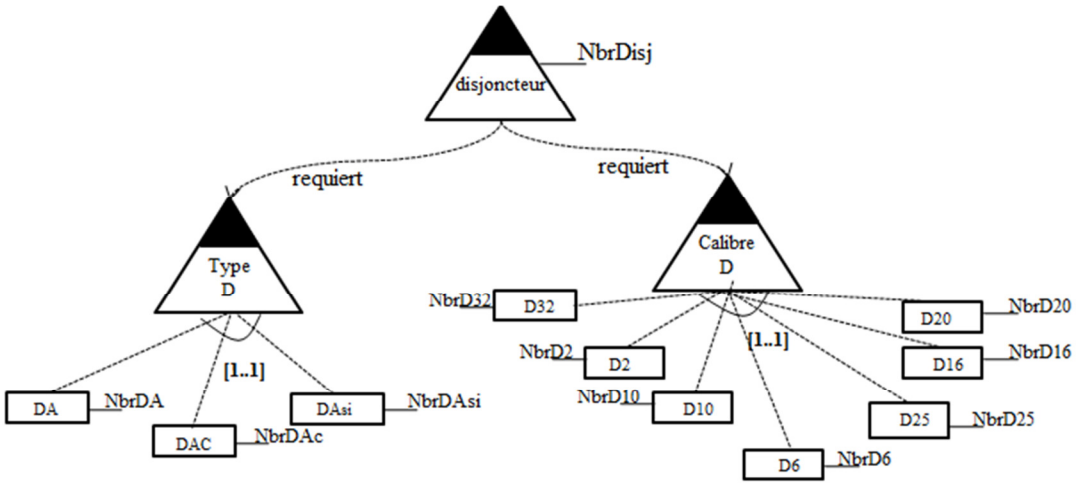


Figure 17 : Extrait du modèle OVM présentant le disjoncteur

Le calibre et le type de disjoncteur sont représentés en tant que points de variation liés au point de variation « *disjoncteur* » par la dépendance « *requiert* ». Cela permet de représenter les deux décisions à prendre lorsqu'on choisit un disjoncteur. Les différentes valeurs respectives du calibre et du type sont représentées comme des variantes porté par chacune d'entre elles. Le nombre d'instances de chaque variante est spécifié par l'attribut « *nbr* ». De même, le nombre total d'instances du disjoncteur est spécifié par l'attribut « *nbrdisj* » assigné au point de variation « *disjoncteur* ». En outre, la cardinalité de groupe *[1..1]* utilisée pour « *typedisj* » et « *calibredisj* » permet de contrôler le nombre de valeurs du type (respectivement calibre) attribué à un disjoncteur.

D'autre part, grâce à sa représentation orthogonale il n'y a pas de hiérarchie entre les caractéristiques de la LP dans le modèle OVM (Pohl et al., 2005). Par conséquent, les caractéristiques multi-instanciées modélisées à différents niveaux hiérarchiques avec FODA peuvent également être modélisées avec OVM. L'attribut « *NbrPh* » définit le nombre d'instances de la variante « *peignehorizontal* » qui requiert un « *circuit* ». Celui-ci possède à son tour un attribut « n*brc* » définissant le nombre de circuits dans une configuration, et possède également des variantes spécifiant toutes ses instances possibles.

Bien que cette stratégie permette la modélisation de la multi-instanciation, il existe encore des limites pour spécifier certaines contraintes sur les caractéristiques multi-instanciées. Par exemple, avec OVM il n'est pas possible de spécifier différentes instances dans une configuration. La contrainte « *Dans le cas particulier des logements de type T1, trois circuits spécialisés au moins sont prévus (un circuit 32A et deux circuits 16A)* » ne peut être spécifiée puisque les calibres « *16* » et « *32* » ne peuvent être sélectionnés dans une même configuration.

5.4.2 Catégorie de langages de modélisation ayant des concepts intégrés pour spécifier la multi-instanciation

La deuxième approche de modélisation qui nous intéresse repose sur un ensemble de langages de modélisation ayant des concepts intégrés pour spécifier la multi-instanciation. C'est par exemple le cas des diagrammes de classes UML Clauss (Clauss, 2001) qui détiennent le principe de multiplicité en utilisant les cardinalités et le principe d'héritage. La Figure 18 représente un extrait simplifié du diagramme de classe du tableau électrique.

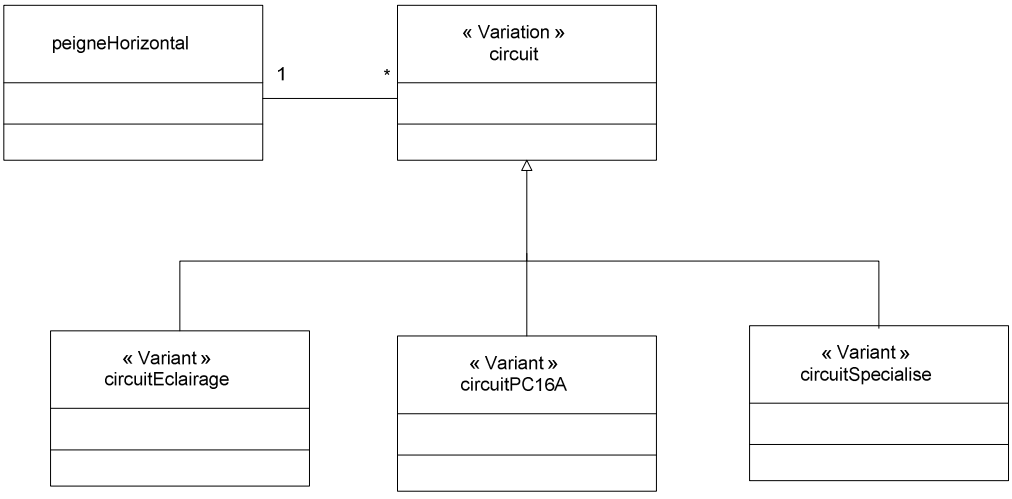


Figure 18 : Extrait du diagramme de classe UML du tableau électrique

Les classes sont modélisées en utilisant des stéréotypes qui spécifient si chaque classe est une « *Variation* », une « *Variant* » ou de type « O*ptionnel* ». Par exemple, la classe « *peigneHorizonatl* » est associée à plusieurs instances de la classe « *circuit* » qui est présentée comme une « V*ariation* ». Un « *circuit* » peut être de différents types : « *circuiteclairage* », « *circuitpc16a* », ou « *circuitspécialisé* ». Ceux-ci sont donc représentés comme des classes « *Variant* » à l'aide du concept d'héritage de diagrammes de classes UML. Les cardinalités sont utilisées pour spécifier combien de fois une classe peut être instanciée dans un modèle de produit. L'est par exemple le cas de la classe « *circuit* » qui peut être instanciée plusieurs fois dans un tableau électrique. En effet, sur un peigne horizontal, plusieurs circuits existent comme les circuits d'éclairage, les circuits PC 16A et les circuits spécialisés.

En outre, un circuit spécialisé peut être de différents types comme a été spécifié par le modèle de caractéristiques avec cardinalités (cf. Figure 19). Toutefois, il n'est pas possible de modéliser les différentes variations de la classe « *circuitspécialisé »* puisqu'une classe de type « *Variation* » ne peut avoir à son tour comme instance des classes du même type. Par conséquent, le diagramme de classes UML ne peut spécifier ni les caractéristiques multi-instanciées à différents niveaux hiérarchiques ni les contraintes qui y sont associées.

5.4.3 Catégorie de langages de modélisation spécifiant des contraintes sur des ensembles d'instances non- distinctes

Cette section présente la troisième approche de modélisation qui consiste à spécifier des contraintes sur des ensembles d'instances non-distinctes telles que le modèle de caractéristiques avec cardinalité, TVL et le langage de programmation par contraintes.

Le modèle de caractéristiques avec cardinalités (Czarnecki et al., 2005) est une évolution du langage FODA qui intègre les concepts de cardinalité individuelle et cardinalité de groupe. La cardinalité individuelle exprime le nombre de fois qu'une caractéristique peut être instanciée. La cardinalité de groupe limite le nombre de caractéristiques à sélectionner. La Figure 19 présente le modèle de caractéristiques avec cardinalités du tableau électrique.

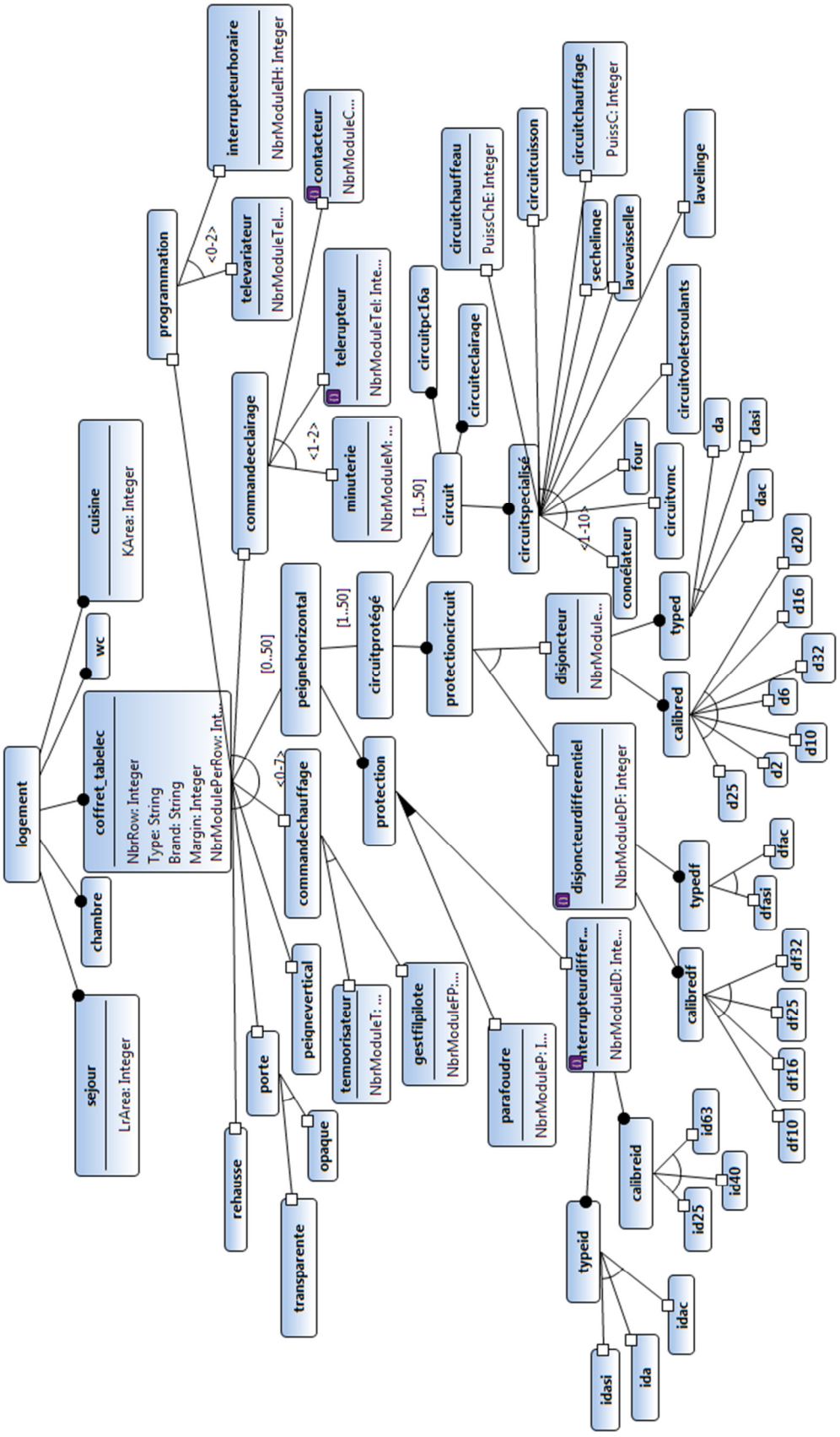


Figure 19 : Le modèle de caractéristiques avec cardinalités du tableau électrique

Les éléments du tableau électrique présentés avec des caractéristiques héritent du concept de refactoring (cf. section 5.4.1.1.) utilisé dans les modèles de caractéristiques (FODA et modèle de caractéristiques avec cardinalités). L'inconvénient de cette représentation réside dans l'absence de distinction entre les caractéristiques concrètes et abstraites. Par exemple, dans la Figure 19 « *circuitprotégé* » est une caractéristique abstraite et « *circuiteclairage* » est une caractéristique concrète. Il est alors difficile de savoir s'il s'agit d'une caractéristique de la LP ou simplement une décision à prendre durant le processus de configuration. Contrairement à FODA, une cardinalité individuelle est assignée à chaque caractéristique multi-instanciée pour définir le nombre d'instances comme la caractéristique « *circuit* » qui a la cardinalité individuelle *[1..*]*.

Comme l'illustre la Figure 19, un tableau électrique peut avoir plusieurs peignes horizontaux. Un peigne horizontal a au moins un « *circuitProtégé* ». Ce dernier a au moins un « *circuit* » de type « *circuiteclairage* », « *circuitpc16a »* ou « *circuitspécialisé* » qui peut être de 10 types. Par conséquent, les caractéristiques multi-instanciées peuvent être modélisées à différents niveaux hiérarchiques du modèle.

Il est à noter que l'utilisation du refactoring dans le tableau électrique a permis de contourner un problème de sémantique qui se pose en spécifiant des contraintes entre les caractéristiques multi-instanciées. Il s'agit du problème de spécification des contraintes croisées entre les caractéristiques multi-instanciées comme le montre la Figure 20. Cette dernière illustre un extrait du tableau électrique modélisé de deux façons différentes : sans et avec refactoring.

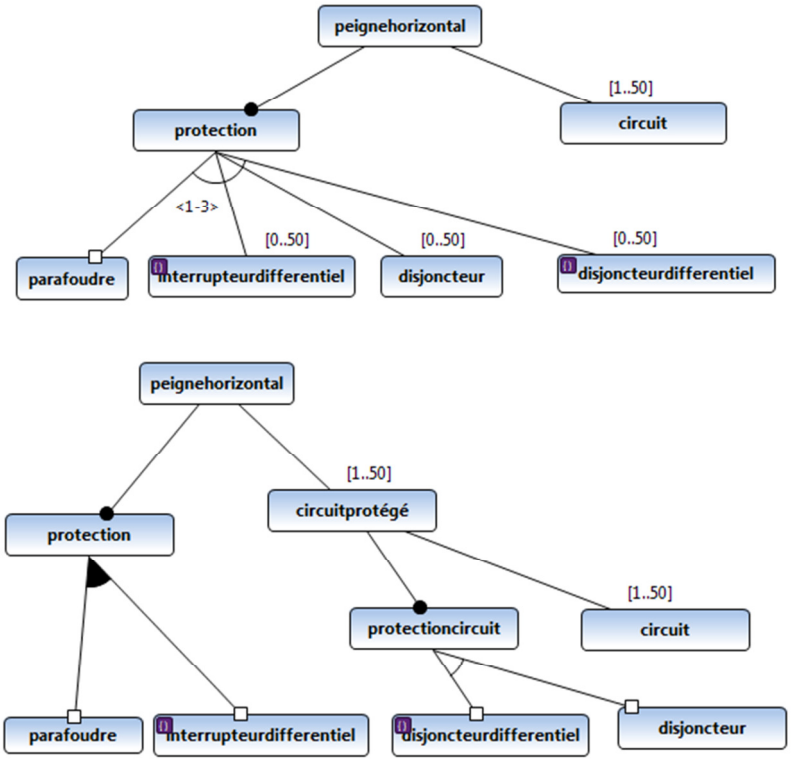


Figure 20 : Extrait du tableau électrique avant et après le refactoring

Dans le premier modèle (sans refactoring), toutes les caractéristiques de protection sont regroupées sous la caractéristique abstraite « *protection* », et une cardinalité individuelle est attribuée à chaque caractéristique-fille multi-instanciée. L'utilisation de la relation « *exclut* » entre le « *disjoncteurDifferentiel* » et l'« *interrupteurDifferentiel*» ne permet pas d'expliciter si cette exclusion est généralisée entre tout « *disjoncteurDifferentiel* » et tout « *interrupteurDifferentiel*».

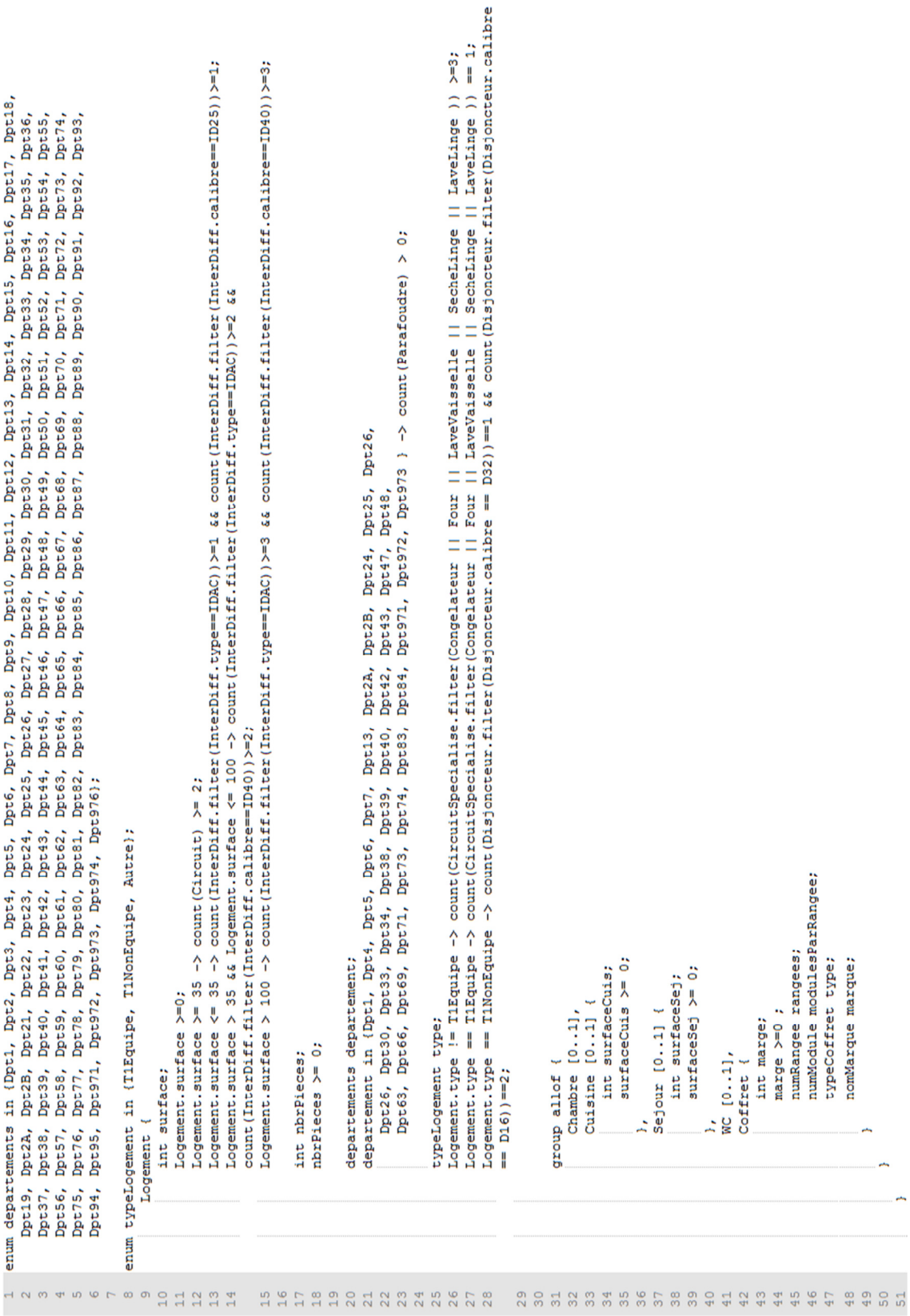
Par ailleurs, l'utilisation du refactoring (deuxième modèle) permet de contourner ce problème. En effet, les éléments de protection sont partitionnés en deux sous-ensembles de caractéristiques : « *circuitProtégé* » (les éléments utilisés pour la protection des circuits) et « *Protection* » (les éléments utilisés pour la protection des personnes et du logement). Le « *circuitProtégé* » comprend un « *circuit* » et une « *protectioncircuit* » qui exige un « *disjoncteur* » ou un « *disjoncteurdifférentiel* ». Par conséquent, il n'est plus nécessaire d'utiliser la cardinalité individuelle.

De nombreuses contraintes appliquées sur des ensembles d'instances telles que « *il doit y avoir au moins 3 circuits spécialisés pour alimenter des appareils de type : lave-linge*, lave-vaisselle, sèche-linge, four et congélateur » peuvent être spécifiées avec le modèle de caractéristiques avec cardinalités. D'autres contraintes appliquées sur des

caractéristiques multi-instanciées sont difficiles à modéliser avec la première approche de modélisation utilisant le langage FODA bien qu'il soit possible de les spécifier avec la deuxième approche utilisant le modèle de caractéristiques avec cardinalités. Par exemple, la contrainte « Le peigne horizontal connecte plusieurs circuits; chaque circuit est protégé par un disjoncteur ou un disjoncteur différentiel » peut être spécifiée avec le modèle de caractéristiques avec cardinalités grâce à la cardinalité individuelle attribuée aux caractéristiques « *circuitprotégé* » et « *circuit* » et pas avec les langages de la première catégories.

Malgré l'utilisation des cardinalités individuelles et groupales, le modèle de caractéristiques avec cardinalités a encore des limites pour spécifier certaines contraintes sur les caractéristiques multi-instanciées. La contrainte « *Lorsque des circuits de chauffage et de chauffe-eau électriques, dont la somme des puissances est supérieure à 8kVA, sont placés en aval d'un même interrupteur différentiel, il faut remplacer un interrupteur différentiel 40A de type AC par un interrupteur différentiel 63A de type AC »* ne peut pas être spécifiée avec le modèle de caractéristiques avec cardinalités en raison de sa structure arborescente qui est parfois restrictive.

Un autre langage appartenant à cette catégorie de modélisation est TVL. Comme le modèle de caractéristiques avec cardinalités, TVL utilise les cardinalités individuelles et groupales pour spécifier les caractéristiques multi-instanciées. Comme son nom l'indique, TVL (Textual Variability Language) est un langage textuel contrairement au modèle de caractéristiques avec cardinalités qui est un langage graphique. Les caractéristiques multi-instanciées sont modélisées avec TVL en utilisant les cardinalités individuelles. Dans TVL, l'énumération est explicite, ce qui permet de définir les attributs et permet d'exprimer des contraintes sur les caractéristiques multi-instanciées (Classen et al., 2011). La Figure 21 présente un extrait du tableau électrique modélisé avec TVL.



```
enum departements in {Dpt1, Dpt2, Dpt3, Dpt4, Dpt5, Dpt6, Dpt7, Dpt8, Dpt9, Dpt10, Dpt11, Dpt12, Dpt13, Dpt14, Dpt15, Dpt16, Dpt17, Dpt18,
  Dpt19, Dpt2A, Dpt2B, Dpt21, Dpt22, Dpt23, Dpt24, Dpt25, Dpt26, Dpt27, Dpt28, Dpt29, Dpt30, Dpt31, Dpt32, Dpt33, Dpt34, Dpt35, Dpt36,
  Dpt37, Dpt38, Dpt39, Dpt40, Dpt41, Dpt42, Dpt43, Dpt44, Dpt45, Dpt46, Dpt47, Dpt48, Dpt49, Dpt50, Dpt51, Dpt52, Dpt53, Dpt54, Dpt55,
  Dpt56, Dpt57, Dpt58, Dpt59, Dpt60, Dpt61, Dpt62, Dpt63, Dpt64, Dpt65, Dpt66, Dpt67, Dpt68, Dpt69, Dpt70, Dpt71, Dpt72, Dpt73, Dpt74,
  Dpt75, Dpt76, Dpt77, Dpt78, Dpt79, Dpt80, Dpt81, Dpt82, Dpt83, Dpt84, Dpt85, Dpt86, Dpt87, Dpt88, Dpt89, Dpt90, Dpt91, Dpt92, Dpt93,
  Dpt94, Dpt95, Dpt971, Dpt972, Dpt973, Dpt974, Dpt976};

enum typeLogement in {T1Equipe, T1NonEquipe, Autre};
    Logement {
      int surface;
      Logement.surface >=0;
      Logement.surface >= 35 -> count(Circuit) >= 2;
      Logement.surface <= 35 -> count(InterDiff.filter(InterDiff.type==IDAC))>=1 && count(InterDiff.filter(InterDiff.calibre==ID25))>=1;
      Logement.surface > 35 && Logement.surface <= 100 -> count(InterDiff.filter(InterDiff.type==IDAC))>=2 &&
      count(InterDiff.filter(InterDiff.calibre==ID40))>=2;
      Logement.surface > 100 -> count(InterDiff.filter(InterDiff.type==IDAC))>=3 && count(InterDiff.filter(InterDiff.calibre==ID40))>=3;

      int nbrPieces;
      nbrPieces >= 0;

      departements departement;
      departement in {Dpt1, Dpt4, Dpt5, Dpt6, Dpt7, Dpt13, Dpt2A, Dpt2B, Dpt24, Dpt25, Dpt26,
        Dpt26, Dpt30, Dpt33, Dpt34, Dpt38, Dpt39, Dpt40, Dpt42, Dpt43, Dpt47, Dpt48,
        Dpt63, Dpt66, Dpt69, Dpt71, Dpt73, Dpt74, Dpt83, Dpt84, Dpt971, Dpt972, Dpt973 } -> count(Parafoudre) > 0;

      typeLogement type;
      Logement.type != T1Equipe -> count(CircuitSpecialise.filter(Congelateur || Four || LaveVaisselle || SecheLinge || LaveLinge )) >=3;
      Logement.type == T1Equipe -> count(CircuitSpecialise.filter(Congelateur || Four || LaveVaisselle || SecheLinge || LaveLinge )) == 1;
      Logement.type == T1NonEquipe -> count(Disjoncteur.filter(Disjoncteur.calibre == D32))==1 && count(Disjoncteur.filter(Disjoncteur.calibre
      == D16))==2;

      group allof {
        Chambre [0..1],
        Cuisine [0..1] {
          int surfaceCuis;
          surfaceCuis >= 0;
        },
        Sejour [0..1] {
          int surfaceSej;
          surfaceSej >= 0;
        },
        WC [0..1],
        Coffret {
          int marge;
          marge >=0 ;
          numRangee rangees;
          numModule modulesParRangee;
          typeCoffret type;
          nomMarque marque;
        }
      }
    }
```

Figure 21 : Extrait du modèle TVL du tableau électrique

Une cardinalité individuelle est attribuée aux caractéristiques multi-instanciées telles que la caractéristique « *chambre* ». Le type d'énumération est utilisé pour prédéfinir toutes les valeurs possibles que peut avoir un attribut, par exemple la caractéristique « *département* » est définie comme un attribut de type énumération. TVL représente les caractéristiques et les contraintes qui s'y appliquent simultanément dans le même corps de définition d'une caractéristique (Boucher et al., 2010) comme par exemple, les contraintes appliquées sur la caractéristique « *surface*». Il existe des fonctions d'agrégation qui peuvent être utilisées avec TVL pour la spécification des contraintes. La fonction « *count* » calcule le nombre de fois qu'une caractéristique est instanciée. Une contrainte telle que « *Pour un logement de surface inférieure ou égale à 35m², au moins un interrupteur différentiel 30mA de courant assigné 25A et de type AC doit être prévu* » peut être alors spécifiée facilement avec TVL comme suit :

logement.*surface* <= 35 -> **count**(InterDiff.*filter*(InterDiff.*type* == *IAC*)) >= 1 && **count**(InterDiff.filter (InterDiff.*calibre* == *I25*)) >= 1;

La Figure 22 présente un autre extrait du modèle TVL du tableau électrique.

```
Protection {
    group [0..2] {
        InterDiff {
            largeur nbrModuleID;
            typeID type;
            calibreID calibre;

            sum(circuitChauffage.map(PuissCh))+sum(circuitChaufEau.map(PuissChE)) > 8000
             -> InterDiff.type == IDAC && InterDiff.calibre == ID63;

            count(Circuit) > count(circuitSpecialise) -> InterDiff.calibre == ID63;
        },
        Parafoudre {
            largeur nbrModuleParafoudre;
        }
    }
},
```

Figure 22 : Exemple de la « Protection » extraite du modèle TVL

La cardinalité de groupe est spécifiée avec la cardinalité *[n..m]*. En outre contrairement au modèle de caractéristiques avec cardinalités, la contrainte « *Lorsque des circuits de chauffage et de chauffe-eau électriques, dont la somme des puissances est supérieure à 8kVA, sont placés en aval d'un même interrupteur différentiel, il faut remplacer un interrupteur différentiel 40A de type AC par un interrupteur différentiel 63A de type AC* » peut être spécifiée avec TVL en utilisant la fonction « *Somme* » comme suit :

SUM (circuitChauffage.map (*puissanceCh*)) + **SUM** (circuitChE.map (*puissanceChE*)) > 8000 →InterDiff.*type* == *IAC* && InterDiff.*calibre* == *I63*;

D'autres contraintes restent difficiles à spécifier avec TVL, comme la contrainte « *Le tableau électrique doit comporter 20% d'emplacements libres* ».

Par conséquent, TVL est une bonne option pour la modélisation des caractéristiques multi-instanciées et les contraintes complexes correspondantes. Néanmoins, les langages graphiques tels que le modèle de caractéristiques avec cardinalités, ont paru plus simple à utiliser que les langages textuels dans le cadre de cette étude. En particulier, la modélisation des caractéristiques multi-instanciées à différents niveaux hiérarchiques avec les modèles de caractéristiques est plus simple que le langage textuel TVL ou le langage de programmation par contraintes. De plus, TVL fonctionne uniquement pour une seule interprétation de la multi-instanciation. Or nous avons été confrontés à des cas d'interprétations multiples. Par exemple, le tableau électrique peut comporter jusqu'à 5 peignes horizontaux. Chacun d'entre eux connecte plusieurs circuits. Cette relation peut être intérprétée de trois façons différentes :

- Pour **une instance** de « *peignehorizontal* », il y a **une instance** de « *circuitprotégé* »
- Pour **un groupe** d'instances de « *peignehorizontal* », il y a **une instance** de « *circuitprotégé* »
- Pour **un groupe** d'instances de « *peignehorizontal* », il y a **un groupe** d'instances de « *circuitprotégé* »

TVL considère uniquement la première interprétation (Cordy et al., 2013) qui ne correspond justement pas à celle attendue pour le tableau électrique.

Par ailleurs, TVL ne permet pas d'exprimer des contraintes temporelles comme la synchronisation ou le caractère séquentiel pour appliquer une contrainte après une autre. TVL ne permet pas la spécification de contraintes telle que « *L'assemblage du tableau électrique nécessite l'identification des interrupteurs différentiels, disjoncteurs différentiels et disjoncteurs avant l'identification des éléments de contrôle tels que le télérupteur, la minuterie, etc.* ».

5.5 Synthèse

Cette section résume la réponse à la question de recherche soulevée au début de ce chapitre :

QR: Quelle approche de modélisation de LP modéliserait « au mieux » le concept de multi-instanciation en tenant compte de toutes les contraintes correspondantes?

Pour répondre à cette question, une comparaison de trois approches différentes a été menée en identifiant les avantages et limites de chacune d'entre elles en se basant sur des critères de comparaison : la simplicité, l'efficacité de la modélisation des caractéristiques multi-instanciées, la capacité à spécifier des contraintes complexes sur des caractéristiques multi-instanciées, le passage à l'échelle, et le processus de configuration.

Pour chaque approche de modélisation de la multi-instanciation, des exemples de langages de modélisation ont été énumérés comme l'illustre le Tableau 4.

Tableau 4 : Récapitulatif des approches de modélisation de la multi-instanciation

Approches		**Avantages**	**Limites**	**Exemples de Langages**
Approche sans cardinalité	Stratégie de spécification explicite des instances	-*Simplicité* : toutes les instances sont explicitement modélisées -*Passage à l'échelle*	-*Efficacité* : grand nombre d'instances sont présentées dans le MLP -*Capacité de spécification des contraintes* : des contraintes assez complexes à spécifier, confusion de spécification des contraintes sur les instances -*Processus de configuration* : Structure arborescente contraignante	FODA OVM DOPLER
	Stratégie d'utilisation d'attributs pour spécifier le nombre d'instances	-*Simplicité* : les attributs sont simples à utiliser, utilisation des cardinalités de groupe -*Passage à l'échelle*	-*Efficacité* : impossibilité d'avoir des instances différentes sur une même configuration -*Capacité de spécification des contraintes* : des contraintes assez complexes à spécifier -*Processus de configuration* : l'aspect orthogonal n'est pas adapté dans ce cas, mais il n'existe pas un problème de hiérarchie	
Approche comprenant des concepts intégrés de multi-instanciation		-*Simplicité* : présenter les caractéristiques multi-instanciées en utilisant les cardinalités et les stéréotypes pour instancier les classes	-*Efficacité* : impossible de spécifier les groupes de caractéristiques : uniquement avec des contraintes OCL. Non spécification de certaines variations - *Capacité de spécification des contraintes* : non spécification de certaines contraintes complexes	UML SysML les Ontologies
Approche spécifiant des contraintes sur des ensembles d'instances non-distinctes	Langages graphiques	-*Simplicité* : les cardinalités individuelles sont simples à utiliser avec un modèle de caractéristiques -*Efficacité* : présenter les caractéristiques multi-instanciées à l'aide des cardinalités individuelles. - *Capacité de spécification des contraintes* : les contraintes sur les caractéristiques multi-instanciées sont spécifiées même si certaines contraintes ne le sont pas. -*Passage à l'échelle*	-*Processus de configuration* : il existe quelques défauts formels comme le problème de contraintes croisées qui posent problème au niveau de la configuration.	Modèle de caractéristiques avec cardinalité
	Langages textuels	-*Efficacité* : présenter les caractéristiques multi-instanciées à l'aide des cardinalités individuelles. Utiliser un type d'énumération explicite pour certains langages comme TVL. -*Capacité de spécification des contraintes* : les contraintes sur les caractéristiques multi-instanciées sont facilement spécifiées (en particulier avec TVL grâce aux fonctions prédéfinies).	-*Simplicité* : la syntaxe est riche et n'est pas simple à utiliser pour les non-experts -*Processus de configuration* : pendant la configuration, TVL fonctionne pour une seule interprétation de la multi-instanciation -*Passage à l'échelle*	TVL Programmation par contraintes (GNU Prolog)

A partir de l'analyse des limites et avantages de chacune des catégories, un arbre de décision a été proposé dans la Figure 23 afin de guider l'utilisateur dans son choix d'approche de modélisation de la multi-instanciation dans une LP.

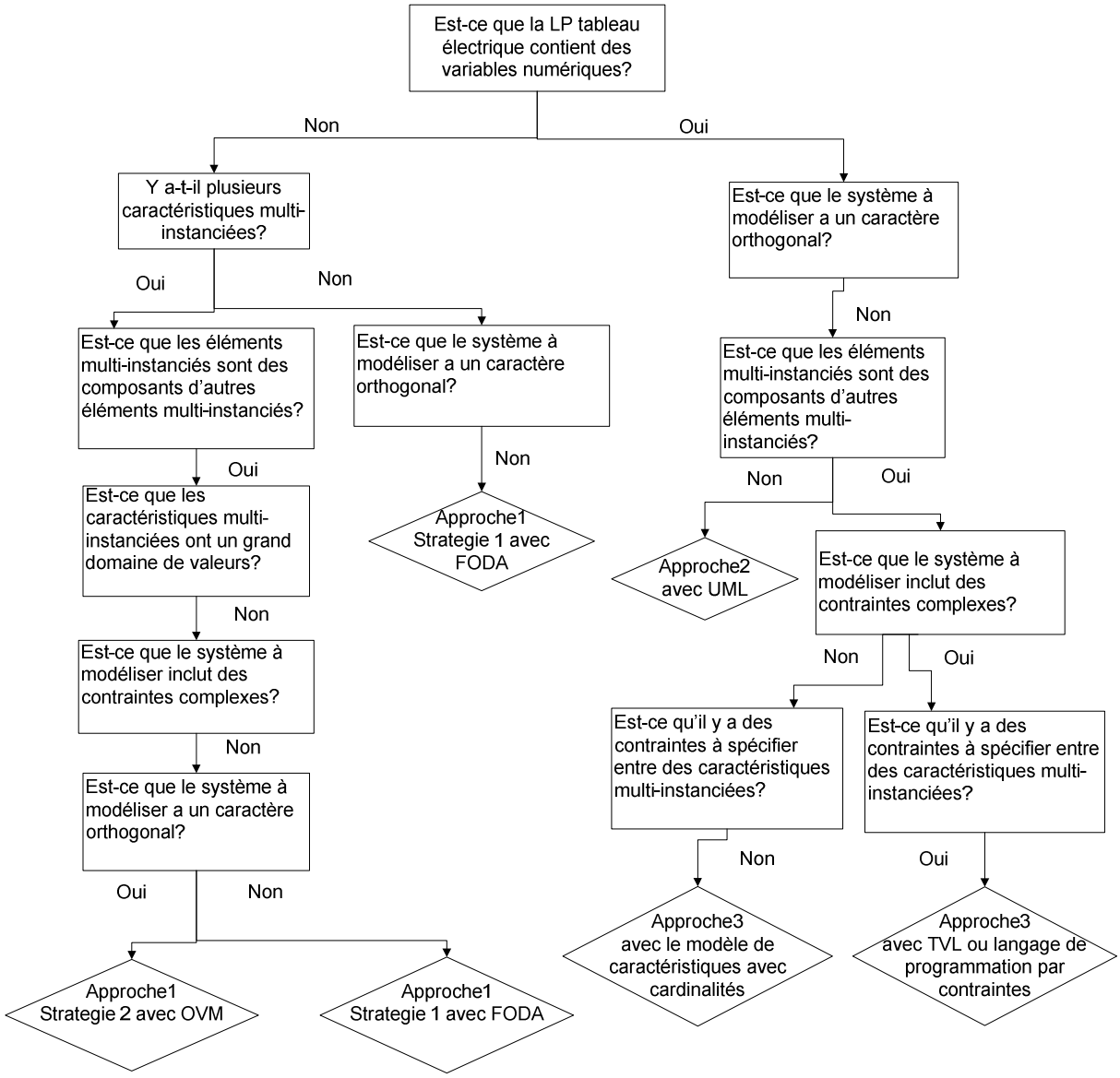


Figure 23 : Arbre de décision pour le choix de l'approche de modélisation

L'arbre de décision est composé d'un ensemble de questions, de réponses de type « *oui* » ou « *non* » et des réponses finales concernant la catégorie de modélisation et le langage correspondant choisis. Les questions définies dans l'arbre de décision concernent l'existence ou non de variables numériques, la présence ou non de cractéristiques multi-instanciées, l'existence ou non de caractéristiques multi-instanciées à différents niveaux hiérarchiques, le domaine de valeurs des caractéristiques multi-instanciées, l'aspect

orthogonal du système à modéliser et la complexité des contraintes et les relations entre elles. Pour mieux expliquer l'utilisation de cet arbre de décision, nous présentons un exemple de son utilisation pour la modélisation du tableau électrique en répondant aux questions définies ci-dessus.

1. *Est-ce que la LP tableau électrique contient des variables numériques ?*
   - Oui, il existe plusieurs variables numériques telles que le nombre de rangées, le nombre de modules nécessaires pour chaque équipement du tableau électrique, et le nombre d'instances de chaque caractéristique. Ainsi, le langage FODA est éliminé.
2. *Est-ce que le système à modéliser a un caractère orthogonal ?*
   - Non, le système n'a pas un aspect orthogonal. En effet, le tableau électrique est décrit d'un seul point de vue à savoir sa structure et son fonctionnement ce qui entraine que la variabilité peut être spécifiée dans un unique modèle. Ainsi, OVM est également écarté.
3. *Est-ce que les éléments multi-instanciés sont des composants d'autres éléments multi-instanciés?*
   - Oui, il y a des caractéristiques multi-instanciées à différents niveaux hiérarchiques, comme la caractéristique « *circuit* » qui est un composant de la caractéristique « *circuitprotégé* » qui à son tour est un composant de la caractéristique « *peignehorizontal* ». Ces trois caractéristiques sont multi-instanciées. Donc, la deuxième approche utilisant le diagramme de classes UML est également écartée.
4. *Est-ce que le système à modéliser inclut des contraintes complexes ?*
   - Oui, la LP tableau électrique contient des contraintes complexes comme les contraintes appliquées sur les caractéristiques multi-instanciées. Donc, la troisième approche est sélectionnée pour modéliser cette LP.
5. *Est-ce qu'il y a des contraintes à spécifier entre des caractéristiques multi-instanciées ?*
   - Oui, il y a une contrainte de ce genre discutée à la section 5.4.3. Ainsi, le modèle de caractéristiques avec cardinalités est éliminé et TVL (ou programmation par contraintes) est sélectionné pour modéliser le tableau électrique.

5.6 Les menaces à la validité

Cette section présente une analyse de la validité de la recherche-action effectuée pour le cas tableau électrique. Cela vise à prouver que les résultats du cas ont une validité suffisante pour la population d'intérêt (Wohlin et al., 2012). En particulier, les experts de Rexel sont la première population d'intérêt; en outre les chercheurs et praticiens sont aussi une population d'intérêt. Les quatre types de menaces à la validité proposées par (Cook & Campbell, 1979) ont été considérées dans le cas tableau électrique.

Ces menaces à la validité sont :

- **La validité de la conclusion** concerne la relation entre le traitement et le résultat et vérifie s'il existe une relation statistique entre eux.
- **La validité interne** concerne la relation entre le traitement et le résultat et vérifie s'il existe un lien de causalité entre eux.
- **La validité conceptuelle** concerne la relation entre la théorie et l'observation et vérifie si le traitement reflète la cause de conception et le résultat reflète l'effet de conception.
- **La validité externe** est la généralisation des résultats de l'étude pour d'autres études en vérifiant la relation causale entre la cause et l'effet de conception et entre le traitement et les résultats.

5.6.1 Validité de la conclusion

Les menaces de validité à la conclusion consistent à tirer les bonnes conclusions sur les résultats d'un traitement appliqué à une expérience (Wohlin et al., 2012). Ces menaces sont dues à la faiblesse des tests statistiques, la faible puissance statistique, ou la violation d'hypothèses des tests statistiques. Toutefois, la méthode à utiliser dans le cas du tableau électrique est de nature qualitative et non quantitative. Ainsi, il n'y a pas de menaces statistiques dans ce cas. La menace de « fishing » pour des résultats spécifiques consiste à influencer les résultats par les chercheurs pour obtenir un résultat particulier (Wohlin et al., 2012). Afin d'éviter cette menace, nous avons demandé à Rexel de préciser les exigences à satisfaire, dès le début du projet avant même l'identification des différentes approches de modélisation. Les exigences de Rexel présentées précédemment dans ce chapitre ont été satisfaites. En effet, leurs produits du tableau électrique présentés dans des catalogues sont

maintenant modélisés avec une LP, et les caractéristiques multi-instanciées avec les contraintes associées y sont effectivement toutes spécifiées.

5.6.2 La validité interne

Les menaces à la validité interne consistent à influencer les résultats de l'expérience par le traitement de l'expérience. Diverses menaces à la validité existent. La validité interne de ce cas concerne les menances historique et de maturation.

Historique : Plusieurs traitements peuvent être appliquées à des moments différents durant une expérience. Par conséquent, les résultats expérimentaux sont affectés par le changement des circonstances (Wohlin et al., 2012). L'expérience avec Rexel a été menée pendant un an et durant la période de travail normale. Le cas du tableau électrique a été réalisé avec un groupe unique. Ainsi, l'exécution de cette expérience évite les périodes telles que les vacances publiques, les fêtes ou les vacances d'été pendant lesquelles l'équipe aurait même momontanément changé de composition.

La maturation : Les participants peuvent modifier leur comportement ou leurs opinions au cours de l'expérience à mesure que le temps passe. Pour contrer cette menace, nous avons élaboré les spécifications du projet en identifiant les objectifs, les règles de fonctionnement et les actions à mettre en oeuvre.

5.6.3 La validité conceptuelle

La validité conceptuelle désigne la généralisation des résultats de l'expérimentation à un concept ou une théorie (Wohlin et al., 2012).

Explication pré-opérationnelle des concepts insuffisante: Cette menace signifie que les concepts de l'expérience ne sont pas suffisamment définis avant d'être traduits en traitements (Wohlin et al., 2012). Afin d'éviter cette menace, nous avons vérifié que tous les concepts de l'expérience sont bien définis et suffisamment clairs en identifiant les questions de recherche du projet et en vérifiant avec les experts-métier de Rexel si elles sont bien spécifiées.

Biais de la mono-méthode: L'utilisation d'un seul type de mesure ou d'observation implique un risque de biais de mesure (Wohlin et al., 2012). Par conséquent, cette expérience inclut deux types d'observations qui peuvent être contre-vérifiées les unes par rapport aux autres. Tout d'abord, nous avons vérifié si toutes les caractéristiques multi-instanciées sont bien présentées dans le MLP et que toutes les contraintes correspondantes

sont bien spécifiées. D'autre part, nous avons utilisé notre outil développé et implémentant l'approche Recodyn avec des caractéristiques multi-instanciées dans la LP.

Généralisation restreinte entre les concepts : Le traitement appliqué durant une expérience peut affecter positivement le concept étudié, mais peut également affecter négativement d'autres concepts involontairement (Wohlin et al., 2012). Afin d'éviter cette menace pour la généralisation des résultats de notre expérience, des concepts pouvant être affectés sont également observés. Par exemple, nous nous sommes intéressés à la meilleure stratégie pour modéliser les caractéristiques multi-instanciées et les contraintes correspondantes pour le cas du tableau électrique. Cependant, la meilleure stratégie de modélisation pourrait avoir un effet secondaire imprévu, comme le problème du passage à l'échelle. Les résultats du traitement appliqué dans notre expérience et leurs effets collatéraux doivent être, par conséquent, pris en compte et analysés.

Attentes de l'expérimentateur : Cette menace concerne les attentes des chercheurs qui ont participé à l'expérience, ce qui pourrait biaiser les résultats. Afin de réduire cette menace, nous avons fait participer des personnes ayant différentes attentes de l'expérience. Les professionnels de Rexel s'attendaient à une représentation unique et intensive de leurs produits pour remplacer les catalogues et faciliter par conséquent le processus de configuration du tableau électrique. Nos attentes étaient de trouver la « meilleure » manière de modéliser le tableau électrique avec ses caractéristiques multi-instanciées en spécifiant les contraintes correspondantes.

5.6.4 Validité externe

La validité externe est la capacité de généraliser les résultats de l'expérience à d'autres situations et à une population plus large que celle de l'expérience elle-même.

Interaction de la sélection et du traitement : Cette menace signifie que la population concernée par l'expériementation n'est pas représentative de la population pour laquelle l'expérimentation sera généralisée. Dans l'expérience avec Rexel, nous avons impliqué les personnes concernées par le cas du tableau électrique tels que les experts-métier de Rexel.

Interaction des réglage et traitement : Cette menace est l'effet de ne pas avoir le cadre expérimental ou les outils expérimentaux nécessaires pour mener l'expérience. Les outils expérimentaux utilisés dans l'expérience avec Rexel sont à jour. Ils ont été choisis par les expérimentateurs et les choix ont été validés par les participants industriels de Rexel en vérifiant leur adaptabilité avec les outils déjà utilisés dans l'entreprise.

5.7 Discussion des résultats

Dans ce chapitre, nous soutenons que la multi-instanciation ne nécessite pas un langage spécifique pour être modélisée. Nous affirmons que la multi-instanciation peut être spécifiée de trois façons différentes à l'aide de langages de modélisation existants.

Trois catégories de langages pour modéliser le concept de la multi-instanciation dans les LPs ont été proposées, et ont été appliquées dans un cas réel. La première stratégie spécifiant des instances explicitement permet de modéliser les caractéristiques multi-instanciées à différents niveaux hiérarchiques. La structure arborescente du langage utilisé fournit une compréhension simple de la navigation de la LP. Néanmoins, l'utilisation d'instances explicites n'est pas pratique, particulièrement quand il s'agit de modéliser plusieurs instances d'une caractéristique comme le peigne horizontal qui alimente plusieurs disjoncteurs différentiels. Spécifier des contraintes sur les instances explicites peut être alors compliqué et peut être source de confusion puisqu'il n'est pas possible de distinguer les instances concernées par la contrainte spécifiée. En l'occurence, la structure arborescente peut être inadaptée pour spécifier certaines contraintes.

La deuxième stratégie, qui utilise des attributs pour calculer le nombre d' instances d'une caractéristique, est (dans l'absolu) une meilleure solution que la stratégie précédente. De plus, la cardinalité groupale permet de calculer le nombre de caractéristiques à sélectionner ce qui n'est pas possible dans la première stratégie. Certaines contraintes, non-spécifiées avec la première stratégie, peuvent être modélisées avec la seconde en utilisant les attributs. Certaines restrictions persistent : par exemple il n'est pas possible de spécifier différentes instances avec différentes valeurs dans la même configuration. Pour conclure, la première approche résout partiellement le problème mais ne permet pas de spécifier toutes les contraintes complexes.

La deuxième approche s'appuie sur des concepts intégrés pour spécifier la multi-instanciation. Il est donc plus facile de modéliser celle-ci avec cette approche. Toutefois, il demeure compliqué de modéliser les caractéristiques multi-instanciées à différents niveaux hiérarchiques.

La troisième catégorie de modélisation permet de spécifier les caractéristiques multi-instanciées et leurs contraintes. Toutefois, des limites demeurent pour spécifier certaines contraintes complexes. Le modèle de caractéristiques avec cardinalités ne peut pas spécifier les contraintes nécessitant des fonctions prédéfinies comme la « *Somme* ». D'autre part, dans le modèle de caractéristiques avec cardinalités, le problème de spécification de

contraintes entre les caractéristiques multi-instanciées est soulevé. Dans certains cas, il est possible de résoudre ce problème en utilisant le refactoring. Par ailleurs, le modèle de caractéristiques avec cardinalités permet de modèliser les caractéristiques multi-instanciées à différents niveaux hiérarchiques. Il n'y a donc pas eu de problème de passage à l'échelle lors de la modélisation graphique.

TVL (ou la programmation par contraintes) n'offre pas la possibilité de modéliser les caractéristiques multi-instanciées à différents niveaux hiérarchiques. Prenons l'exemple d'une LP avec des caractéristiques multi-instanciées. On suppose, d'autre part, que toute sous-caractéristique est multi-instanciée. Dans une telle situation, un problème de passage à l'échelle apparait, ce qui rend impossible la modélisation de cette LP avec un langage textuel. Le problème initial de variabilité réapparait: la variabilité est exponentielle!

D'autre part, TVL n'est pas simple à utiliser : c'est un langage textuel avec une syntaxe riche qui n'est pas facile à maîtriser pour les non experts. Le langage n'offre pas de représentations graphiques qui sont plus faciles à comprendre et à manipuler, à l'instar des modèles de caractéristiques, pour les gens de métier. En outre, TVL ne modélise pas les contraintes temporelles. Par exemple dans les LPs dynamiques, il n'est pas possible d'exprimer les séquences ou les synchronisations.

5.8 Conclusion

Dans ce chapitre, un cas réel a été décrit et modélisé afin de pouvoir y appliquer la méthode Recodyn. Une comparaison a été effectuée entre trois catégories de langages de modélisation. Pour chaque catégorie, un ensemble d'avantages et limites a été fourni. Un arbre de décision a été proposé pour ensuite guider l'utilisateur dans la sélection de l'approche de modélisation adéquate et le langage de modélisation correspondant. Dans le cas du tableau électrique, Rexel a opté pour la troisième approche de modélisation pour sa capacité à spécifier au mieux la LP et a choisi le langage TVL pour le faire. Le chapitre suivant présente l'approche Recodyn en détails.

CHAPITRE 6: L'approche Recodyn

6.1 Introduction

Ce chapitre décrit en détails l'algorithme Recodyn permettant de générer des recommandations sur des configurations partielles d'une LP. La description des étapes de Recodyn est détaillée en second lieu. Ce chapitre décrit la manière d'adapter Recodyn en tenant compte de la multi-instanciation dans une LP. Cette approche est, enfin, appliquée sur l'exemple Ordinateur Laptop/Notebook DELL.

6.2 L'algorithme Recodyn

6.2.1 Description de Recodyn

L'approche Recodyn combine la recommandation et la configuration d'une LP par l'application de la recommandation tout le long du processus de configuration de la LP. Pour ce faire, il est nécessaire de disposer des spécifications de la LP qui seront utilisées pour parcourir ses caractéristiques et vérifier ses contraintes lors du processus de recommandation.

La complexité de Recodyn réside en l'entrelaçage de la recommandation et de la configuration d'une façon dynamique. Cette complexité se traduit par le fait de pouvoir prendre en compte à la volée des décisions prises par l'utilisateur, au fur et à mesure de son avancement, durant la configuration d'un produit. Cet avancement définit à chaque étape un nouvel ensemble de caractéristiques, ce qui redéfinit la base de caractéristiques utilisées pour la recommandation.

L'algorithme Recodyn accompagne l'utilisateur dans ses choix dans le but d'obtenir une configuration complète de son produit dans un nombre réduit d'étapes. Afin de réaliser cet objectif et face à la complexité évoquée ci-dessus, Recodyn se base sur le principe de « *Diviser pour régner* » qui consiste à ramener la résolution d'un problème complexe à plusieurs sous problèmes moins complexes.

Le problème de recommandation lors de la configuration d'un produit est ainsi de re-définir comme une série de problèmes de recommandation sur des sous-ensembles de caractéristiques comme le montre la Figure 24.

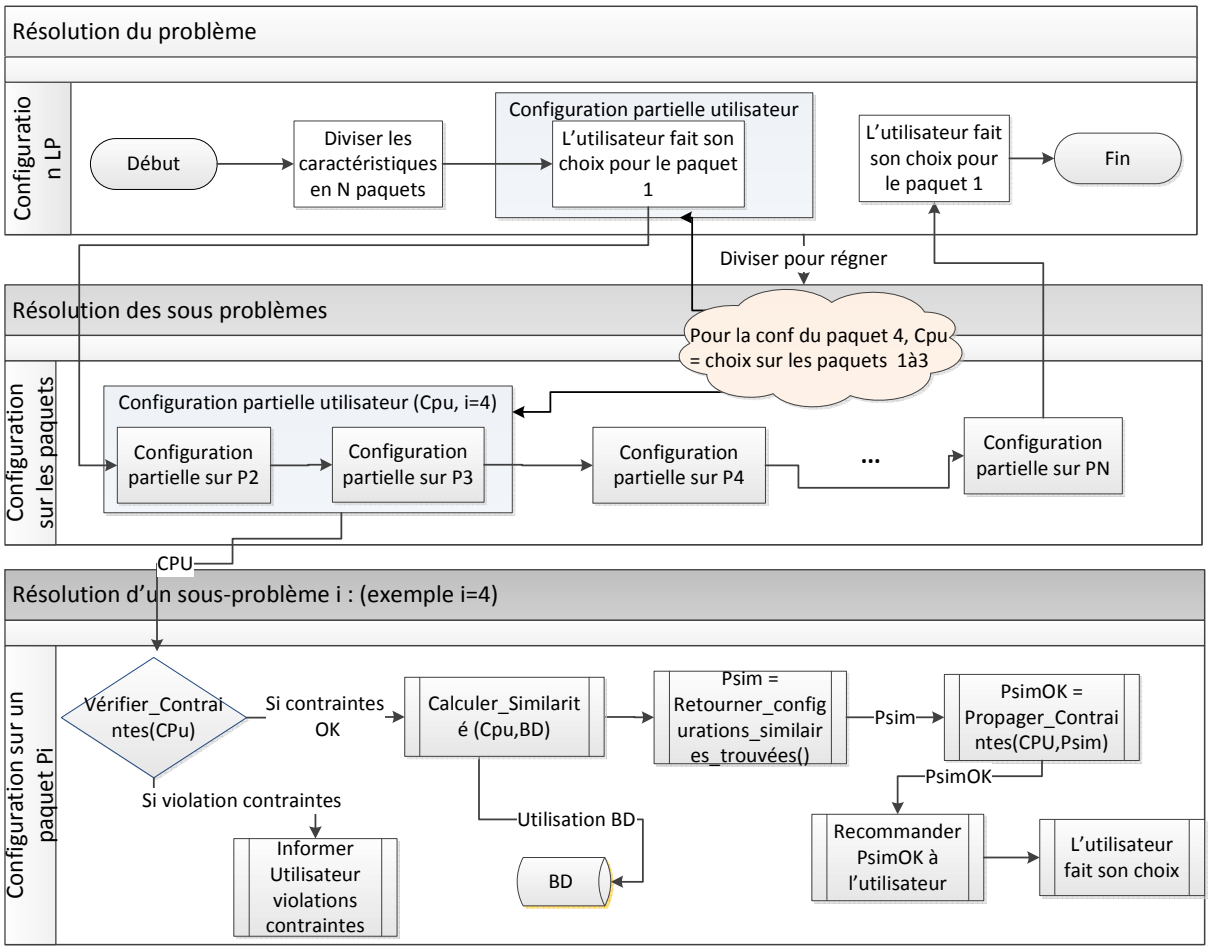


Figure 24 : Processus de Recodyn

Les recommandations offertes par Recodyn pour chaque configuration partielle : (a) satisfait les exigences de l'utilisateur, (b) respecte les contraintes de la LP, et (c) tient compte des décisions prises sur les configurations partielles précédentes. Pour ce faire, l'approche Recodyn combine deux types de recommandation :

1. La recommandation *basée sur la connaissance*, en particulier celle basée sur les contraintes. Ce type de recommandation a été adopté dans Recodyn afin d'assurer le respect des contraintes en recommandant des configurations partielles.
2. La recommandation *basée sur le contenu*. Celle-ci est utilisée pour recommander une configuration partielle satisfaisant les exigences définies dans le profil utilisateur et similaires à des configurations dans la base de données (BD). La technique de recommandation kNN est utilisée pour le calcul de similarité. Afin que la recommandation basée sur le contenu soit

efficace, il faut disposer d'une description riche des produits. Dans le contexte de l'approche Recodyn, les connaissances sur les produits sont fournies par les spécifications (caractéristiques et contraintes) de la LP.

L'algorithme Recodyn est présenté ci-dessous de manière macroscopique.

Algorithme 1 : Algorithme Recodyn

1 : EP=Diviser les caractéristiques de la LP en N paquets
2 : CPu = L'utilisateur fait son choix sur le paquet P_1

3 : ***Pour*** *i=2 à N* ***faire***
4 : Vérifier_Contraintes (CPu)
5 : Calculer_similarité (CPu)
6 : Psim = Retourner les configurations similaires trouvées
7 : Psimok = Propager_contraintes (CPu, Psim)
8 : Proposer à l'utilisateur l'ensemble Psimok recommandé
9 : CPi = Choix de l'utilisateur sur le paquet P_i
10 : CPu = CPu + CPi
11 : ***Fin Pour***
12 : Vérifier_Contraintes (CPu)
13 : CPfinale = CPu

Avec :

- *CPu* : la configuration courante de l'utilisateur.
- *Psim* : les configurations en adéquation avec la configuration de l'utilisateur.
- *Psimok* : les configurations en adéquation avec la configuration de l'utilisateur et respectant les contraintes.
- *CPfinale* : configuration finale

La description détaillée des étapes de l'approche Recodyn est la suivante :

Etape 1 : Composition des paquets de caractéristiques

Les caractéristiques de la LP sont divisées en *N* paquets de caractéristiques calculés de manière statique. A chaque itération de l'algorithme, le processus de recommandation porte sur un seul paquet de caractéristiques. En effet, les paquets de caractéristiques peuvent être présélectionnés par l'utilisateur. Ce dernier peut être un client qui achète un produit et classe les caractéristiques en paquets selon ses préférences, ou un expert-métier qui classe les caractéristiques en paquets selon les différentes stratégies entreprises de gestion de

produit ou de vente. Une des stratégies consiste à classer les paquets de caractéristiques selon les habitudes de configuration des clients de la société. Une autre stratégie est de se concentrer sur les éléments de la LP ayant le moins de variabilité étant donné qu'il y a plus de chance de trouver des solutions correctes lorsqu'on se concentre sur les éléments les moins variables que lorsqu'on se concentre sur les variables ayant plus de valeurs possibles. Un autre exemple de stratégie consiste à exploiter les dépendances entre les décisions et regroupe les caractéristiques dans les paquets qui ont le moins de dépendances, de sorte que la prise de décision sur les caractéristiques d'un seul paquet se propagera aussi peu que possible sur les caractéristiques dans d'autres paquets.

Etape 2 : Spécification des exigences de l'utilisateur

L'utilisateur spécifie explicitement ses exigences sous la forme d'une configuration partielle en attribuant des valeurs aux caractéristiques souhaitées de l'un des paquets définis dans la première étape. La Figure 25 présente le profil-utilisateur.

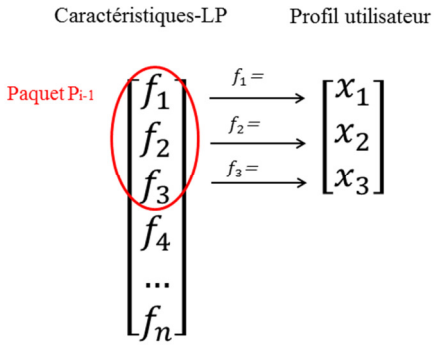


Figure 25 : Représentation de l'espace vectoriel du profil-utilisateur

Le profil-utilisateur est présenté comme un vecteur de valeurs des caractéristiques du paquet concerné.

Etape 3: Recommandation de configurations partielles sur un paquet

L'objectif de cette étape est de rechercher l'intersection entre les configurations partielles recommandées, celles qui sont correctes, et celles qui satisfont les exigences de l'utilisateur spécifiées et vérifiées. La recommandation est alors utilisée pour aider l'utilisateur à prendre des décisions sur le prochain paquet de caractéristiques. Le processus s'effectue de manière incrémentale (paquet par paquet) sur l'ensemble des paquets de caractéristiques, les sous-étapes ne sont cependant détaillées que pour une seule itération.

a. Vérification des contraintes

Une fois que l'utilisateur a déterminé ses exigences relativement à un paquet de caractéristiques donné (itération précédente), l'algorithme vérifie leur cohérence avec les contraintes de la LP. La vérification s'établit pour les valeurs de caractéristiques du paquet considéré et les valeurs déjà attribuées aux caractéristiques des paquets précédents.

Prenons l'exemple de la contrainte: $si\ f_1 = x_1\ alors\ (f_2 = x_2)\ ou\ (f_2 = y_2)$,

Lorsqu'il attribue la valeur x_1 à f_1, l'utilisateur doit aussi attribuer à f_2 la valeur x_2 ou y_2, sinon une notification lui sera envoyée pour l'informer de l'incompatibilité des valeurs de caractéristiques et de rectifier ses exigences.

b. Calcul de similarité

Après avoir vérifié la cohérence de la configuration partielle précédente spécifiée par l'utilisateur, une recommandation basée sur le contenu est utilisée pour comparer les exigences déjà spécifiées avec celles de la BD (par exemple, des anciennes configurations générées dans d'autres contextes ou spécifiées par les experts-métier). Pour ce faire, une mesure de similarité est appliquée sur les paquets de caractéristiques précédents le paquet actuel. La mesure de similarité est calculée en utilisant la TR des kNN qui utilise une représentation vectorielle du profil-utilisateur et des configurations (cf. Figure 26). Les configurations les plus similaires à la configuration partielle de l'utilisateur (CPu) sont retenues. Le calcul de similarité est détaillé plus tard dans la prochaine sous-section.

c. Traitement des résultats du calcul de similarité

Le calcul de similarité de l'étape précédente permet de trouver les configurations en adéquation avec les exigences de l'utilisateur. Cette étape permet d'extraire à partir de ces résultats les valeurs des caractéristiques correspondant au paquet courant afin de les recommander comme l'illustre la Figure 26.

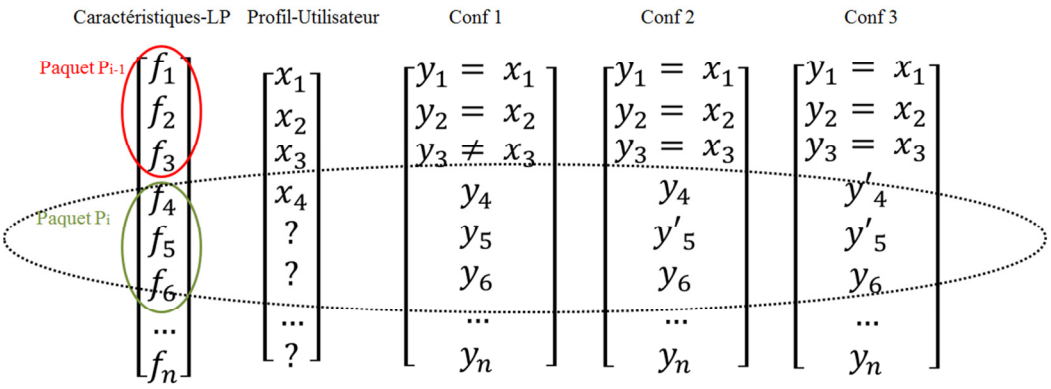


Figure 26 : Représentation de l'espace vectoriel du profil utilisateur et des configurations enregistrées dans la base de données

Supposons que le résultat de la comparaison au paquet P_{i-1} entre les exigences de l'utilisateur et les configurations partielles de la BD fournit les configurations *Conf1*, *Conf2* et *Conf3* comme étant les plus adéquates au profil utilisateur. Au paquet P_i, les configurations partielles $CP_1 = \{y_4, y_5, y_6\}$, $CP_2 = \{y_4, y'_5, y_6\}$ et $CP_3 = \{y'_4, y'_5, y_6\}$ pourraient être alors recommandées à l'utilisateur.

d. Propagation des contraintes

La propagation des contraintes consiste, d'une part, à vérifier la cohérence des configurations partielles sur un paquet de caractéristiques P_i avec les contraintes de la LP et d'autre part à propager les contraintes sur les paquets P_{i+1} à partir des décisions prises précédemment et en utilisant les spécifications de la LP.

Ainsi, avant de recommander à l'utilisateur les configurations partielles trouvées dans l'étape précédente, une propagation de contraintes est nécessaire entre les exigences spécifiées dans le profil utilisateur et les configurations partielles résultantes de l'étape précédente de calcul de similarité. Les configurations partielles résultantes qui ne respectent pas les contraintes de la LP sont éliminées. En effet, certaines caractéristiques sont explicitement sélectionnées ou éliminées suite aux choix précédents de l'utilisateur.

e. Recommandation des configurations partielles

Des configurations partielles correctes, retenues de l'étape précèdente, sont recommandées à l'utilisateur dans l'ordre décroissant de similarité.

f. Spécification des choix de l'utilisateur

L'utilisateur fait son choix en sélectionnant, parmi la liste de configurations partielles recommandées, celle qui lui correspond. Le choix ne porte que sur le paquet de caractéristiques courant.

g. Combinaison des choix pour le paquet courant avec la configuration partielle de l'utilisateur

Le choix effectué par l'utilisateur sur le paquet courant est enregistré et ajouté à la configuration partielle de l'utilisateur.

Etape 4: Propagation des contraintes

Suite au choix de l'utilisateur sur le dernier paquet, une vérification de contraintes est réalisée sur la configuration dans son ensemble. En cas de violation de contraintes, l'utilisateur est invité à réctifier ses choix.

Etape 5: Retour de la configuration finale

Cette étape permet de renvoyer la configuration finale complète du produit après avoir fini de dérouler toutes les itérations de l'étape 3.

6.2.2 Calcul de similarité

Durant le processus de configuration, la taille de la configuration partielle de l'utilisateur change selon son avancement. Le calcul de similarité se base sur l'ensemble de caractéristiques représentant cet avancement. Comme le profil utilisateur et les configurations sont représentées dans un espace vectoriel, nous avons trouvé que la mesure de similarité du cosinus est adécquate pour comparer la configuration partielle de l'utilisateur à celles de la base de données en se basant sur le même paquet. L'algorithme implémentant kNN avec la formule du Cosinus est présenté ci-dessous.

Algorithme 2 : algorithme implémentant la TR kNN

```
1 : ucs = 0
2 : pour i = 1 à k faire                    //k = nombre de caractéristiques
3 :     a1 = 0
4 :    a2 = 0
5 :    pour i2= 1 à k faire
6 :          a1 = a1+au_Carré (poidsCarac(J2))
7 :         a2 = a2+au_Carré(poidsCarac(J2))
```

8 : *fin pour*
9 : w_{ic}= *poidsCarac (i)*
10 : s_1= *racine_Carrée (a1)*
11 : w_{is}=*poidsCarac (i, conf)* //conf = configuration concernées
12 : s_2 = *racine_Carrée (a2)*
13: *ucs = ucs + ((w_{ic} /s_1) * (w_{is} /s_2))*
14 : *fin pour*
15 : *résultat = ucs*

Le calcul de similarité de la méthode kNN se base sur la comparaison des valeurs de caractéristiques mono-instanciées.

Cependant, le cas du tableau électrique a soulevé le problème de modélisation de la multi-instanciation dans une LP ce qui a permis de revisiter l'utilisation de la TR kNN. Le choix de la catégorie de modélisation de la multi-instanciation a un effet sur la manière d'utiliser la TR kNN et par conséquent impacte sur le résultat de l'algorithme Recodyn.

Déclinaison de l'algorithme pour la première catégorie de langages de modélisation :

a. *Première stratégie*

Dans ce cas de modélisation où les instances d'une caractéristique multi-instanciée sont explicitement spécifiées, le calcul de similarité est réalisé de manière classique sur chaque instance de caractéristique tout en comparant entre deux valeurs (celle spécifiée dans le profil-utilisateur et celle spécifiée dans une configuration partielle de la BD) comme le montre la Figure 27.

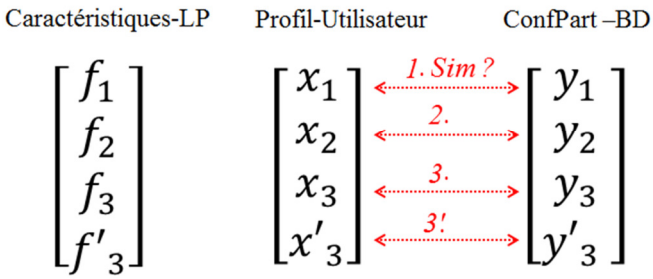


Figure 27 : Utilisation de la TR kNN de manière classique

La caractéristique f_3 est une caracétristique multi-instanciée. Chaque instance est représentée comme une caractéristique différente d'où la notation f_3'. Le calcul de similarité pour une caractéristique multi-instanciée est alors effectué de la même manière que celui de la caractéristique mono-instanciée en comparant valeur par valeur.

b. Deuxième stratégie

L'attribut « *Nbr* », spécifique à chaque variante ou point de variation, utilisé pour compter le nombre d'instances d'une caractéristique est considéré dans ce contexte comme une caractéristique. Dans ce cas, la similarité est alors calculée entre les valeurs de ces caractéristiques. Par conséquent, uniquement le nombre d'instances est recommandé.

Déclinaison de l'algorithme pour la deuxième et la troisième catégorie de langages de modélisation **:**

Comme ces deux catégories de langages permettent de spécifier les caractéristiques multi-instanciables, le calcul de similarité est établi entre deux ensembles de valeurs et non entre deux valeurs. Pour ce faire, la TR kNN est utilisée de manière imbriquée pour les ensembles de valeurs des caractéristiques multi-instanciées comme le montre la Figure 28.

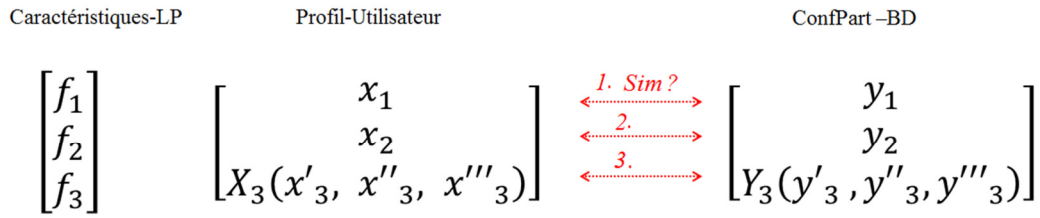


Figure 28 : Utilisation de la TR kNN de manière récursive

La similarité entre les valeurs x_1 et y_1 de f_1 se calcule de manière classique. La similarité entre les valeurs x_3 et y_3 de la caractéristique multi-instanciée f_3 se calcule de manière classique sur leurs sous-ensembles de valeurs respectifs (x'_1, x''_2, x'''_3) et (y'_1, y''_2, y'''_3). Par conséquent, Recodyn recommande implicitement les différentes valeurs d'instances d'une caractéristique avec leurs nombres.

6.3 Application de l'approche sur l'exemple Ordinateur Laptop/Notebook DELL

Cette section décrit les différentes étapes de l'approche Recodyn appliquée sur l'exemple des ordinateurs DELL. Les détails des étapes du processus de recommandation et configuration sont décrits ci-dessous :

Etape 1 : Composition des paquets de caractéristiques

Les caractéristiques de la LP sont divisées en 3 paquets P_1, P_2 et P_3 (cf. Chapitre 3) suivant une méthode statique.

Etape 2 : Spécification des exigences de l'utilisateur sur P1

En entrée de Recodyn, l'utilisateur spécifie la configuration partielle appelée U_P sélectionnée à partir de P_1 comme le montre la Figure 29 :

$U_P = \{Catégorie\_Produit = « Mini\_Notebook », Système\_Exploitation = « Ubuntu\_Linux »\}$

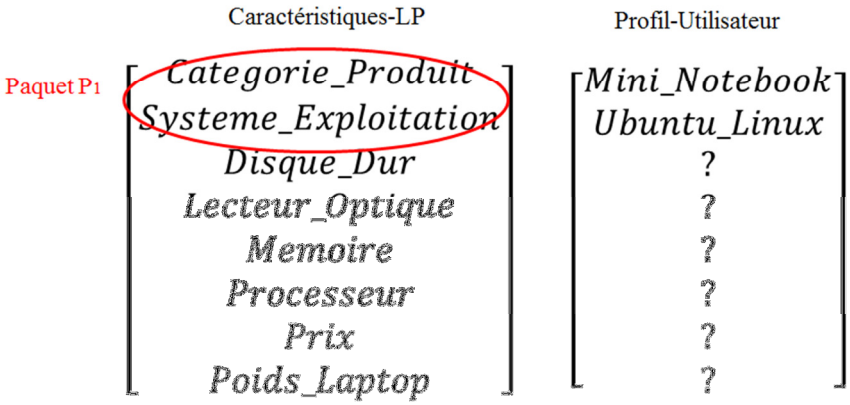


Figure 29 : Spécification des exigences de l'utilisateur sur P_1

Etape 3 **:** *Recommandation de configurations partielles sur P2*

a. *Vérification des contraintes*

Les exigences spécifiées par l'utilisateur sont cohérentes avec les contraintes de la LP.

b. *Calcul de similarité*

La comparaison entre les valeurs des caractéristiques du paquet P_1 est effectuée selon l'utilisation classique de la TR kNN. Les configurations partielles les plus similaires à U_P sont *C1* et *C2* (cf. Figure 32).

c. *Traitement des résultats du calcul de similarité*

Les configurations partielles sur P_2 sont extraites à partir de *C1* et *C2* comme le montre la Figure 30. Elles sont respectivement appelées *C1.1* et *C2.2.*

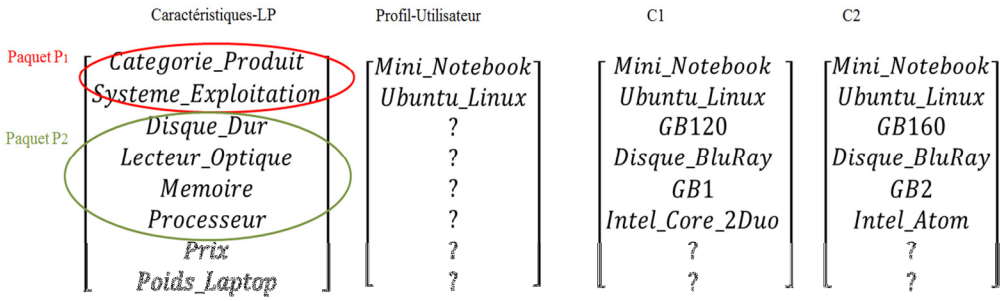


Figure 30 : Traitement des résultats du calcul de similarité sur P_2

d. Propagation des contraintes

Les contraintes sont propagées sur le paquet P_2. En effet, la configuration partielle *C2.1* est cohérente avec les contraintes de la LP, ce qui n'est pas le cas pour *C1* qui ne respecte pas la contrainte: $\neg(Mini\_Notebook \wedge Intel\_Core\_2Duo)$

La configuration *C1* est alors éliminée.

e. Recommandation des configurations partielles

La configuration partielle *C2.1* est recommandée à l'utilisateur.

f. Spécification des choix de l'utilisateur sur P_2

L'utilisateur fait son choix sur P2 soit en suivant la recommandation de Recodyn par la sélection de *C2.1* ou non. On suppose dans ce cas que l'utilisateur a choisi de sélectionner *C2.1*.

g. Combinaison des deux configurations partielles sur les paquets P_1 et P_2

Etape 4 : Recommandation de configurations partielles sur P3

a. Vérification des contraintes

La configuration partielle de l'utilisateur UP est maintenant composée des exigences spécifiées par l'utilisateur sur P1 et P2. Comme il a sélectionné C2.1 sur P2 alors UP est correcte.

b. Calcul de similarité

La comparaison entre les valeurs des caractéristiques du paquet P_3 est effectuée selon l'utilisation classique de la TR kNN. La configuration partielle *C3* est la plus similaire à U_P (cf. Figure 33).

c. Traitement des résultats du calcul de similarité sur P_3

La Figure 31 montre l'extraction de la configuration partielle sur P_3 à partir de *C3*. Elle est notée *C3.1*.

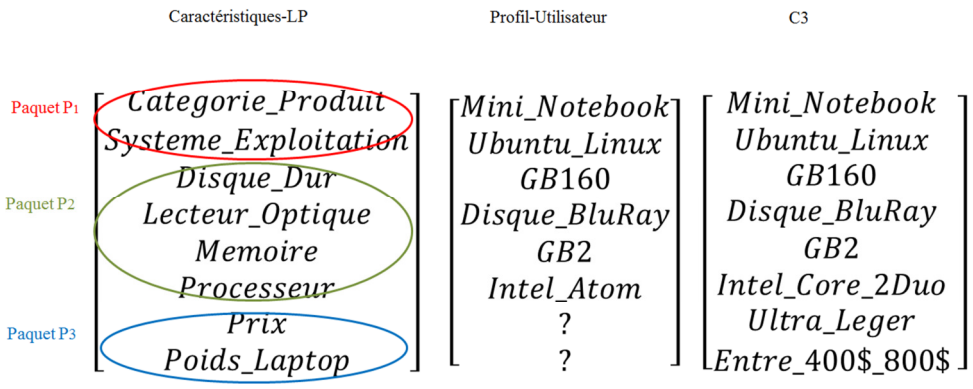


Figure 31 : Traitement des résultats du calcul de similarité sur P_3

d. Propagation des contraintes

Les contraintes sont propagées sur le paquet P_3. *C3.1* est cohérente avec les contraintes de la LP et est alors retenue.

e. Recommandation des configurations partielles

C3.1 est recommandée à l'utilisateur.

f. Spécification des choix de l'utilisateur sur P_3

L'utilisateur choisit de sélectionner *C3.1*.

g. Combinaison des deux configurations partielles U_p et C3.1

Etape 5 : Propagation des contraintes

Les itérations sur les paquets sont terminées. U_p est maintenant une configuration complète. Les contraintes sont propagées sur P_3. Comme l'utilisateur a choisi une recommandation de Recodyn alors U_p est correcte.

Etape 6 : La configuration finale Up est retournée à l'utilisateur

6.4 Conclusion

Ce chapitre a présenté l’approche Recodyn. Il s’agit, en effet, d’un algorithme utilisant la recommandation basée sur la connaissance et celle basée sur le contenu. Recodyn recommande dynamiquement des configurations partielles en raisonnant sur des paquets de caractéristiques. Cet approche est générique : elle peut être utilisée pour toute LP et prend en compte le concept de la multi-instanciation. L'évaluation de Recodyn est étudiée dans le prochain chapitre.

CHAPITRE 7: Évaluation

7.1 Introduction

Afin d'évaluer l'approche Recodyn, nous avons développé un logiciel baptisé DynaElec qui permet la configuration de la LP tableau électrique.

Pour la pertinence de l'étude de faisabilité de l'apporoche proposée, nous avons choisi de développer le moteur de recommandation Recodyn qui sera utilisé par DynaElec. Le choix de faire un moteur générique, indépendant de la LP, permet de montrer que l'approche est applicable à toute LP.

L'approche Recodyn est évaluée en termes de faisabilité, temps de réponse et passage à l'échelle.

7.2 Le moteur de recommandation Recodyn

Le moteur Recodyn est un programme qui propose des recommandations pour une LP par le calcul de similarité combiné à la résolution de contraintes. Le moteur développé est générique et indépendant de la spécification de la LP. En effet, il ne se base pas sur un type de données particulier, car une configuration est présentée sous forme d'un ensemble de couples (clé, valeur). Pour une caractéristique de la LP, la clé est toujours de type chaine de caractères et la valeur peut-être instanciée, selon la description de la LP, pour prendre différentes formes (Annexe B) :

- Types simples
- Objets complexes
- Liste de valeurs du même type dans le cas d'une caractéristique multi-instanciée.

Il se base sur plusieurs entrées :

- Spécifications de la LP (redéfinies comme un ensemble de variables et contraintes)
- Définitions des paquets constituant l'ensemble des caractéristiques.

Le moteur Recodyn permet de calculer les recommandations à l'aide d'une approche basée sur le contenu appliquée aux valeurs des caractéristiques et de propager les contraintes pour vérifier la cohérence des configurations comme le montre la Figure 32.

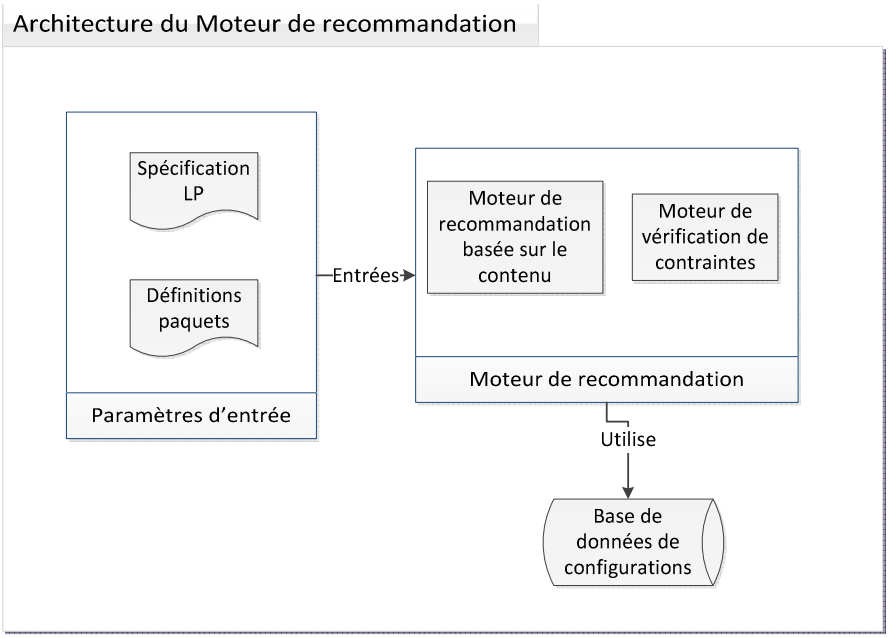


Figure 32 : Architecture du moteur de recommandation Recodyn

Le moteur est composé de deux sous-modules :

- Le module de recommandation basée sur le contenu : ce module est utilisé pour le calcul de la similarité entre les valeurs de la configuration partielle de l'utilisateur et les valeurs des configurations enregistrées dans la BD.

- Le module de vérification des configurations vérifie si la configuration recommandée par le module de recommandation respecte les contraintes de la LP. Si c'est le cas, la configuration recommandée est renvoyée à l'utilisateur, sinon, le module de recommandation cherche une autre configuration similaire.

7.3 Le Logiciel DynaElec

Afin d'évaluer la POC (Proof of concept) de Recodyn, nous avons développé un logiciel pour configurer une LP réelle. Ce logiciel, baptisé « DynaElec », permet de configurer le tableau électrique selon ses spécifications (cf. chapitre 5).

DynaElec intègre le moteur Recodyn et l'associe à :

- Des IHMs développées pour le cas de la LP tableau électrique
- La BD contenant les configurations relative à la LP

DynaElec permet à l'utilisateur de :

- Spécifier ses exigences sous forme de configuration partielle.
- Le guider dans ses choix tout le long du processus de configuration, en évaluant les décisions effectuées et recommandant d'autres.
- Pouvoir effectuer un retour en arrière, s'il décide de changer son choix sur des anciens paquets de caractéristiques.

La Figure 33 présente l'architecture globale de DynaElec avec l'intégration du moteur Recodyn. Les technologies utilisées pour l'implémentation de DynaElec sont JAVA, Maven, SQL Server 2008 et JDBC.

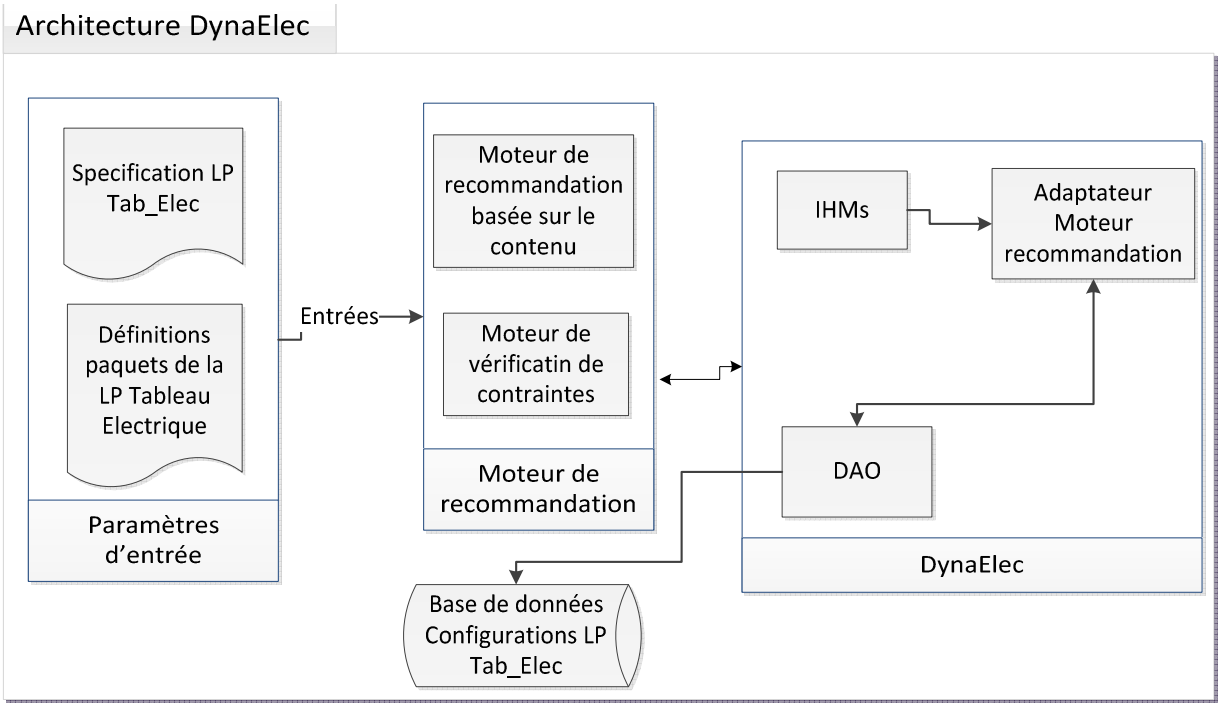


Figure 33 : Architecture globale DynaElec

Cette architecture est composée des éléments suivants :

- DynaElec :
  - L'interface graphique de l'utilisateur (IHM) : permet à celui-ci de faire ses choix.
  - L'adaptateur du moteur d'intégration : Ce module est le chef d'orchestre de DynaElec. Il représente le programme qui coordonne les

différents composants dans le but d'utiliser le moteur Recodyn pour la recommandation et la configuration d'un tableau électrique.

- Le DAO (Data Access Object) : permet de faire l'interface entre l'outil et la base de données pour extraire les données.

- Le moteur Recodyn
  - Moteur de recommandation CBF
  - Moteur de vérification de contraintes
- Les paramètres d'entrées spécifiques :
  - Le MLP du tableau électrique : permet la spécification de la LP tableau électrique au niveau de l'outil et la BD.
  - La définition des paquets de la LP
- La BD des configurations du tableau électriques : contenant des configurations générées par le solveur GnuProlog en se basant sur la spécification de la LP.

7.4 Expérimentation de DynaElec

Lors de la spécification de la LP pour DynaElec, les experts métier de Rexel ont choisi de prédéfinir sept paquets de caractéristiques dont le profil utilisateur. Le scénario de configuration d'un tableau électrique passe donc par sept IHMs sur lesquels sont effectués les choix relatifs aux caractéristiques de chaque paquet.

Le choix de la stratégie d'ordre de sélection des caractéristiques prédéfini a été fait en se basant sur les habitudes de ventes chez Rexel. Les paquets de caractéristiques ont été ordonnés selon le processus de configuration habituel (selon les experts-métier Rexel) du tableau électrique. Le premier des sept paquets (P_0) correspond au profil utilisateur sur lequel il spécifie ses exigences au début du processus.

P_0 = {Marque, technologie, type_logement, nombre_pièces, surface_logement, departement, surface_cuisine, équipement_cuisine, surface_sejour, nombre_socles, nombre_circuitEclairage}

P_1 = {nombre_rangées, nombre_modules_parRangee}

P_2 = {Rehausse, type_coffret, type_porte}

P_3 = {parafoudre}

P_4={télerupteur, contacteur, minuterie, interrupteur_horaire, télevariateur, temporisateur, gestionnaire_fil_pilote}

P_5 = {type_InterDiff, calibre_InterDiff}

P_6 = {type_disjoncteur, calibre _disjoncteur }

Scénario de configuration d'un tableau électrique

Profil utilisateur : La Figure 34 représente l'IHM permettant à l'utilisateur de spécifier ses exigences sur le premier paquet.

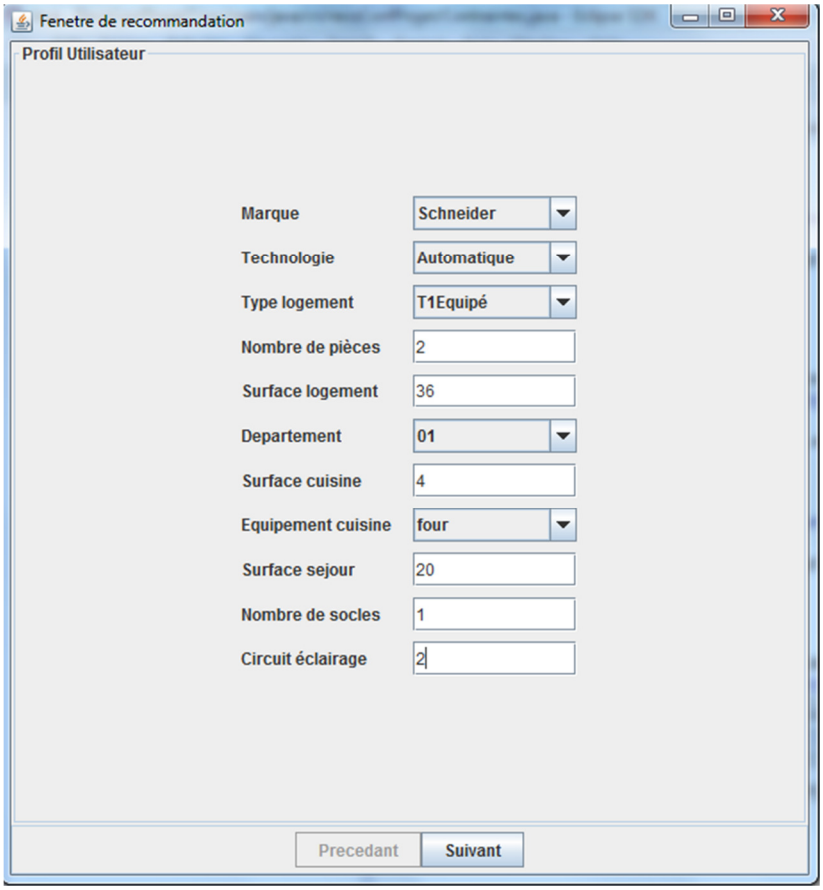


Figure 34 : Ecran de spécification des exigences

Pour les valeurs des caractéristiques prédéfinies comme la « *marque* », la « *surface_logement* », etc., l'utilisateur doit sélectionner une valeur parmi la liste des choix. Dans le cas contraire, il définit la valeur de la caractéristique correspondante. Après avoir spécifié les exigences de l'utilisateur, DynaElec vérifie les contraintes avant de commencer le calcul de similarité. La contrainte vérifiée dans ce cas est :

C_{01} : « Le nombre de circuits d'éclairage doit au moins être égal à deux dans les logements de surface supérieure à 35m². »

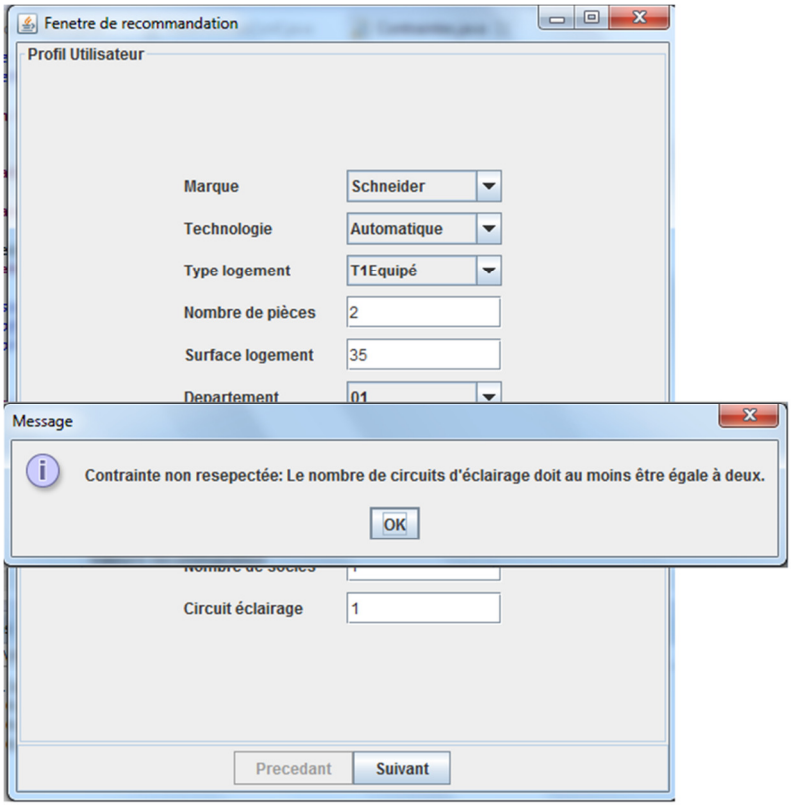


Figure 35 : Ecran de notification sur le paquet 0

Dans le cas où la spécification de l'utilisateur ne respecte pas cette contrainte, une notification est affichée à l'utilisateur lui demandant de réctifier son choix comme le montre la Figure 35.

Configuration du premier paquet : La Figure 36 présente la configurations sur le paquet P_1 = {nombre_rangées, nombre_modules_parRangee}. Les caractéristiques de ce paquet ne sont pas multi-instanciables.

Une fois les exigences de l'utilisateur sont spécifiées sur le paquet P_0, DynaElec calcule les similarités entre les exigences de l'utilisateur et les configurations de la BD et vérifie les contraintes des configurations partielles les plus similaires.

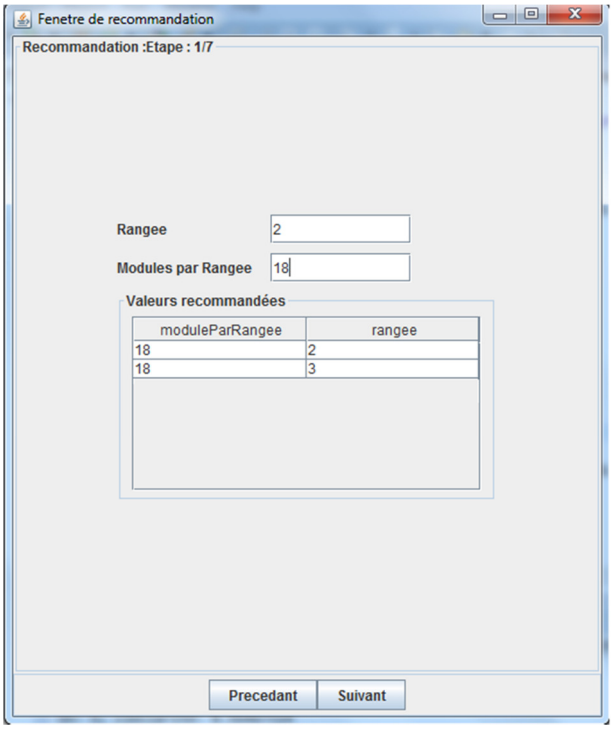


Figure 36 : Ecran de spécification du paquet 1

La seule contrainte à vérifier dans ce cas est qu'il faut avoir au moins 20% d'emplacements libres dans le coffret. Ainsi, deux configurations partielles sont recommandées à l'utilisateur. Ce dernier choisit la première qui lui est proposée.

Configuration du deuxième paquet : La Figure 37 présente la configuration sur le paquet P_2 = {Rehausse, type_coffret, type_porte}. Ce paquet concerne des caractéristiques optionnelles du tableau électrique permettant de spécifier s'il possède une rehausse, le type du coffret (encastré ou sailli) et le type de la porte (avec porte transparente, avec porte opaque, ou sans porte).

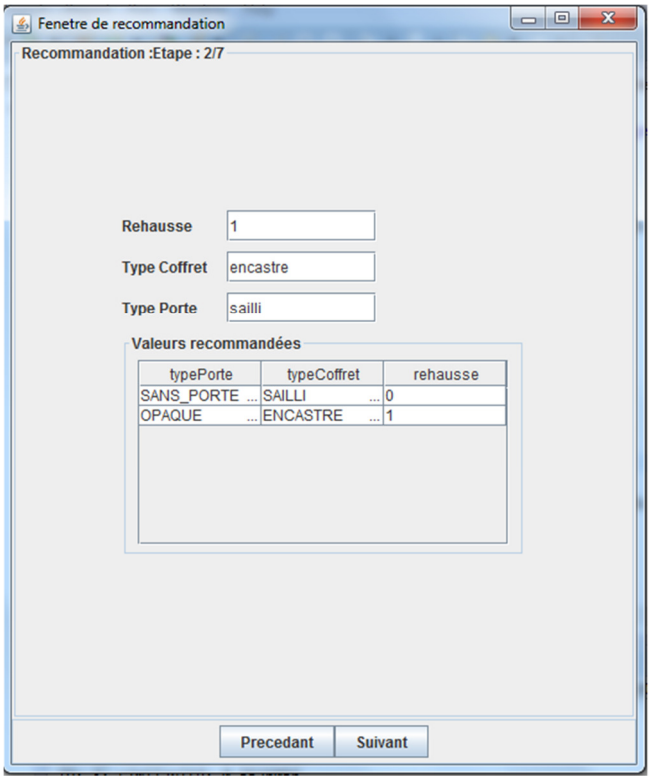


Figure 37 : Ecran de spécification du paquet 2

Dans cette étape, il n'existe pas de contraintes à vérifier (non spécifiées par Rexel). Ainsi, uniquement un calcul de similarité est effectué entre les nouvelles exigences de l'utilisateur spécifiées sur P_0 et P_1, et les configurations de la BD. Deux configurations partielles sont donc recommandées à l'utilisateur qui choisit l'une d'entre elles.

Configuration du troisième paquet : La Figure 38 représente la configuration sur le paquet P_3 = {parafoudre}. Dans cette étape, DynaElec recommande à l'utilisateur d'installer un parafoudre suite à son exigence spécifiée sur le paquet P_0. En effet, l'utilisateur a spécifié que son numéro de département est le « *01* » qui a niveau kéraunique supérieur à 25. Ainsi, l'installation du parafoudre a été recommandée par DynaElec.

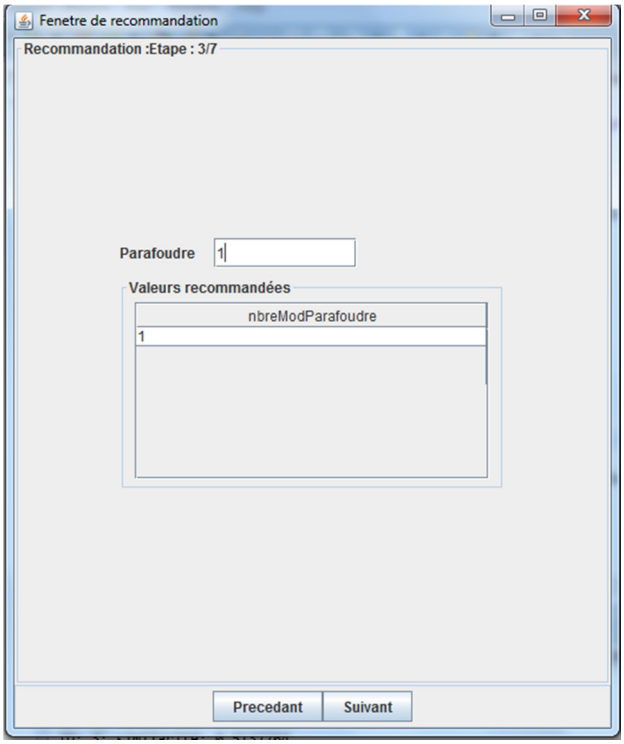


Figure 38 : Ecran de spécification du paquet 3

L'utilisateur a choisi de suivre la recommandation du système.

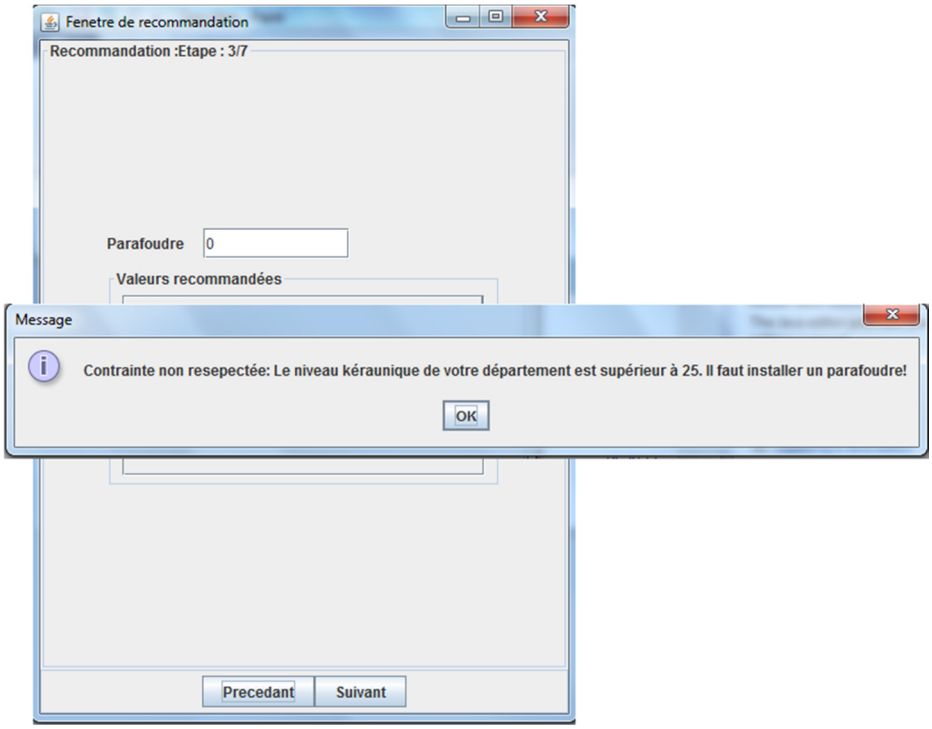


Figure 39 : Ecran de notification de contrainte non respectée sur le paquet 3

Dans le cas où l'utilisateur n'aurait pas choisi d'installer un parafoudre, une notification est affichée à l'utilisateur lui indiquant qu'il faut réctifier son choix comme le montre la Figure 39.

Configuration du quatrième paquet : La Figure 40 représente la configuration sur le paquet P_4={télerupteur, contacteur, minuterie, interrupteur_horaire, télevariateur, temporisateur, gestionnaire_fil_pilote}. Dans cette étape, le système recommande la présence ou l'absence dans la configuration des télérupteur, contacteur, minuterie, interrupteur horaire, télévariateur, temporisateur et le gestionnaire du fil pilote.

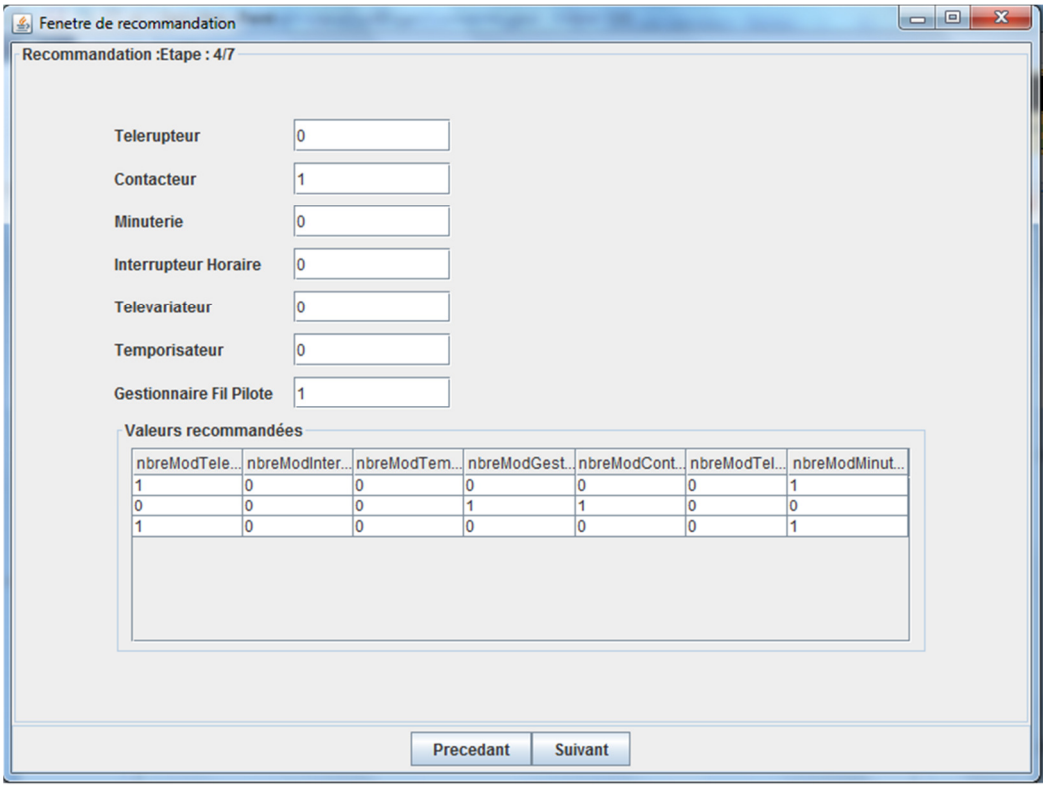


Figure 40 : Ecran de spécification du paquet 4

Les similarités sont calculées entre les nouvelles exigences de l'utilisateur spécifiées sur P_0, P_1, P_2 et P_3, et les configurations de la BD. Ensuite une vérification des contraintes est effectuée sur les configurations les plus similaires. Les contraintes à vérifier lors de cette étape sont :

C_{41} : « temporisateur exclut gestionnaireFilPilote »

C_{42} : « telerupteur exclut contacteur »

Trois configurations partielles ont été recommandées à l'utilisateur qui a choisi une parmi elles.

Configuration du cinquième paquet : La Figure 41 représente la configuration sur le paquet P_5 = {type_InterDiff, calibre_InterDiff}. Après le calcul de similarité, les contraintes suivantes sont vérifiées :

C_{51} : « Pour un logement de surface supérieure à 35m² et inférieure ou égale à 100m², au moins deux interrupteurs différentiels 30mA de courant assigné 40A et de type AC doivent être prévus. »

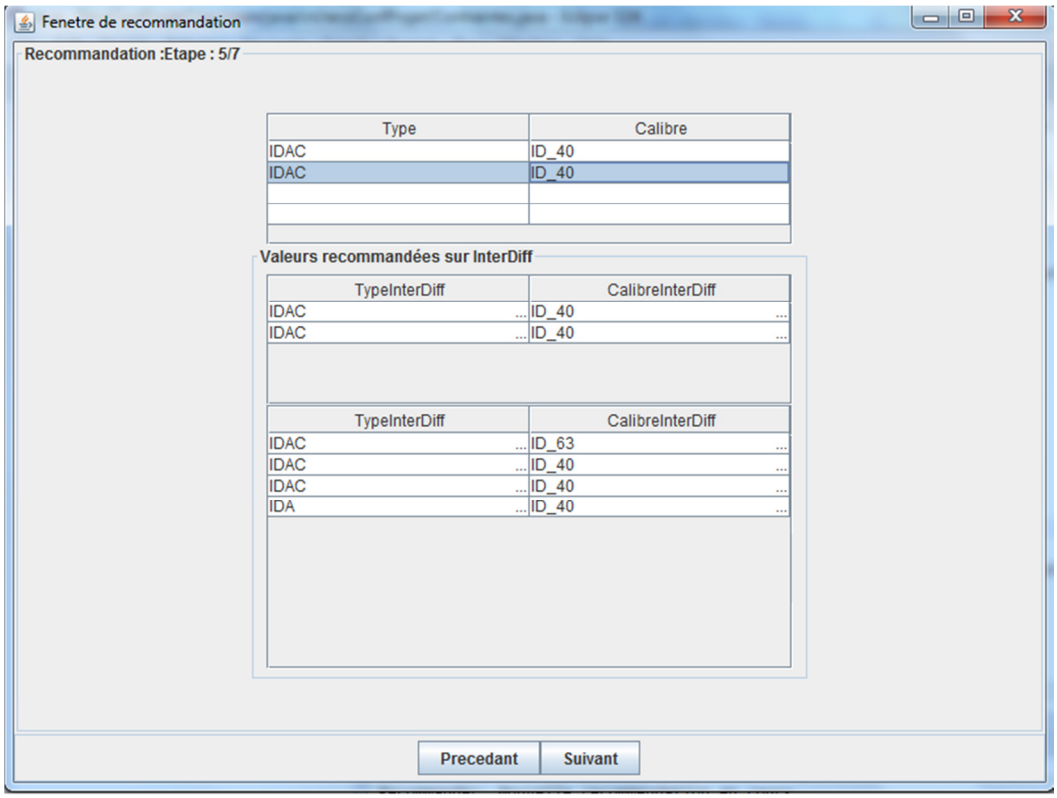


Figure 41 : Ecran de spécification du paquet 5

Comme il s'agit dans ce paquet de caractéristiques multi-instanciées, les valeurs de chaque instance correspondant au type de l'interrupteur differentiel et au calibre de l'interrupteur différentiel sont recommandées avec le nombre correspondant. Par exemple, DynaElec a recommandé en premier lieu une configuration partielle avec deux valeurs pour le calibre et deux valeurs pour le type.

Configuration du sixième paquet : La Figure 42 représente la configuration sur le paquet P_6 = {type_disjoncteur, calibre _disjoncteur }. La similarité est calculée entre les exigences spécifiées sur les paquets P_0, P_1, P_2, P_3, P_4 et P_5 et les configurations de la BD. Comme il s'agit de caractéristiques multi-instanciables sur le paquet P_5, la similarité est calculée de manière imbriquée entre chacune des valeurs du type de l'interrupteur différentiel (resp. calibre de l'interrupteur différentiel) spécifiée dans les exigences de

l'utilisateur et celles des configurations de la BD. Ensuite, une vérification de la contrainte C_{61} est effectuée.

C_{61} : « Les circuits d'éclairage doivent être protégés par un disjoncteur de courant assigné maximal de 16A. »

Comme il existe deux circuits d'éclairage dans les exigences spécifiées par l'utilisateur, il doit donc y avoir au moins deux disjoncteurs de calibre 16A.

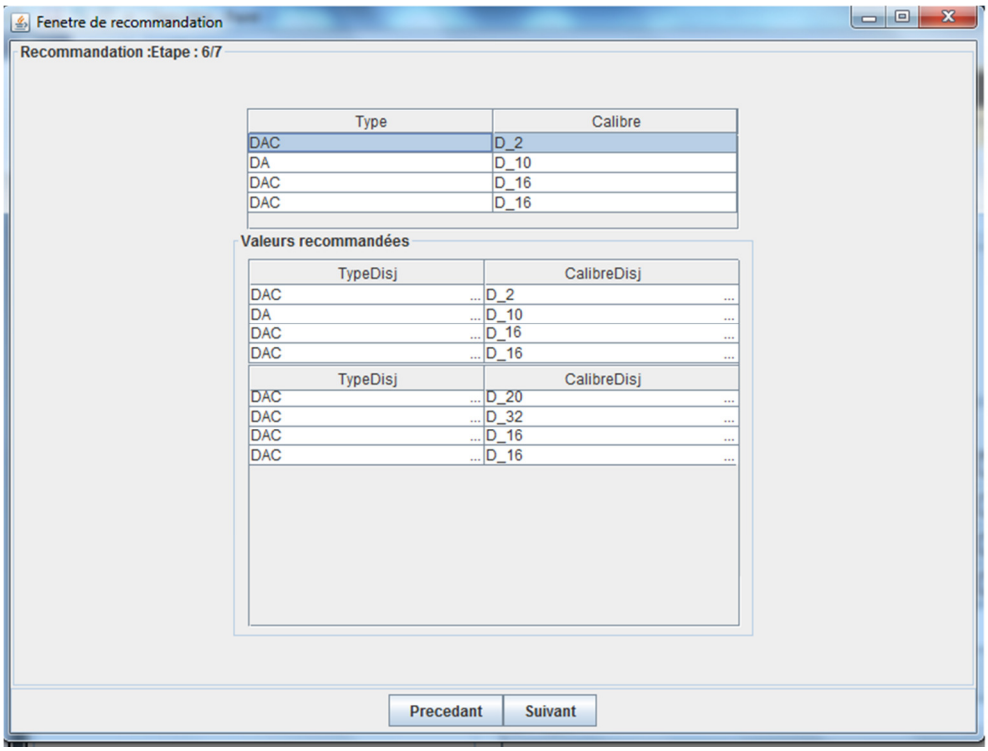


Figure 42 : Ecran de spécification du paquet 6

L'utilisateur fait son choix sur le paquet en sélectionnant une des deux recommandations proposées.

Résultats

La Figure 43 récapitule le résultat final qui consiste en une configuration complète.

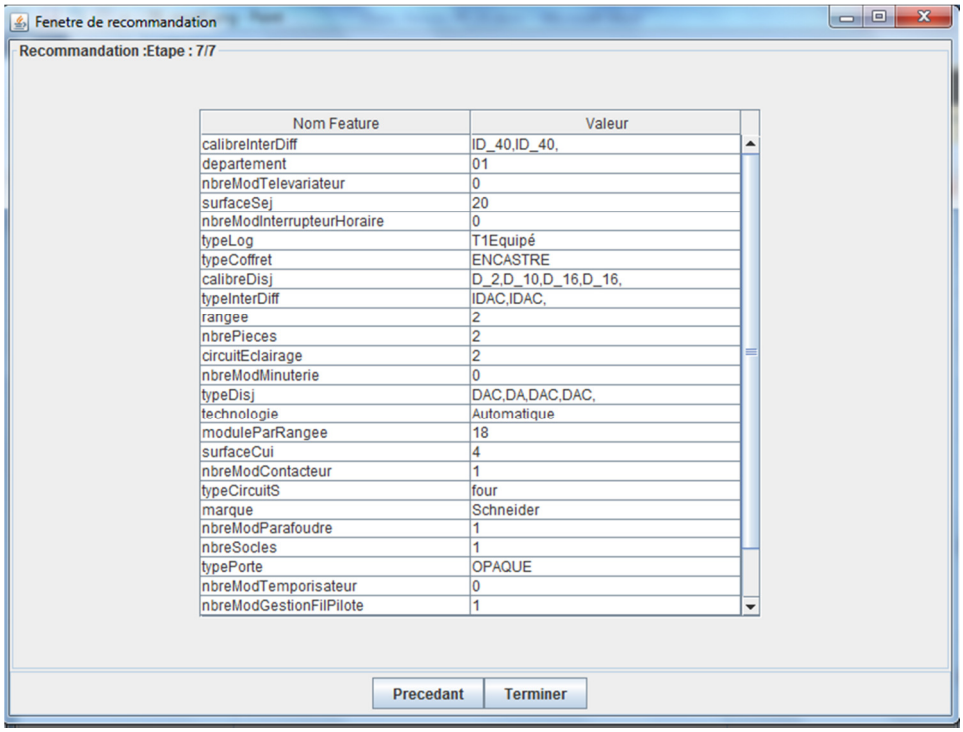


Figure 43 : Exemple d'une configuration complète

La recommandation complète fournie par DynaElec est représentée comme un vecteur des caractéristiques de la LP et un vecteur des valeurs des caractéristiques choisies lors du processus de configuration et recommandation.

7.5 Evaluation de l'expérimentation

7.5.1 Faisabilité « Proof of Concept »

DynaElec a permis de réaliser la recommandation durant la configuration de la LP tableau électrique. Cette implémentation de Recodyn a permis de montrer la faisabilité et l'efficacité de l'approche proposée. Les résultats de cette expérimentation ont été approuvés par Rexel.

7.5.2 Analyse des données

Les données stockées dans la BD de l'outil DynaElec sont sous forme de configurations complètes. Ces configurations représentent deux groupes :

- Des configurations générées par le solveur GnuProlog.
- Des configurations représentant les profils client et des achats qui ne violent pas la spécification de la LP. Les achats antérieurs peuvent être des anciens

achats du client lui-même ou d'autres clients. Ces achats peuvent être de différents types :

- Achats d'éléments avec la même marque
- Achats d'éléments avec différentes marques

7.5.3 Temps de réponse

Pendant les premières exécutions de l'outil DynaElec, il était remarquable que le calcul de certaines opérations fût long. L'analyse du problème a révélé qu'il y avait des processus redondants. Le problème concerne en particulier le calcul du poids. De façon générale, les caractéristiques sont instanciées une seule fois dans une seule configuration. Ainsi, le poids d'un élément reste presque toujours le même. Pour améliorer les performances de l'outil, une technique de buffering (utilisation de mémoire tampon) a été développée. Elle consiste à éviter le calcul de poids de toute opération effectuée plus d'une fois. Dans ce cas, le système n'exécute pas à nouveau l'opération. A la place, il récupère directement le résultat (poids), déjà calculé et enregistré dans la mémoire tampon. Les performances de Recodyn en termes de temps de réponse se sont nettement améliorées à l'issue de cette adaptation.

Le résultat final est présenté à la Figure 44. Les temps de réponse enregistrés représentent des moyennes calculées sur plusieurs mesures. L'environnement dans lequel se sont effectuées ces mesures est :

- Un ordinateur portable DELL LATITUDE E5400
- Système d'exploitation : Windows 7
- Processeur : Intel (R) Core (TM) 2 DUO CPU P8600 2.4 GHz
- Mémoire : 4Go

L'exemple du tableau électrique utilisé pour l'évaluation contient 27 caractéristiques, 15 contraintes et 530 configurations dans la BD.

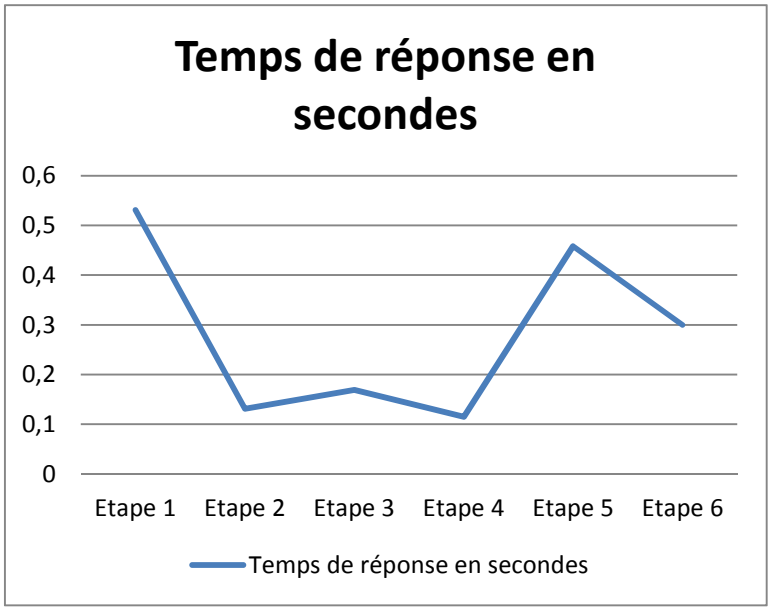


Figure 44 : Temps de réponse en fonction de chaque étape

A l'issue de cette évaluation, plusieurs intérprétations sont éffectuées :

- Le temps de réponse est plus important à la première étape. Cette étape correspond à la configuration sur le premier paquet de caractéristiques. Il contient le plus de caractéristiques selon le choix de Rexel. Par conséquent, le nombre d'opérations de calcul de poids et la vérification des contraintes qui sont effectués pour la première fois seront évités aux prochaines étapes grâce à la mémoire tampon. La recommandation s'effectue en tenant compte des décisions effectuées précédemment. En fonction de l'avancement de ces décisions, le filtrage sur la BD devient plus restreint et par conséquent le résultat est plus rapide. Cette progressivité du processus de recommandation entraine une diminution du temps de réponse en fonction de l'avancement.
- Une autre augmentation du temps de réponse est constatée à l'étape 5 puisqu'elle représente des caractéristiques multi-instanciées avec des contraintes présentant des dépendances importantes.
- Le temps de réponse le plus long est inférieur à 1s.

7.5.4 Le passage à l'échelle

Le passage à l'échelle est défini de façon générale comme « *l'aptitude d'une solution à un problème de fonctionner même si la taille du problème augmente* » (Andrey et al., 2006). Dans notre contexte, on s'intéresse à évaluer si Recodyn maintient sa performance

en terme de temps de réponse dans son environnement d'exécution lors de la montée en charge des données à gérer. Comme Recodyn repose sur un algorithme de filtrage par contenu, le temps de réponse dépend principalement de deux critères :

1. La taille de la BD
2. La complexité de la LP

Ces deux critères jouent un rôle important sur les requêtes de filtrage exécutées sur la BD.

1. La taille de la BD : Dans un filtrage par contenu, le volume de données dans la BD est un critère important auquel le temps de réponse est corrélé, car il est crucial de répondre à la requête de l'utilisateur en un temps raisonnable. L'expérimentation, présentée par la Figure 45, présente l'évolution du temps de réponse en fonction de l'augmentation progressive de la taille de la BD.

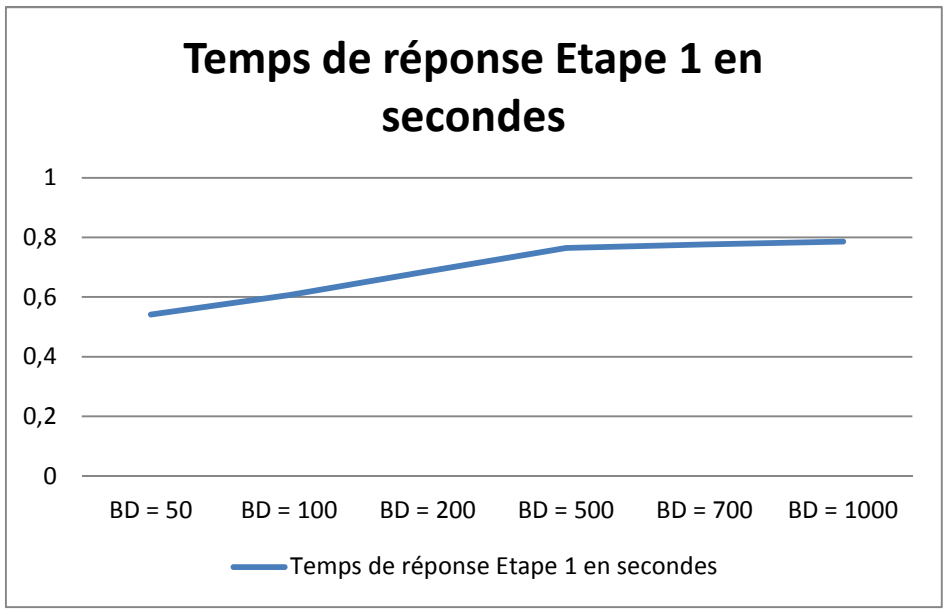


Figure 45 : Le temps de réponse en fonction du nombre de configurations pour la première étape

L'étape 1 est choisie pour effectuer l'évaluation puisqu'elle a été identifiée comme l'étape la plus couteuse en temps de réponse. Nous avons remarqué que le temps de réponse a augmenté d'une façon linéaire en fonction du volume jusqu'au seuil de 500 configurations enregistrées dans la BD. A partir de ce seuil le temps de réponse reste

stable. DynaElec garde ses performances en termes de temps de réponse en augmentant la taille de la BD. Nous avons continué l'expérimentation jusqu'à doubler ce seuil et nous avons observé la même stabilité. A un seuil de 1000 configurations nous avons arrêté l'expérimentation avec une conclusion positive quant à la montée en charge donnant un temps de réponse qu'on estime raisonnable.

2. La complexité de la LP : Pour tester le temps de réponse de Recodyn en terme de complexité de la LP, nous avons estimé que la LP tableau électrique est suffisamment complexe avec 27 caractéristiques et 15 contraintes implémentées. Les temps de réponse observés à la Figure 45 correspondant à une taille de BD de 1000 configurations et une LP complexe sont satisfaisants dans ce contexte.

7.6 Les menaces à la validité

Les principales menaces de validité sont comme suit :

Validité de la conclusion : Le passage à l'échelle a été mesuré en termes de la taille de la BD des achats passés. C'est le critère le plus important pour les temps de calcul, mais il serait judicieux de mesurer aussi ce critère en termes de taille du MLP, du nombre et la nature des contraintes, et du nombre de clients utilisant le système afin d'en analyser l'effet sur les performances. Nous estimons que ce dernier critère reste maitrisable par le biais de solutions existantes de montée en charge pour le Web, comme les serveur web, serveur d'applications ainsi que des BDs supportant cette montée en charge, voire du Bigdata. Ces aspects sont à définir au cas par cas pour chaque entreprise souhaitant mettre en place l'approche.

- **Validité interne :** Les données stockées dans la BD ne sont pas toutes collectées à partir des achats réels de l'entreprise Rexel. Elles sont principalement générées à l'aide du solver GNU Prolog. Cela n'affecte pas négativement les résultats de Recodyn, mais ne reflète pas les résultats sur des achats réels.
- **Validité conceptuelle :** En raison de contraintes de temps, la recherche n'a été appliquée que sur un seul cas réel. Pour pousser encore la généralisabilité, cette recherche devrait s'appliquer à différents cas de LP. Une option serait

d'analyser le comportement de Recodyn face à des complexités de LP différentes avec un ordre de nombre de critères différents.

- **Validité externe :** Dans le cas de notre approche, notre évaluation s'est basée principalement sur le temps de réponse et le passage à l'échelle. Dans d'autres contextes, d'autres critères pourraient se rajouter.

7.7 Conclusion

Ce chapitre a présenté une description de l'architecture du moteur de recommandation Recodyn. Puis, il a présenté l'application de Recodyn sur le cas réel du tableau électrique par le biais du logiciel DynaElec. Ensuite, une analyse des résultats expérimentaux a été effectuée. Ces expérimentations ont été réalisées dans un contexte industriel avec la société Rexel. L'application de l'approche Recodyn sur l'exemple du tableau électrique a montré sa faisablilté de notre approche, ses bonnes performances en termes de temps de réponse et son passage à l'échelle en augmentant la taille de la BD.

CHAPITRE 8: Conclusion et Perspectives

8.1 Conclusion

Dans cette thèse, une nouvelle approche de recommandation est proposée pour les produits configurables notamment les LPs. Elle permet d'accompagner l'utilisateur à prendre des décisions pendant le processus de configuration et de lui proposer des recommandations correspondant à ses exigences en respectant les contraintes de la LP. Cette thèse vise alors à répondre à la question de recherche pertinente suivante :

« Comment combiner la recommandation avec la configuration des LPs ? »

Pour ce faire, les contributions suivantes ont été réalisées dans cette thèse :

1. Un état de l'art sur les approches de configuration des LPs, les types de recommandation et leurs techniques. Cette étude a montré que :
    - Les approches de configuration ne sont pas suffisantes pour accompagner l'utilisateur à configurer un produit complexe selon ses exigences. Une forme complémentaire de guidage, telle que la recommandation, est alors nécessaire.
    - Les TRs ne sont pas uniquement applicables pour les produits simples, elles peuvent être adaptées et appliquées dans le cas des produits complexes.
2. Un cas réel (concernant la LP du tableau électrique) a été développé dans un contexte industriel avec l'entreprise Rexel. Cette étude a été, en premier lieu, conduite à des fins exploratoires. Elle nous a permis de reformuler les objectifs du projet de recherche, de soulever puis de résoudre un certain nombre de verrous tels que l'impact de la modélisation de la multi-instanciation dans une LP sur la recommandation et configuration de LP, et redéfinir cette dernière. Dans ce contexte, la méthode recherche-action a été utilisée pour le processus de validation.
3. Trois catégories de langages de modélisation de la multi-instanciation ont été proposées et étudiées en les utilisant pour modéliser la LP du tableau électrique. Une comparaison entre les trois catégories de langages a été réalisée en discutant les avantages et limites de chacune. Suite à cette comparaison, un arbre de décision a été proposé permettant de guider l'utilisateur à choisir la catégorie de langages de modélisation adéquate à son contexte.
4. L'approche de recommandation Recodyn a été proposée pour la configuration interactive des LPs. Elle génère des recommandations en interagissant avec l'utilisateur. Ce dernier a la possibilité de spécifier ses exigences au fur et à mesure d'étapes du processus de configuration. Elle combine la recommandation basée sur la connaissance avec celle basée

sur le contenu. Son principe est de raisonner sur un sous-ensemble de caractéristiques. La recommandation est calculée par une mesure de similarité entre les exigences de l'utilisateur définies sur un paquet de caractéristiques et les configurations partielles de la BD spécifiées sur le même paquet. En utilisant la spécification de la LP, une vérification des contraintes est effectuée sur les recommandations similaires retournées. Les recommandations sont sous forme de configurations partielles des valeurs de caractéristiques. Recodyn permet de prendre en compte les caractéristique multi-instanciées et retourne des recommandations suivant la catégorie de langages de modélisation.

5. Une Evaluation de l'approche Recodyn a été réalisée. Pour ce faire, le moteur de recommandation générique Recodyn a été développé. Il s'agit de l'implémentation logicielle de l'approche Recodyn. L'outil DynaElec a été développé. Il utilise le moteur Recodyn dans le cas du tableau électrique.

8.2 Perspectives

Cette section présente quelques perspectives de recherche sur la recommandation dans un contexte de configuration de produits.

Une des perspectives de recherche est d'utiliser d'autres stratégies de sélection des caractéristiques comme la stratégie dynamique en fonction de l'avancement du processus de configuration, ou celle basée sur des fonctions objectives pour faire des optimisations.

Une autre perspective est de réaliser un benchmark de plusieurs LPs modélisées avec différents langages afin d'évaluer l'approche Recodyn.

Finalement plusieurs TRs pourraient être utilisées pour la configuration d'une LP. Une comparaison de performance et de pertinence de chacune serait réalisée. Une proposition de combiner à la fois plusieurs TRs pourraient aussi être incluses dans cette étude. Pour la TR utilisée pour l'expérimentation de l'approche proposée, les bases de données relationnelles ont été suffisantes. Dans le contexte du passage à l'échelle avec des quantités de données importantes, une étude de combiner l'utilisation des TRs avec des approches de traitement de méga données tel que Bigdata, ElasticSearch ou No SQL pourrait être envisagée.

Annexe A

Modélisation du tableau électrique

Des détails supplémentaires de la modélisation du tableau électriques sont présentées ci-dessous. Les contraintes du tableau électrique fournies par Rexel sont :

- Les départements dont le niveau kéraunique est supérieur à 25 sont : 01, 04, 05, 06, 07, 13, 2A, 2B, 24, 25, 26, 30, 33, 34, 38, 39, 40, 42, 43, 47, 48, 63, 66, 69, 71, 73, 74, 83, 84, 971, 972 et 973.
- Il doit être prévu au moins trois circuits spécialisés pour l'alimentation d'appareils du type : lave-linge, lave-vaisselle, sèche-linge, four, congélateur.
- Lorsque l'emplacement du congélateur est défini, il convient de prévoir un circuit spécialisé protégé par DDR 30mA spécifique à ce circuit.
- Dans le cas particulier des logements de type T1, et si l'équipement n'est pas fourni, trois circuits spécialisés au moins sont prévus (un circuit 32A et deux circuits 16A).
- Si une application parmi chauffe-eau électrique, chaudière, pompe à chaleur, climatiseur, chauffage électrique, chauffage de salle de bain, piscine, circuits extérieurs, volets roulants électriques, automatismes domestiques, VMC non collective, tableau divisionnaire est prévue, il doit exister un circuit spécialisé.
- Le nombre de circuits d'éclairage doit au moins être égal à deux dans les logements de surface supérieure à 35m².
-
- Si la puissance des appareils ou équipements de chauffage alimentés par le même circuit est comprise entre 5750W et 7250W, la protection est assurée par un disjoncteur de courant assigné 32A.
- Si la puissance des appareils ou équipements de chauffage alimentés par le même circuit est comprise entre 4500W et 5750W, la protection est assurée soit par un disjoncteur de courant assigné 25A et la section minimale des conducteurs est 4mm², soit par un fusible de courant assigné 25A.
- Si la puissance des appareils ou équipements de chauffage alimentés par le même circuit est comprise entre 3500W et 4500W, la protection est assurée

soit par un disjoncteur de courant assigné 20A, soit par un fusible de courant assigné 20A.

- Si la puissance des appareils ou équipements de chauffage alimentés par le même circuit est comprise entre 2250W et 3500W, la protection est assurée soit par un disjoncteur de courant assigné 16A, soit par un fusible de courant assigné 16A.
- Si la puissance des appareils ou équipements de chauffage alimentés par le même circuit est inférieure à 2250W, la protection est assurée soit par un disjoncteur de courant assigné 16A, soit par un fusible de courant assigné 10A.
- Pour un logement de surface inférieure ou égale à 35m², au moins un interrupteur différentiel 30mA de courant assigné 25A et de type AC doit être prévu.
- Pour un logement de surface supérieure à 35m² et inférieure ou égale à 100m², au moins deux interrupteurs différentiels 30mA de courant assigné 40A et de type AC doivent être prévus.
- Pour un logement de surface supérieure à 100m², au moins trois interrupteurs différentiels 30mA de courant assigné 40A et de type AC doivent être prévus.
- Lorsque des circuits de chauffage et de chauffe-eau électriques, dont la somme des puissances est supérieure à 8kVA, sont placés en aval d'un même interrupteur différentiel, il faut remplacer un interrupteur différentiel 40A de type AC par un interrupteur différentiel 63A de type AC.
- Dans le cas particulier où l'interrupteur différentiel 40A de type A est amené à protéger un ou deux circuits spécialisés supplémentaires, son courant assigné doit être égal à 63A.
- Il est conseillé de protéger par des DDR 30mA spécifiques les appareils mêlant eau et électricité (lave-linge, lave-vaisselle, etc.).
- Les circuits d'éclairage, de volets roulants, de prises commandées doivent être protégés par un fusible de courant assigné maximal 10A ou par un disjoncteur de courant assigné maximal de 16A.
- Les circuits de VMC doivent être protégés par un disjoncteur de courant assigné maximal de 16A.

- Les circuits d'asservissement tarifaire, le fil pilote, le gestionnaire d'énergie, etc., doivent être protégés par un disjoncteur de courant assigné maximal de 2A.
- Les circuits de chauffe-eau électrique non instantané doivent être protégés par un fusible de courant assigné maximal 16A ou par un disjoncteur de courant assigné maximal de 20A.
- Le tableau de répartition doit comporter 20% d'emplacements libres.
- La dimension du peigne vertical correspond au nombre de rangées à alimenter dans le coffret.

Le modèle TVL complet est présenté ci-dessous :

```
int numRangee in {1, 2, 3, 4, 5, 6};
int numModule in {13, 18, 24};
int hauteur in {1, 2, 3, 4, 5, 6};
int largeur in [0..24];

enum departements in {Dpt1, Dpt2, Dpt3, Dpt4, Dpt5, Dpt6,
Dpt7, Dpt8, Dpt9, Dpt10, Dpt11, Dpt12, Dpt13, Dpt14, Dpt15,
Dpt16, Dpt17, Dpt18, Dpt19, Dpt2A, Dpt2B, Dpt21, Dpt22,
Dpt23, Dpt24, Dpt25, Dpt26, Dpt27, Dpt28, Dpt29, Dpt30,
Dpt31, Dpt32, Dpt33, Dpt34, Dpt35, Dpt36, Dpt37, Dpt38,
Dpt39, Dpt40, Dpt41, Dpt42, Dpt43, Dpt44, Dpt45, Dpt46,
Dpt47, Dpt48, Dpt49, Dpt50, Dpt51, Dpt52, Dpt53, Dpt54,
Dpt55, Dpt56, Dpt57, Dpt58, Dpt59, Dpt60, Dpt61, Dpt62,
Dpt63, Dpt64, Dpt65, Dpt66, Dpt67, Dpt68, Dpt69, Dpt70,
Dpt71, Dpt72, Dpt73, Dpt74, Dpt75, Dpt76, Dpt77, Dpt78,
Dpt79, Dpt80, Dpt81, Dpt82, Dpt83, Dpt84, Dpt85, Dpt86,
Dpt87, Dpt88, Dpt89, Dpt90, Dpt91, Dpt92, Dpt93,    Dpt94,
Dpt95, Dpt971, Dpt972, Dpt973, Dpt974, Dpt976};

enum nomMarque in {Schneider, Legrand, Hager};

enum typeLogement in {T1Equipe, T1NonEquipe, Autre};
```

```
    enum typeCoffret in {Encastre, Sailli};

    enum typeID in {IDAC, IDA, IDASi};
    enum calibreID in {ID25, ID40, ID63};

    enum typeDisj in {DAC, DA, DASi };
    enum calibreDisj in {D2, D6, D10, D16, D20, D25, D32};

    enum typeDisjDiff in {DDAC,  DDASi};
    enum calibreDisjDiff in {DD10, DD16, DD20, DD25, DD32};

    enum typeFusible in {FA, FASi};
    enum calibreFusible in {F10, F16, F20, F25};

    root Rexel {
    group allof {
    Logement {
    int surface;
         Logement.surface >=0;
         Logement.surface >= 35 -> count(Circuit) >= 2;
         Logement.surface          <=          35          ->
count(InterDiff.filter(InterDiff.type==IDAC))>=1          &&
count(InterDiff.filter(InterDiff.calibre==ID25))>=1;
        Logement.surface > 35 && Logement.surface <= 100 ->
count(InterDiff.filter(InterDiff.type==IDAC))>=2          &&
count(InterDiff.filter(InterDiff.calibre==ID40))>=2;
        Logement.surface          >          100          ->
count(InterDiff.filter(InterDiff.type==IDAC))>=3          &&
count(InterDiff.filter(InterDiff.calibre==ID40))>=3;

        int nbrPieces;
        nbrPieces >= 0;
        nbrPieces  ==  count(Chambre)  +  count(Cuisine)  +
count(Sejour) + count(WC) ;
```

```
        departements departement;
        departement in {Dpt1, Dpt4, Dpt5, Dpt6, Dpt7, Dpt13,
Dpt2A, Dpt2B, Dpt24, Dpt25, Dpt26, Dpt26, Dpt30, Dpt33,
Dpt34, Dpt38, Dpt39, Dpt40, Dpt42, Dpt43, Dpt47, Dpt48,
Dpt63, Dpt66, Dpt69, Dpt71, Dpt73, Dpt74, Dpt83, Dpt84,
Dpt971, Dpt972, Dpt973 } -> count(Parafoudre) > 0;

        typeLogement type;
        Logement.type != T1Equipe ->
count(CircuitSpecialise.filter(Congelateur || Four ||
LaveVaisselle || SecheLinge || LaveLinge )) >=3;
        Logement.type == T1Equipe ->
count(CircuitSpecialise.filter(Congelateur || Four ||
LaveVaisselle || SecheLinge || LaveLinge )) == 1;
        Logement.type == T1NonEquipe ->
count(Disjoncteur.filter(Disjoncteur.calibre == D32))==1 &&
count(Disjoncteur.filter(Disjoncteur.calibre == D16))==2;

        group allof {
             Chambre [0..1],
             Cuisine [0..1] {
                  int surfaceCuis;
                  surfaceCuis >= 0;
             },
             Sejour [0..1] {
                  int surfaceSej;
                  surfaceSej >= 0;
             },
             WC [0..1],
             Coffret {
                  int marge;
                  marge >=0 ;
                  numRangee rangees;
```

```
                    numModule modulesParRangee;
                    typeCoffret type;
                    nomMarque marque;
                    bool automatique;
                    (modulesParRangee * rangees * 10) > (12 *
               (sum(InterDiff.map(nbrModuleID))
    +sum(Parafoudre.map(nbrModuleP))
    +sum(Disjoncteur.map(nbrModuleD))
+sum(DisjDiff.map(nbrModuleDD))
+sum(Fusible.map(nbrModuleF))
+sum(PeigneHorizontal.map(nbrModulePE))
    +Contacteur.nbrModuleConta+ Telerupteur.nbrModuleTel
    +sum(Minuterie.map(nbrModuleMinuterie))
+sum(Temporisateur.map(nbrModulesTemp))
    +sum(GestionnaireFilPilote.map(nbrModulesGFP))
+sum(InterrupteurHoraire.map(nbrModuleIH))
    +sum(TeleVariateur.map(nbrModuleTelevariateur))
                    )
    ) ;

                    group someof {
                         Porte {
                              group oneof {
                                   Opaque,
                                   Transparent
                              }
                         },
                         Rehausse,
                         PeigneVertical {
                              hauteur h;
                              h == rangees;
                         },
                         PeigneHorizontal [1..2] {
                              bool embroche;
```

```
                              embroche -> automatique;
                              !embroche -> !automatique;
                              largeur nbrModulePE;
                              embroche -> nbrModulePE > 0;
                              !embroche -> nbrModulePE == 0;
                              group allof {
                                   Protection {
                                        group [0..2] {
   InterDiff {
   Largeur nbrModuleID;
                                          typeID type;
   calibreID calibre;


   sum(CircuitChauffage.map(puissanceChauffage))+sum(CircuitC
hauffeEau.map(puissanceChauffeEau)) > 8000 -> InterDiff.type
== IDAC && InterDiff.calibre == ID63;

   count(Circuit)     >     count(CircuitSpecialise)     ->
InterDiff.calibre == ID63;
                                        },

                                        Parafoudre {
   largeur nbrModuleP;
                                        }
                                   }
                              },
                              CircuitProtege [1..2] {
                                   group allof {
                                        ProtectionCircuit{
                                             group oneof {
                                              Disjoncteur{
   Largeur nbrModuleD;
   calibreDisj calibre;
```

```
    typeDisj type;
    },

                                                    DisjDiff {
    largeur nbrModuleDD;
    typeDisjDiff type;
    calibreDisjDiffcalibre;
    },

                                                    Fusible {
    largeur nbrModuleF;
    typeFusible type;
    calibreFusible calibre;
                                                      }
                                                   }
                                              },
                                              Circuit  [1..2]  {
                                                   group oneof {
                    CircuitEclairage{
    CircuitEclairage                                          ->
count(Disjoncteur.filter(Disjoncteur.calibre  ==  D16))==1  ||
count(Fusible.filter(Fusible.calibre == F10))==1;
                                              },

                                              CircuitPC16A {
    int nbrSocles in [0..8];
                                              },

                                              CircuitSpecialise   {
                                                 group oneof {
                                                      LaveLinge {
    count(DisjDiff) > 0;
                                                        },
                                              LaveVaisselle{
```

```
                                        count(DisjDiff) > 0;
                                               },
                                               Four {
                                        count(DisjDiff) > 0;
                                               },
                                           SecheLinge {
                                        count(DisjDiff) > 0;
                                           },
                                           Congelateur {
                                        count(DisjDiff) > 0;
                                           },
                                         CircuitVoletsRoulants{
                                       CircuitVoletsRoulants  -
> count(Disjoncteur.filter(Disjoncteur.calibre == D16))==1 ||
count(Fusible.filter(Fusible.calibre == F10))==1;
                                       },
                                            CircuitVMC {
                                         CircuitVMC          ->
count(Disjoncteur.filter(Disjoncteur.calibre  ==  D16))==1  ;
                                         },
                                       CircuitChauffeEau {
                                  int puissanceChauffeEau;
                                  puissanceChauffeEau >= 0;
                                  CircuitChauffeEau          ->
count(Disjoncteur.filter(Disjoncteur.calibre  ==  D20))==1  ||
count(Fusible.filter(Fusible.calibre == F16))==1;
                                       },
                                       CircuitCuisson,
                                       CircuitChauffage {
                                       bool filPilote;
                                       int puissanceChauffage;
                                            puissanceChauffage
>=0;
```

```
   puissanceChauffage          <=          3500          ->
count(Disjoncteur.filter(Disjoncteur.calibre == D16))==1;

   puissanceChauffage > 3500 && puissanceChauffage <= 4500 ->
count(Disjoncteur.filter(Disjoncteur.calibre == D20))==1;

   puissanceChauffage > 4500 && puissanceChauffage <= 5750 ->
count(Disjoncteur.filter(Disjoncteur.calibre == D25))==1;

   puissanceChauffage > 5750 && puissanceChauffage <= 7250 ->
count(Disjoncteur.filter(Disjoncteur.calibre == D32))==1;
                                        }
                                   }
                              }
                              }
                              }
                              }
                              }
                              }
   },
                              CommandeEclairage {
                                   group [1..2] {
                                        Telerupteur {
                                        largeur nbrModuleTel;
                                        },
                                        Contacteur {
                                        largeur nbrModuleConta;
                                        },
                                        Minuterie {
                                   largeur nbrModuleMinuterie;
                                        }
                                   }
                              Telerupteur excludes Contacteur;
                              },
```

```
                              CommandeChauffage {
                                   group oneof {
                                        Temporisateur {
                                        largeur nbrModulesTemp;
                                        },
                                        GestionnaireFilPilote {
                                        largeur nbrModulesGFP;
   GestionnaireFilPilote                                          ->
count(Disjoncteur.filter(Disjoncteur.calibre == D2))==1;
                                        }
                                   }
                              },
                              opt Programmation {
                                   group someof {
                                        InterrupteurHoraire {
                                        largeur nbrModuleIH;
                                        },
                                        TeleVariateur {
                              Largeur nbrModuleTelevariateur;
                                        }
                                   }
                              }
                         }
                    }
               }
          }
    }
    }
```

Annexe B

Implémentation du moteur Recodyn et du logiciel DynaElec

L'implémentation du moteur Recodyn est présentée ci-dessous.

Algorithme 1. Algorithme de recommandation des configurations partielles similaires

```
     private List<Integer> listeIdAllConf_  ;
     private List<Feature> listeFeatures = new
ArrayList<Feature>();


     public List< Map<String, Object>> recommander (final
Map<String, Object> confCl ) {

          initReco(confCl);

          TreeMap<Float, Map<String, Object>> recommanded = new
TreeMap<Float, Map<String, Object>
>(Collections.reverseOrder());
          List<Map<String, Object> > liste = new
ArrayList<Map<String, Object> >();

          int numSouSConfCourant =
ParamsSousConf.identifierSousConf(confCl);
          int numSousConfSuivant = numSouSConfCourant+1;

          System.out.println("Recommander: Nouvelle
recommandation en cours...");
          float similarite = 0;
          for (int id : listeIdAllConf_)
          {
               similarite = ucs(id);
               if (similarite>=0)
               {
                    Map<String, Object> conf =
DbDAO.getFeaturesNextPack(numSousConfSuivant, id);
                    while (recommanded.containsKey(similarite))
                         similarite+=0.0000001;
                    recommanded.put(similarite, conf);
                    System.out.println("ID: "+id+"; similarité:
"+similarite);
               }
          }

          System.out.println();
```

```
        for (Float f : recommanded.keySet())
        {
            liste.add(recommanded.get(f));
        }

        return liste;
    }

    public void initReco(final Map<String, Object> confCl)
    {
        listeIdAllConf_ = DbDAO.getlisteIdAllConf();


    for(String feature : confCl.keySet())
        {
            Feature f = new Feature();
            f.table_ = ParamsSousConf.getNomTable(feature);
            f.nom_ = feature;

            if(confCl.get(feature) instanceof String)
            {
                f.valeur_ = (String)confCl.get(feature);
                listeFeatures.add(f);
            }
            else if(confCl.get(feature) instanceof List)
            {
                List l = (List)confCl.get(feature);
                for(Object o : l)
                {
                    f.valeur_=(String)o;
                    listeFeatures.add(f);

                }
            }

        }
    }
```

Algorithme 2. Algorithme de calcul de similarité

```
    public float ucs(int id)
    {
        float ucs1 = 0;
        for (int i=0;i<listeFeatures.size();i++)
            ucs1+=somme(i,id);
        return ucs1;
    }

    public double wc(int i)
    {
```

```
            Feature f = listeFeatures.get(i);
            return poidsClient(f.valeur_, f.nom_, f.table_);
      }

      public double ws(int i, int id)
      {
            Feature f = listeFeatures.get(i);
            return poidsFeature(f.valeur_, f.nom_, f.table_ ,id);
      }

      public double poidsFeature(String valeurFeature, String
nomFeature, String nomTable, int idConf){
            double nb1= DbDAO.nbOccValeurDansConfX(valeurFeature,
nomFeature, nomTable, idConf);
            double nb2=
DbDAO.nbConfTermeIaumoins1eFois(valeurFeature, nomFeature,
nomTable);
            if(nb1==0 || nb2==0)
                  return 0;
            else
                  return nb1 *
Math.log(listeIdAllConf_.size()/nb2);
      }

      private float somme (int i, int id)
      {
            float somme = 0;
            somme = (float) (( wc(i)/Math.sqrt(sousSomme1()))  *
( ws(i,id) /Math.sqrt(sousSomme2(id))));
            return somme;
      }

      private float sousSomme1()
      {
            float sousSomme = 0;

            for (int i=0;i<listeFeatures.size();i++)
                  sousSomme+=Math.pow(wc(i), 2)  ;
            return sousSomme;
      }

      private float sousSomme2(int id)
      {
            float sousSomme = 0;
            for (int i=0;i<listeFeatures.size();i++)
                  sousSomme+=Math.pow(ws(i,id), 2)  ;
            return sousSomme;
      }

      private double poidsClient (String valeurFeature, String
nomFeature, String nomTable){
```

```
          int nb1 = 0;
          for (Feature f : listeFeatures)
               if (f.valeur_ == valeurFeature && f.nom_ ==
nomFeature)
                    nb1++;

          double nb2=
DbDAO.nbConfTermeIaumoins1eFois(valeurFeature, nomFeature,
nomTable);
          if(nb1==0 || nb2==0)
               return 0;
          else
               return  nb1 *
Math.log(listeIdAllConf_.size()/nb2);
     }
```

Algorithme 3. Algorithme de vérification de contraintes

```
     public static boolean
verifContraintesSousConf0(Map<String, Object> confCl)
      {
           nbrePieces = (String) confCl.get("nbrePieces");
           departement = (String) confCl.get("departement");
           typeLog = (String) confCl.get("typeLog");
           surfaceLog = (String) confCl.get("surfaceLog");
           surfaceCui = (String) confCl.get("surfaceCui");
           surfaceSej = (String) confCl.get("surfaceSej");
           marque = (String) confCl.get("surfaceSej");
           technologie = (String) confCl.get("technologie");
           circuitEclairage = (String)
confCl.get("circuitEclairage");
           typeCircuitS = (String) confCl.get("typeCircuitS");
           nbreSocles = (String) confCl.get("nbreSocles");


           if(!(Integer.valueOf(surfaceLog)>=35 &&
Integer.valueOf(circuitEclairage)==2))
                {
                   JOptionPane.showMessageDialog(null,
"Contrainte non respectée: Le nombre de circuits d'éclairage
doit au moins être égale à deux.");
                   return false;
                }


          if(!(typeLog.equals("T1NonEquipé") &&
(typeCircuitS.equals("laveLinge")||typeCircuitS.equals("laveVa
isselle")||typeCircuitS.equals("secheLinge")||typeCircuitS.equ
als("four")||typeCircuitS.equals("congelateur"))))
```

```
                {
                   JOptionPane.showMessageDialog(null,
"Contrainte TC0014 non respectée");
                   return false;
                }



     if(!(typeCircuitS.equals("circuitChauffeEau"))||(typeCircu
itS.equals("circuitChauffage"))||(typeCircuitS.equals("circuit
VMC"))||(typeCircuitS.equals("circuitVoletsRoulants")))
                {
                   JOptionPane.showMessageDialog(null,
"Contrainte TC0020 non resepectée");
                   return false;
                }

           return true;
      }

      public static boolean
verifContraintesSousConf1(Map<String,Object> sousConf)
      {
           if(!verifContraintesSousConf0(sousConf))
                return false;

           rangee = (String) sousConf.get("rangee");
           moduleParRangee = (String)
sousConf.get("moduleParRangee");

           return true;
      }

      public static boolean
verifContraintesSousConf2(Map<String,Object> sousConf)
      {
           boolean test = verifContraintesSousConf1(sousConf);

           if(!test)
                return false;

           rehausse = (String) sousConf.get("rehausse");
           typeCoffret = (String) sousConf.get("typeCoffret");
           typePorte = (String) sousConf.get("typePorte");


           return true;
      }

      public static boolean
verifContraintesSousConf3(Map<String,Object> sousConf)
```

```
      {

            boolean test = verifContraintesSousConf2(sousConf);

            if(!test)
                  return false;

            nbreModParafoudre = (String)
sousConf.get("nbreModParafoudre");


            Set<String> listeDepNivKanInf25 = new
HashSet<String>(Arrays.asList("01", "04", "05", "06", "07",
"13", "2A", "2B", "24", "25", "26", "30", "33", "34", "38",
"39", "40", "42", "43", "47", "48", "63", "66", "69", "71",
"73", "74", "83", "84", "971", "972", "973"));

            if (!(listeDepNivKanInf25.contains(departement) &&
nbreModParafoudre.equalsIgnoreCase("1")))
            {
                  JOptionPane.showMessageDialog(null, "Contrainte
non resepectée: Le niveau kéraunique de votre département est
supérieur à 25. "
                              + "Il faut installer un parafoudre!");
                return false;
            }

            return true;
      }

      public static boolean
verifContraintesSousConf4(Map<String,Object> sousConf)
      {
            boolean test = verifContraintesSousConf3(sousConf);

            if(!test)
                  return false;

            nbreModTelerupteur = (String)
sousConf.get("nbreModTelerupteur");
            nbreModContacteur = (String)
sousConf.get("nbreModContacteur");
            nbreModMinuterie = (String)
sousConf.get("nbreModMinuterie");
            nbreModInterrupteurHoraire = (String)
sousConf.get("nbreModInterrupteurHoraire");
            nbreModTelevariateur = (String)
sousConf.get("nbreModTelevariateur");
            nbreModTemporisateur = (String)
sousConf.get("nbreModTemporisateur");
```

```
            nbreModGestionFilPilote = (String)
sousConf.get("nbreModGestionFilPilote");


            return true;
      }

      public static boolean
verifContraintesSousConf5(Map<String,Object> sousConf)
      {
            boolean test = verifContraintesSousConf4(sousConf);

            if(!test)
                  return false;

            typeInterDiff = (List<String>)
sousConf.get("typeInterDiff");
            calibreInterDiff = (List<String>)
sousConf.get("calibreInterDiff");
            int count = 0;


            if (Integer.valueOf(surfaceLog)<=35)
                  for (int i = 0; i<typeInterDiff.size();i++)
                {
                        String v1 = typeInterDiff.get(i);
                        String v2 = calibreInterDiff.get(i);

                  if (v1.equalsIgnoreCase("idac") &&
v2.equalsIgnoreCase("id_25"))
                        count ++;
                }

            if (count==0 && Integer.valueOf(surfaceLog)<=35)
            {
                  JOptionPane.showMessageDialog(null, "Contrainte
TC0042 non resepectée: Pour un logement de surface inférieure
ou égale à 35m², au moins un interrupteur différentiel 30mA de
courant assigné 25A et de type AC doit être prévu.");
                  return false;
            }


            if (Integer.valueOf(surfaceLog)>35 &&
Integer.valueOf(surfaceLog)<=100 )
                  for (int i = 0; i<typeInterDiff.size();i++)
                {
                  if
(typeInterDiff.get(i).equalsIgnoreCase("idac") &&
calibreInterDiff.get(i).equalsIgnoreCase("id_40"))
                        count ++;
```

```
            }

          if (count<2 && Integer.valueOf(surfaceLog)>35 &&
Integer.valueOf(surfaceLog)<=100)
          {
               JOptionPane.showMessageDialog(null, "Contrainte
TC0043 non resepectée: Pour un logement de surface supérieure
à 35m² et inférieure ou égale à 100m², au moins deux
interrupteurs différentiels 30mA de courant assigné 40A et de
type AC doivent être prévus.");
               return false;
          }

          if (Integer.valueOf(surfaceLog)>100)
               for (int i = 0; i<typeInterDiff.size();i++)
             {
               if
(typeInterDiff.get(i).equalsIgnoreCase("idac") &&
calibreInterDiff.get(i).equalsIgnoreCase("id_40"))
                    count ++;
             }

          if (count<3 && Integer.valueOf(surfaceLog)> 100)
          {
               JOptionPane.showMessageDialog(null, "Contrainte
TC0044 non resepectée: Pour un logement de surface supérieure
à 100m², au moins trois interrupteurs différentiels 30mA de
courant assigné 40A et de type AC doivent être prévus.");

               return false;
          }
          return true;
     }

     public static boolean
verifContraintesSousConf6(Map<String,Object> sousConf)
     {
          boolean test = verifContraintesSousConf5(sousConf);

          if(!test)
               return false;

          typeDisj = (List<String>) sousConf.get("typeDisj");
          calibreDisj = (List<String>)
sousConf.get("calibreDisj");
          int count1 = 0;
          int count2 = 0;
          int nbr=0;


          if(typeLog=="T1NonEquipé" )
```

```
                for (int i = 0; i<calibreDisj.size();i++)
                    {
                      if
(calibreDisj.get(i).equalsIgnoreCase("d_32"))
                            count1 ++;

                      if
(calibreDisj.get(i).equalsIgnoreCase("d_16"))
                            count2 ++;

                    }

           if ( (count1<1 || count2<2) &&
typeLog=="T1NonEquipé")
                {
                     JOptionPane.showMessageDialog(null,
"Contrainte TC0019 non resepectée: Dans le cas particulier des
logements de type T1, et si l'équipement n'est pas fourni,
trois circuits spécialisés au moins sont prévus (un circuit
32A et deux circuits 16A)");
                    return false;
                }


           if(typeCircuitS.equals("circuitCuisson") ||
typeCircuitS.equals("laveLinge") ||
typeCircuitS.equals("cfour") ||
typeCircuitS.equals("laveVaisselle") ||
typeCircuitS.equals("secheLinge") ||
typeCircuitS.equals("congelateur") ||
typeCircuitS.equals("circuitVoletsRoulants") ||
typeCircuitS.equals("circuitChauffage") ||
typeCircuitS.equals("circuitVMC") ||
typeCircuitS.equals("circuitChauffeEau") ||
Integer.valueOf(circuitEclairage)!=0 ||
Integer.valueOf(nbreSocles)!=0 )
                nbr=calibreDisj.size();
           if(nbr==0 && (typeCircuitS.equals("circuitCuisson")
|| typeCircuitS.equals("laveLinge") ||
typeCircuitS.equals("four") ||
typeCircuitS.equals("laveVaisselle") ||
typeCircuitS.equals("secheLinge") ||
typeCircuitS.equals("congelateur") ||
typeCircuitS.equals("circuitVoletsRoulants") ||
typeCircuitS.equals("circuitChauffage") ||
typeCircuitS.equals("circuitVMC") ||
typeCircuitS.equals("circuitChauffeEau") ||
Integer.valueOf(circuitEclairage)!=0 ||
Integer.valueOf(nbreSocles)!=0 ))
           {
```

```
                JOptionPane.showMessageDialog(null, "Contrainte
TC0055 non resepectée: Tout circuit doit être protégé par un
dispositif de protection qui est un disjoncteur.");
                return false;
           }


           if(typeCircuitS.equals("circuitVoletsRoulants") ||
typeCircuitS.equals("circuitVMC") ||
Integer.valueOf(circuitEclairage)!=0 ||
Integer.valueOf(nbreSocles)!=0)
                for (int i=0; i<calibreDisj.size(); i++)
                {

if(calibreDisj.get(i).equalsIgnoreCase("d_16"))
                          count1++;
                }
           if(count1==0 &&
(typeCircuitS.equals("circuitVoletsRoulants") ||
Integer.valueOf(circuitEclairage)!=0 ||
Integer.valueOf(nbreSocles)!=0))
           {
                JOptionPane.showMessageDialog(null, "Contrainte
TC0056 && TC0057 non resepectées: Les circuits d'éclairage, de
volets roulants, de prises commandées, et les circuits de VMC
doivent être protégés par un disjoncteur de courant assigné
maximal de 16A.");
                return false;
           }


           if(typeCircuitS.equals("circuitChauffeEau"))
                for (int i=0; i<calibreDisj.size(); i++)
                {

if(calibreDisj.get(i).equalsIgnoreCase("d_20"))
                          count1++;
                }
           if(count1==0 &&
typeCircuitS.equals("circuitChauffeEau"))
           {
                JOptionPane.showMessageDialog(null, "Contrainte
TC0059 non resepectée: Les circuits de chauffe-eau électrique
non instantané doivent être protégés par un disjoncteur de
courant assigné maximal de 20A.");
                return false;
           }

           return true;

      }
```

Algorithme 4. Algorithme de division des caractéristiques de la LP en paquets de caractéristiques

```
 public static int NB_SOUS_CFG = 9;
     public static List<List<String>> PARAM_SOUS_CONFIGS = new
ArrayList<List<String>>();

     private static String [] tables_ = {"logement", "coffret",
"interDiff", "disjoncteur","fusible" ,"disjDiff",
"peigneHorizontal","circuitSpecialise", "circuitPc16a"};

     static {

          for (int i= 0;i<NB_SOUS_CFG;i++)
         {
          switch (i)
          {
               case 0 :{
                 String [] t =
{"nbrePieces","departement","typeLog","surfaceLog","surfaceCui
", "surfaceSej",
"marque","technologie","circuitEclairage","typeCircuitS","nbre
Socles"};
                 PARAM_SOUS_CONFIGS.add(Arrays.asList(t));
                 break;
                }
                case 1 :{
               String [] t = {"rangee","moduleParRangee"};
               PARAM_SOUS_CONFIGS.add(Arrays.asList(t));
               break;
             }
             case 2 :{
                String [] t =
{"rehausse","typeCoffret","typePorte"};
                 PARAM_SOUS_CONFIGS.add(Arrays.asList(t));
                 break;
             }
             case 3 :{
                String [] t = {"nbreModParafoudre"};
                  PARAM_SOUS_CONFIGS.add(Arrays.asList(t));
                 break;
             }
             case 4 :{
                 String [] t =
{"nbreModTelerupteur","nbreModContacteur","nbreModMinuterie","
nbreModInterrupteurHoraire","nbreModTelevariateur","nbreModTem
porisateur","nbreModGestionFilPilote"};
                 PARAM_SOUS_CONFIGS.add(Arrays.asList(t));
                 break;
             }
             case 5 :{
```

```
                String [] t =
{"typeInterDiff","calibreInterDiff"};
                 PARAM_SOUS_CONFIGS.add(Arrays.asList(t));
               break;
             }
             case 6 :{
                String [] t = {"typeDisj","calibreDisj"};
                 PARAM_SOUS_CONFIGS.add(Arrays.asList(t));
                 break;
             }
             case 7 :{
                 String [] t =
{"typeDisjDiff","calibreDisjDiff"};
                 PARAM_SOUS_CONFIGS.add(Arrays.asList(t));
                 break;
             }
             case 8 :{
                String [] t = {"typeFusible","calibreFusible"};
                 PARAM_SOUS_CONFIGS.add(Arrays.asList(t));
                 break;
             }


          }
        }
     }

     public static String getNomTable(String nomFeature)
     {
          if (PARAM_SOUS_CONFIGS.get(6).contains(nomFeature))
               return tables_[3];
          else if
(PARAM_SOUS_CONFIGS.get(5).contains(nomFeature))
               return tables_[2];
          else if
(PARAM_SOUS_CONFIGS.get(8).contains(nomFeature))
               return tables_[4];
          else if
(PARAM_SOUS_CONFIGS.get(7).contains(nomFeature))
               return tables_[5];
          else if
(PARAM_SOUS_CONFIGS.get(0).contains(nomFeature))
          {
               switch (nomFeature)
               {

                 case "typeCircuitS"  : return tables_[7];
                 case "nbreSocles" : return tables_[8];

                 case "circuitEclairage" :
                 case "technologie" :
```

```
                case "marque": return tables_[1];

                default: return tables_[0];
              }
          }


          return tables_[1];
     }


     public static int identifierSousConf(final Map<String,
Object> confCl)
     {
          List<String> nomfeaturesConfCL = new
ArrayList<String>();
          for (String key : confCl.keySet())
                    nomfeaturesConfCL.add(key);

          int lastSousconf = -1;

          for (int i = 0; i<ParamsSousConf.NB_SOUS_CFG;i++)
               if
(nomfeaturesConfCL.containsAll(ParamsSousConf.PARAM_SOUS_CONFI
GS.get(i)))
                    lastSousconf = i;

          return lastSousconf;
     }
```

Algorithme 5. Algorithme de configuration (recommandation+vérification de contraintes) sur l'étape 3

```
List<JPanel> listePanneaux_;
private List<Map<String, Object> > listeConfs_ ;
private final AtomicInteger etapeCourante = new
AtomicInteger(0);
Map<String, Object> confCl = new HashMap<String, Object>();

if (etapeCourante.get()==3)
          {
            if (listeConfs_ != null)
                 listeConfs_.clear();


((Panneau2)listePanneaux_.get(2)).ecrireConfUser(confCl);
            Contraintes.verifContraintesSousConf2(confCl);

            listeConfs_ = reco_.recommander(confCl);
            ((Panneau3)listePanneaux_.get(3)).rafraichir(true,
listeConfs_);   }
```

L'implémentation du logiciel DynaElec est présentée ci-dessous.

Algorithme 6. Le DAO (Data Access Object) et le développement du buffer

```
public class DbDAO {

     private static final Logger LOGGER =
Logger.getLogger(DbDAO.class);

     //définition d'un buffer pour améliorer le temps de calcul
de poids
     static Map<String, Integer> buffer_= new
HashMap<String,Integer>();

     public static List<Integer> getlisteIdAllConf() {
          Connection conn = null;
          PreparedStatement ps = null;
          ResultSet rs = null;
          List<Integer> listeId = new ArrayList<Integer>();

          try {

               conn = RecoConnexionFactory.creerConnexion();
               String sql = "select idConf from logement";
               ps = conn.prepareStatement(sql);
               rs = ps.executeQuery();

               while (rs.next()) {
                    listeId.add(
Integer.parseInt((rs.getString(1))));
               }
          } catch (InstantiationException e) {
               LOGGER.error(e.getStackTrace(), e);
          } catch (IllegalAccessException e) {
               LOGGER.error(e.getStackTrace(), e);
          } catch (ClassNotFoundException e) {
               LOGGER.error(e.getStackTrace(), e);
          } catch (SQLException e) {
               LOGGER.error(e.getStackTrace(), e);
          }

          finally {
               try {
                    if (rs != null)   rs.close();
                    if (ps != null)   ps.close();
                    if (conn != null) conn.close();
               }
```

```
                catch (SQLException e) {
LOGGER.error(e.getStackTrace(), e); }

          }
          return listeId;
     }

     public static int nbOccValeurDansConfX(String
valeurFeature, String nomFeature, String nomTable, int idConf)
{
          String sql = "select COUNT(*) from "+nomTable+" where
" + nomFeature+ "='" + valeurFeature + "'and idConf=" +
idConf;
          if (buffer_.containsKey(sql))
          {
               return buffer_.get(sql);
          }

          Connection conn = null;
          PreparedStatement ps = null;
          ResultSet rs = null;
          int rowCount = 0;
          try {

               conn = RecoConnexionFactory.creerConnexion();
               ps = conn.prepareStatement(sql);
               rs = ps.executeQuery();

               while (rs.next()) {
                    rowCount =
Integer.parseInt((rs.getString(1)));
               }

          } catch (InstantiationException e) {
               LOGGER.error(e.getStackTrace(), e);
          } catch (IllegalAccessException e) {
               LOGGER.error(e.getStackTrace(), e);
          } catch (ClassNotFoundException e) {
               LOGGER.error(e.getStackTrace(), e);
          } catch (SQLException e) {
               LOGGER.error(e.getStackTrace(), e);
          }

          finally {
               try {
                         if (rs != null)   rs.close();
                         if (ps != null)   ps.close();
                         if (conn != null) conn.close();
                    }
                    catch (SQLException e) {
LOGGER.error(e.getStackTrace(), e); }
```

```
          }

          buffer_.put(sql, rowCount);
          return rowCount;

     }

     public static int nbConfTermeIaumoins1eFois(String
valeurFeature, String nomFeature, String nomTable) {
          String sql = "select COUNT(distinct idConf) from
"+nomTable+" where " + nomFeature+ "='" + valeurFeature + "'";
          if (buffer_.containsKey(sql))
          {
               return buffer_.get(sql);
          }

          Connection conn = null;
          PreparedStatement ps = null;
          ResultSet rs = null;
          int rowCount = 0;

          try {

               conn = RecoConnexionFactory.creerConnexion();
               ps = conn.prepareStatement(sql);
               rs = ps.executeQuery();

               while (rs.next()) {
                    rowCount =
Integer.parseInt((rs.getString(1)));

               }
          } catch (InstantiationException e) {
               LOGGER.error(e.getStackTrace(), e);
          } catch (IllegalAccessException e) {
               LOGGER.error(e.getStackTrace(), e);
          } catch (ClassNotFoundException e) {
               LOGGER.error(e.getStackTrace(), e);
          } catch (SQLException e) {
               LOGGER.error(e.getStackTrace(), e);
          }

          finally {
               try {
                    if (rs != null)   rs.close();
                    if (ps != null)   ps.close();
                    if (conn != null) conn.close();
               }
               catch (SQLException e) {
LOGGER.error(e.getStackTrace(), e); }
```

```
          }
          buffer_.put(sql, rowCount);
          return rowCount;
     }

     public static Map<String,Object> getFeaturesNextPack (int
numSousConf, int idConf)
     {
          Connection conn = null;
          PreparedStatement ps = null;
          ResultSet rs = null;
          Map<String,Object> lg = new HashMap<String,
Object>();
          List <String>
L=ParamsSousConf.PARAM_SOUS_CONFIGS.get(numSousConf);
          String sqlFeatures= "";
          String sqlTable="";


          for(int i=0; i<L.size(); i++)
               {
                    sqlFeatures+=L.get(i);
                    sqlFeatures+=",";

     sqlTable=ParamsSousConf.getNomTable(L.get(i));
               }

          int index = sqlFeatures.length()-1;
          sqlFeatures = sqlFeatures.substring(0, index);

          try {

               conn = RecoConnexionFactory.creerConnexion();
               String sql = "select "+sqlFeatures+" from
"+sqlTable+" where idConf = "+idConf;
               ps = conn.prepareStatement(sql);
               rs = ps.executeQuery();

               if (numSousConf >= 5 && numSousConf <= 8) {
                    for (int i = 0; i < L.size(); i++) {
                         String feature = L.get(i);
                         List<String> vals = new
ArrayList<String>();
                         while (rs.next())
                         {
                              vals.add(rs.getString(feature));
                         }
                         rs = ps.executeQuery();
                         for (String s : vals)

                         lg.put(feature, vals);
```

```
                    }
               }
               else
                    while (rs.next()) {
                         for (int i = 0; i < L.size(); i++) {
                              String feature = L.get(i);
                              lg.put(feature,
rs.getString(feature));
                         }
               }

          } catch (InstantiationException e) {
               LOGGER.error(e.getStackTrace(), e);
          } catch (IllegalAccessException e) {
               LOGGER.error(e.getStackTrace(), e);
          } catch (ClassNotFoundException e) {
               LOGGER.error(e.getStackTrace(), e);
          } catch (SQLException e) {
               LOGGER.error(e.getStackTrace(), e);
          }

          finally {
               try {
                         if (rs != null)   rs.close();
                         if (ps != null)   ps.close();
                         if (conn != null) conn.close();
                    }
                    catch (SQLException e) {
LOGGER.error(e.getStackTrace(), e); }
          }
          return lg;
     }
}
```

Algorithme 7.La première IHM de DynaElec (premier paquet)

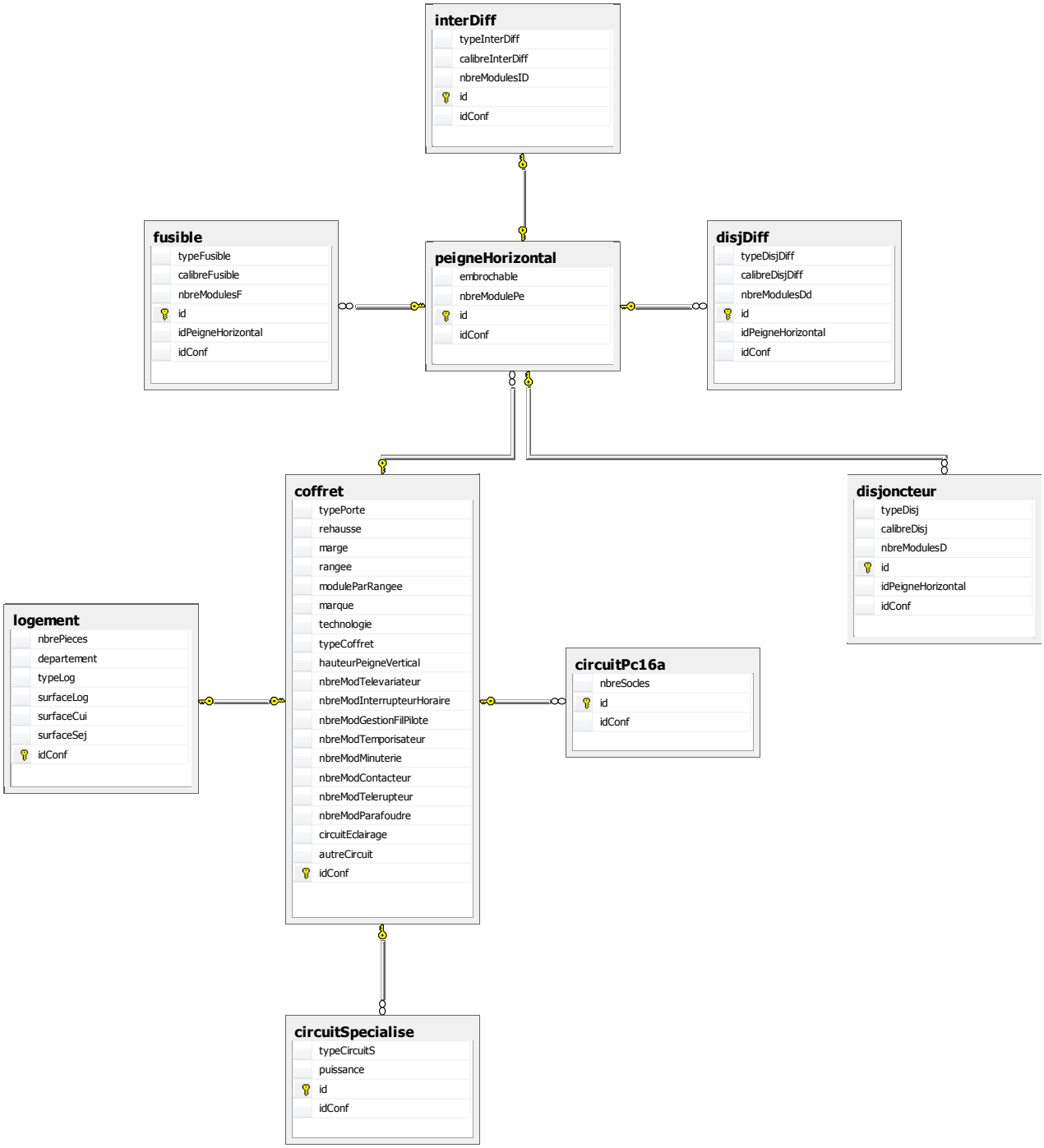
```
public class Panneau0 extends JPanel {

     private JComboBox<String> comboMarques_;
     private JComboBox<String> comboTechno_;
     private JComboBox<String> comboTypeLog_;
     private JComboBox<String> comboDepartement_;
     private JComboBox<String> comboEquipCuisine_;

     private JTextField txtNbrePiece = new JTextField();
     private JTextField txtSurfaceLogement = new JTextField();
     private JTextField txtSurfaceCuisine = new JTextField();
     private JTextField txtSurfaceSejour = new JTextField();
```

```
    private JTextField txtCircuitPC16A = new JTextField();
    private JTextField txtCircuitEclairage = new JTextField();

    private String[] equipCuisineValues = { "Neant",
"circuitCuisson", "laveLinge",
              "secheLinge", "four", "laveVaisselle",
"congelateur" };

    public Panneau0() {
        super(new GridBagLayout());

        String[] marques = { "Schneider", "Legrand", "Hager"
};
        comboMarques_ = new JComboBox<String>(marques);

        String[] technologies = { "Automatique", "Manuelle"
};
        comboTechno_ = new JComboBox<String>(technologies);

        String[] typeLog = { "T1Equipé", "T1NonEquipé",
"Autre" };
        comboTypeLog_ = new JComboBox<String>(typeLog);

        String[] departement = { "01", "02", "03", "04",
"05", "06", "07", "08" };
        comboDepartement_ = new
JComboBox<String>(departement);

        String[] EquipCuisine = { "Neant", "table de
cuisson", "lave linge",
                  "sèche linge", "four", "lave vaisselle",
"congelateur" };


        comboEquipCuisine_ = new
JComboBox<String>(EquipCuisine);

        DialogUtils.sameSizeComponents(comboMarques_,
comboTechno_,
                  comboTypeLog_, comboDepartement_,
comboEquipCuisine_,
                  txtNbrePiece, txtSurfaceLogement,
txtSurfaceCuisine,
                  txtSurfaceSejour, txtCircuitPC16A,
txtCircuitEclairage);

        GridBagConstraints c = new GridBagConstraints();
        c.insets = new Insets(0, DialogUtils.ESPACE,
DialogUtils.ESPACE,
                  DialogUtils.ESPACE);
        c.anchor = GridBagConstraints.LINE_START;
```

```
c.gridx = 0;
c.gridy = 0;
add(new JLabel("Marque"), c);
c.gridx = 1;
c.gridy = 0;
add(comboMarques_, c);

c.gridx = 0;
c.gridy++;
add(new JLabel("Technologie"), c);
c.gridx = 1;
add(comboTechno_, c);

c.gridx = 0;
c.gridy++;
add(new JLabel("Type logement"), c);
c.gridx = 1;
add(comboTypeLog_, c);

c.gridx = 0;
c.gridy++;
add(new JLabel("Nombre de pièces"), c);
c.gridx = 1;
add(txtNbrePiece, c);

c.gridx = 0;
c.gridy++;
add(new JLabel("Surface logement"), c);
c.gridx = 1;
add(txtSurfaceLogement, c);

c.gridx = 0;
c.gridy++;
add(new JLabel("Departement"), c);
c.gridx = 1;
add(comboDepartement_, c);

c.gridx = 0;
c.gridy++;
add(new JLabel("Surface cuisine"), c);
c.gridx = 1;
add(txtSurfaceCuisine, c);

c.gridx = 0;
c.gridy++;
add(new JLabel("Equipement cuisine"), c);
c.gridx = 1;
add(comboEquipCuisine_, c);

c.gridx = 0;
```

```
            c.gridy++;
            add(new JLabel("Surface sejour"), c);
            c.gridx = 1;
            add(txtSurfaceSejour, c);

            c.gridx = 0;
            c.gridy++;
            add(new JLabel("Nombre de socles"), c);
            c.gridx = 1;
            add(txtCircuitPC16A, c);

            c.gridx = 0;
            c.gridy++;
            add(new JLabel("Circuit éclairage"), c);
            c.gridx = 1;
            add(txtCircuitEclairage, c);
      }

      public void ecrireConfUser(Map<String, Object> confCl) {
            confCl.put("nbrePieces", txtNbrePiece.getText());
            confCl.put("departement", (String)
comboDepartement_.getSelectedItem());
            confCl.put("typeLog", (String)
comboTypeLog_.getSelectedItem());
            confCl.put("surfaceLog",
txtSurfaceLogement.getText());
            confCl.put("surfaceCui",
txtSurfaceCuisine.getText());
            confCl.put("surfaceSej", txtSurfaceSejour.getText());

            confCl.put("marque", (String)
comboMarques_.getSelectedItem());
            confCl.put("technologie", (String)
comboTechno_.getSelectedItem());
            confCl.put("circuitEclairage",
txtCircuitEclairage.getText());


            confCl.put("typeCircuitS",
      equipCuisineValues[comboEquipCuisine_.getSelectedIndex()])
;


            confCl.put("nbreSocles", txtCircuitPC16A.getText());
      }


      }
```

Schéma 1. Schéma de la BD



Références bibliographiques

(Aamodt & Plaza, 1994): Aamodt, A., Plaza, E. (1994).“Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches”. AI Communications 7(1):39-52.

(Adomavicius & Tuzhilin, 2005): Adomavicius, G., Tuzhilin, A. (2005). “Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions”. IEEE Trans. Knowl. Data Eng. 17(6), 734.

(Andersson & Runeson, 2007): Andersson, C., Runeson, P. (2007). “A replicated quantitative analysis of fault distributions in complex software systems”. IEEE Transactions on Software Engineering, 33(5):273–286.

(Andrey et al., 2006) : Andrey, L., Festor, O., Lahmadi, A. (2006). “Evaluation du passage a l’échelle des systèmes de gestion : métriques et modèles”. Colloque Francophone sur l'Ingénierie des Protocoles. Tozeur/Tunisia, Hermes.

(Apt, 2003): Apt, K. (2003). “Principles of Constraint Programming”. Cambridge University Press.

(Astesana et al. 2010): Astesana, J.M., Cosserat, L.,Fargier, H. (2010). “Constraint-based vehicle configuration: a case study”. In International Conference on Tools with Artificial Intelligence (ICTAI), IEEE Computer Society. Arras.

(Atkinson et al., 2001): Atkinson, C., Bayer, J.,Bunse, C.,Kamsties, E.,Laitenberger, O., Laqua, R., Muthig, D., Paech, B., Wüst, J., Zettel, J. (2001). “ Component-Based Product Line Engineering with UML”. Component Software Series, Addison-Wesley.

(Balabanovic&Shoham, 1997): Balabanovic, M., Shoham Y. (1997). “Combining Content-based and collaborativerecommendation”. Communications of the ACM, 40 (3).

(Bell et al., 2007) :Bell, R., Koren, Y., Volinsky, C. (2007b). “Modeling relationships at multiple scales to improve accuracy of large recommender systems”. In 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 95–104, New York, NY, USA, ACM.

Bell, R., Koren, Y. (2007c). “Lessons from the Netflix Prize Challenge”, SIGKDD Explorations 9, 75–79.

(Benavides et al., 2006) : Benavides, D., Segura, S., Trinidad, P., Ruiz-Cortés, A. (2006).« A first step towards a framework for the automated analysis of feature models”. In Managing Variability for Software Product Lines: Working With Variability Mechanisms.

(Benavides et al., 2005) : Benavides, D., Ruiz-Cortés, A., Trinidad, P. (2005). "Using constraint programming to reason on feature models". In The Seventeenth International Conference on Software Engineering and Knowledge Engineering, pages 677–682.

(Bishop, 2006): Bishop, C.M. (ed.) (2006)."Pattern Recognition and machine learning". Berlin : Springer.

(Bonnin, 2010) : Bonnin, G. (2010). " Vers des systèmes de recommandation robustes pour la navigation Web: inspiration de la modélisation statistique du langage ". Thèse, Université Nancy 2.

(Boucher et al., 2010) : Boucher, Q., Classen, A., Faber, P., Heymans, P. (2010). " Introducing TVL, a text-based feature modeling language". In Variability Modeling of Software-Intensive Systems (VaMoS), pages 159-162.

(Breese et al., 1998): Breese, J. S., Heckerman, D., Kadie, C. (1998). "Empirical analysis of predictive algorithms for collaborative filtering". In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Madison, WI.

(Burke, 2007) : Burke, R. (2007). "Hybrid Web Recommender Systems". The Adaptive Web 2007, pages 377-408.

(Burke, 2002) : Burke, R. (2002). "Hybrid recommender systems: survey and experiments". User Modeling and User-Adapted Interaction, 12(4), pages 331–370.

(Burke, 2000) : Burke, R. (2000). "Knowledge-based recommender systems". Library and Information Systems, 69(32), pages180–200.

(Burke & Ramezani, 2011): Burke, R., Ramezani, M.(2011). "Matching Recommendation Technologies and Domains". In Ricci F, Rokach L, Shapira, B. (eds) Recommender Systems Handbook,pages 367-386, Springer US.

(Chickering et al., 1997): Chickering, D., Heckerman, D., Meek, C. (1997). "A Bayesian approach to learning Bayesian networks with local structure". In Proceedings of Thirteenth Conference on Uncertainty in Artificial Intelligence, Providence, RI. Morgan Kaufmann.

(Classen et al., 2011) : Classen, A., Boucher, Q., Heymans, P. (2011). "A text-based Approach to Feature Modeling: Syntax and semantics of TVL", Science of Computer Programming, Special Issue on Software Evolution 76 (12), pages 1130–1143.

(Classen et al., 2009): Classen, A., Hubaux, A., Heymans, P. (2009). " A formal semantics for multi-level staged configuration". In Proceedings of the Third International Workshop on Variability Modeling of Software-intensive Systems (VaMoS'09), Sevilla, Spain.

(Clauss, 2001): Clauss, M. (2001). “Generic modeling using UML extensions for variability”. OOPSLA 2001 Workshop on Domain Specific Visual Languages.

(Cohen & Hirsh, 1998): Cohen, W., Hirsh, H. (1998). “Joins that Generalize: Text Classification Using WHIRL”. In Proceedings of the Fourth International Conference on Knowledge Discovery & Data Mining, NewYork, pages 169-173.

(Cook & Campbell, 1979): Cook, T.D., Campbell, D.T.(1979).“Quasi-experimentation – Design and Analysis Issues for Field Settings”. Houghton Mifflin Company, Boston.

(Cöster et al., 2002): Coester, R., Gustavsson, A., Olsson, R. Rudstroem, A. (2002). “Enhancing web-based configuration with recommendations and cluster-based help”.In AH’02 Worksh. On Recommendation and Personalization in EComm., Malaga, Spain.

(Czarnecki et al., 2005) :Czarnecki, K.,Helsen, S.,Eisenecker,U. (2005a). “Staged configuration through specialization and multi-level configuration of feature models”. Software Process Improvement and Practice, 10(2).

Czarnecki, K.,Helsen, S.,Eisenecker,U.W. (2005c). “Formalizing cardinality-based feature models and their specialization”. Software Process: Improvement and Practice, 10(1), pages 7–29.

(Czarnecki et al., 2004): Czarnecki, K.,Helsen, S.,Eisenecker, U. (2004). “Staged configuration using feature models”. In Proceedings of the Third Software Product Line Conference, volume 3154 of Lecture Notes in Computer Sciences, pages 266–282. Springer–Verlag.

(Czarnecki, 1998) : Czarnecki,K. (1998). “Generative Programming: Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models”. Ph.D. Thesis, Computer Science Department, Technical University of Ilmenau, Ilmanau, Germany.

(Dauron & Astesana, 2010): Dauron A., Astesana J-M. (2010). “ Spécification et configuration de la ligne de produits véhicule de Renault ”. Journée Lignes de Produits. Université Pantéon Sorbonne, Paris, France.

(Deelstra et al., 2005): Deelstra, S., Sinnema, M., Bosch, J. (2005). “Product derivation in software product families: a case study”. Journal of Systems and Software, volume 74, issue 2, pages 173-194.

(Djebbi, 2011): Djebbi, O. (2011). “L’ingénierie des exigencies par et pour les lignes des produits”. Thèse: Université Paris 1 Panthéon-Sorbonne.

(Duda et al., 2001): Duda, R. O., Hart,P. E., Stork, D. G. (2001). “Pattern Classification”. John Wiley & Sons.

(Falkner et al., 2011): . Falkner, A., Felfernig, A., Haag, A.(2011). “Recommendation Technologies for Configurable Products”. AI Magazine 32(3), pages 99-108.

(Felfernig, 2014): Felfernig, A. (2014). “Biases in decision making”. In International workshop on decision making and recommender systems, volume 1278, pages 32-37. CEUR Proceedings.

(Felfernig et al., 2014): Felfernig, A., Jeran, M., Ninaus, G., Reinfrank, F., Reiterer, S., Stettinger, M. (2014). “Basic approaches in recommendation systems”. In Recommendation Systems in Software Engineering, pages 15–37, Springer.

(Felfernig & Burke, 2008): Felfernig, A., Burke, R. (2008). “Constraint-based recommender systems: technologies and research issues”. In Proceedings of the 10th international conference on electronic commerce (ISEC08).

(Felfernig et al., 2007) : Felfernig, A., Isak, K., Szabo, K., Zachar, P. (2007). “The VITA Financial Services Sales Support Environment”. AAAI/IAAI, pages 1692-1699, Vancouver, Canada.

(Herlocker et al., 2004): Herlocker, J., Konstan, J., Terveen, L., Riedl, J. (2004). “Evaluating Collaborative Filtering Recommender Systems”. ACM Transactions on Information Systems, 22(1), pages 5-53.

(Hevner et al., 2004): Hevner, A.R., March, S.T., Park, J., Ram, S. (2004). “Design science in information system research”. MIS Quarterly, 28(1), pages 75-105.

(Hubaux & Heymans, 2009): Hubaux, A., Heymans, P. (2009). “On the evaluation and improvement of feature-based configuration techniques in software product lines”. In ICSE 2009, pages 367–370. IEEE.

(Jannach et al., 2010): Jannach, D., Zanker, M., Felfernig, A., Friedrich, G. (2010). “Recommender systems: an introduction”. Cambridge University Press.

(Janota et al., 2010): Janota, M., Botterweck, G., Grigore, R., Marques-Silva, J. (2010). “How to complete an interactive configuration process?”. In SOFSEM. Springer Verlag, pages 528-539.

(Kang et al., 1990): Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, S. (1990). “Feature – Oriented Domain Analysis (FODA) Feasibility Study”. Technical Report CMU/SEI-90-TR-21. Software Engineering Institute, Carnegie Mellon University.

(Kitchenham, 2004): Kitchenham B. (2004). “Procedures for performing systematic reviews”. Technical report, Keele University and NICTA.

(Koren, 2010): Koren, Y. (2010). “Factor in the Neighbors: Scalable and Accurate Collaborative Filtering”. ACM Transactions on Knowledge Discovery from Data (TKDD).

(Koshy et al., 2011): Koshy, E., Koshy, V., Waterman, H. (2011). “Action research in healthcare”. Thousand Oaks, CA: Sage.

(Linden et al, 2003): Linden, G., Smith, B., York, J. (2003). “Amazon.com recommendations: item-to-item collaborative filtering”. IEEE Internet Computing, pages 76-80.

(March & Smith, 1995): March, A.T., Smith, G.F. (1995). “Design and natural science research on information technology”. Decision Support Systems, 15(4), pages 251-266.

(Mazo et al., 2014): Mazo, R., Dumitrescu, C., Salinesi, C., Diaz, D. (2014). “Recommendation Heuristics for Improving Product Line Configuration Processes”. In Recommendation Systems in Software Engineering. Robillard M, Maalej W., Walker R. and Zimmermann T. (Eds.), Springer-Verlag Berlin Heidelberg.

(Mazo, 2011): Mazo, R. (2011). “A generic approach for automated verification of Product Line Models”. (Thèse). Université Paris1 Panthéon–Sorbonne, Paris, France.

(Metzger et al., 2007): Metzger, A., Pohl, K., Heymans, P., Schobbens, P.Y., Saval, G. (2007). “Disambiguating the documentation of variability in software product lines: A separation of concerns, formalization and automated analysis”. In 15th IEEE International Requirements Engineering Conference.

(Meyer, 2012): Meyer, F. (2012). “Recommender systems in industrial contexts”. (Thèse). University of Grenoble, France.

(Mirzadeh & Ricci, 2007) : Mirzadeh, N., Ricci, F. (2007). “Cooperative Query Rewriting for Decision Making Support”. Journal of Applied Artificial Intelligence. 21(10), pages 895—932.

(Mirzadeh et al., 2005): Mirzadeh, N., Ricci, F., Bansal, M. (2005). « Feature Selection Methods for Conversational Recommender Systems”. In IEEE International Conference on e-Technology, e-Commerce, and e-Service (EEE’05), Hongkong, pages 772-777.

(O'Brien, 2001) : O'Brien, R. (2001) “An Overview of the Methodological Approach of Action Research”. Roberto Richardson (Ed.) Theory and Practice of Action Research, Brazil, Universidade Federal da Paraíba.

(Oufaida & Nouali, 2009): OUFAIDA, H., NOUALI, O. (2008). « Le filtrage collaboratif et le web 2.0. Etat de l'art». Document numérique, 2008/1, Vol. 11, pages 13-35.

(Pazzani & Billsus, 2007) : Pazzani, M., Billsus, D. (2007). “Content-Based Recommendation Systems”. In The Adaptive Web: Methods and Strategies of Web

Personalization, Lecture Notes in Computer Science, Vol. 4321. Springer-Verlag, Berlin Heidelberg New York.

(Pazzani & Billsus, 1997): Pazzani, M. and D. Billsus. (1997). “Learning and revising user profiles: The identification of interesting web sites”. Machine Learning, Vol. 27, pages 313-331.

(Peffers et al., 2007): Peffers, K., Tuunanen, T., Chatterjee, M.A, Rothenberger, S. (2007). “A design science research methodology for information systems research”. Journal of Management Information Systems 24(3), pages 45--77.

(Pohl et al., 2005): Pohl, K., Böckle, G., van der Linden, F. (2005). “Software Product Line Engineering – Foundations, Principles, and Techniques”. Springer, Heidelberg.

(Pu & Chen, 2008): Pu, P., Chen, L. (2008). “User-Involved Preference Elicitation for Product Search and Recommender Systems”. AI Magazine, 29(4), pages 93—103.

(Rabiser et al., 2007) : Rabiser, R., Grünbacher, P., Dhungana, D. (2007). “Supporting Product Derivation by Adapting and Augmenting Variability Models”. In Proceedings of the 11th International Software Product Line Conference, pages 141–150. IEEE.

(Rao & Talwar, 2008): Rao, N.K., Talwar, V.G. (2008). “Application domain and functional classification of recommender systems a survey”. In Desidoc journal of library and information technology, 28(3), pages 17-36.

(Resnick et al., 1994) : Resnick, P., Iakovou, N., Sushak, M., Bergstrom, P., Riedl. J. (1994). “GroupLens: An open architecture for collaborative filtering of netnews”. In Proceedings of the Computer Supported Cooperative Work Conference.

(Runeson et al., 2012): Runeson, P., Höst, M., Rainer, A. Regnell, B. (2012). “Case Study Research in Software Engineering: Guidelines and Examples”. Wiley.

(Sabin & Weigel, 1998): Sabin, D., Weigel, R. (1998). “Product configuration frameworks - a survey”. IEEE Intelligent Systems and their Applications 13(4), pages 42-49.

(Salinesi et al., 2011): Salinesi, C., Mazo, R., Djebbi, O., Diaz, D., Lora-Michiels, A. (2011). “Constraints: the Core of Product Line Engineering”. In the Fifth IEEE International Conference on Research Challenges in Information Science. Pages 29-38).Guadeloupe-French West Indies, France. PA: IEEE Press.

(Salinesi et al., 2010): Salinesi,C., Mazo, R., Diaz, D., Djebbi, O. (2010). “Solving Integer Constraint in Reuse Based Requirements Engineering”. In 18th IEEE International Conference on Requirements Engineering. Sydney, Australia. PA: IEEE Press.

(Salinesi & Mazo, 2010) : Salinesi, C., Mazo, R. (2010). « Application engineering with reuse ». Paris: Université Paris1 Panthéon-Sorbonne.

(Salton, 1989) : Salton, G. (1989). "Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer". Addison-Wesley, Reading, MA.

(Sarwar et al., 2001): Sarwar, B., Karypis, G., Konstan, J., Reidl, J. (2001). "Item-based collaborative filtering recommendation algorithms". In Proceedings of the 10th international conference on World Wide Web, pages 285-295.

(Shardanand & Maes, 1995): Shardanand, U., Maes, P. (1995). "Social Information Filtering: Algorithms For Automating Word Of Mouth". In Proceedings of the Human Factors In Computing Systems Conference.

(Sinnema & Deelstra, 2006): Sinnema, M., Deelstra, S. (2006). "Classifying variability modeling techniques". Information and Software Technology, 49(7), pages 717–739.

(Susman, 1983): Susman, G. I. (1983). "Action Research: A Sociotechnical Systems Perspective". G. Morgan (Ed.). London: Sage Publications.

(Susman & Evered, 1978): Susman, G.I., Evered, R.D. (1978). "An Assessment of the Scientific Merits of Action Research". Administrative Science Quaterly, Vol. 23, pages 582-603.

(Thüm et al., 2009): Thüm, T., Batory, D., Kastner, C. (2009). "Reasoning about edits to feature models". In International Conference on Software Engineering (ICSE).

(Tiihonen & Felfernig, 2010): Tiihonen, J., Felfernig, A. (2010). "Towards RecommendingConfigurable Offerings". International Journal of Mass Customization, 3(4), pages 389-406.

(Tran & Cohen, 2000): Tran, T., Cohen, R. (2000). "Hybrid Recommender Systems for Electronic Commerce". In Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical Report WS-00-04, AAAI Press.

(Triki et al., 2014): Triki, R., Mazo, R., Salinesi, C. (2014). "Combining configuration and recommendation to enable an interactive guidance of product line configuration". In Gérald KEMBELLEC, Ghislaine CHARTRON and Imad SALEH (Ed.). Recommender Systems. pages 135-154. Paris-France: Wiley-ISTE.

(Webster & Watson, 2002): Webster, J., Watson, R. (2002). "Analyzing the past to prepare for the future: Writing a literature review". MIS Quarterly,26(2), pages xiii–xxiii.

(Wieringa, 2014): Wieringa, R. (2014). "Design Science Methodology for Information Systems and Software Engineering". Springer.

(Wieringa & Morali, 2012): Wieringa, R., Morali, A. (2012). “Technical action research as a validation method in information systems design science”. Design science research in information systems. Advances in theory and practice, 7286, pages 220-238. Springer Berlin Heidelberg.

(Witten & Frank, 2000): Witten, I. H., Frank, E. (2000). “Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations”. Morgan Kaufmann, San Francisco.

(White et al., 2009) : White, J., Doughtery, B., Schmidt, D., Benavides, D. (2009). “Automated reasoning for multi-step software product line configuration problems”. In Proceedings of the Sofware Product Line Conference, pages 11–20.

(Wohlin et al., 2012) : Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., Wesslén, A. (2012). “Experimentation in software engineering”. Springer Heidelberg New York Dordrecht London.

(Yang, 1999): Yang, Y. (1999). “An Evaluation of Statistical Approaches to Text Categorization”. Information Retrieval, 1(1), pages 67-88.

(Yu et al., 2004): Yu, K., Schwaighofer, A., Tresp, V., Xu, X., Kriegel, H.-P. (2004). “Probabilistic Memory-Based Collaborative Filtering”. IEEE Transactions on Knowledge and Data Engineering, 16(1), pages 56-69.

(Zaid et al., 2008) : Abo Zaid, L., Houben, G-J., De Troyer, O., Kleinermann, F. (2008). “An OWL-Based Approach for Integration in Collaborative Feature Modelling”. Workshop on Semantic Web Enabled Software Engineering.

(Ziadi et al., 2005): Ziadi, T., Jézéquel, J-M. (2005). “Manipulation de lignes de produits logiciels : Une approche dirigée par les modèles”. In 1ère Journées sur l'Ingénierie Dirigée par les Modèles (IDM’05), Paris.

(Ziadi, 2004) : Ziadi, T. (2004). “Manipulation de Lignes de Produits en UML”. (Thèse). Université de Rennes 1.

(Ziadi et al., 2003) : Ziadi, T., Jézéquel, J-M., Fondement, F. (2003). “Product line derivation with uml”. In Proceedings Software Variability Management Workshop, University of Groningen Departement of Mathematics and Computing Science.

(Zhang et al., 2002): Zhang, Y., Callan, J., Minka, T. (2002). “Novelty and redundancy detection in adaptive filtering”. In Proceedings of the 25th Annual International ACM SIGIR Conference, pages 81-88.